Distributed Document-Oriented Process Management in Healthcare

---

Verteiltes Dokumenten-orientiertes Prozessmanagement im Gesundheitswesen

Der Technischen Fakultät der
Universität Erlangen-Nürnberg
zur Erlangung des Grades

DOKTOR-INGENIEUR

vorgelegt von

Christoph P. Neumann

Erlangen – 2012

Impressum

Bibliographische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibiliographie; detailierte bibliographische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

Dissertation Universität Erlangen-Nürnberg, 2012

Distributed Document-Oriented Process Management in Healthcare
von Christoph P. Neumann

Christoph P. Neumann • Bielefelder Str. 38a • D-90403 Nürnberg
Internet: http://www6.cs.fau.de/people/cpn/ • E-Mail: c.p.neumann@gmail.com

Als Dissertation genehmigt von
der Technischen Fakultät der
Universität Erlangen-Nürnberg

Tag der Einreichung:	06. 07. 2012
Tag der Promotion:	16. 11. 2012
Dekanin:	Prof. Dr.-Ing. habil. Marion Merklein
Berichterstatter:	Prof. Dr.-Ing. habil. Richard Lenz
	Prof. Dr. biol. hom. habil. Hans-Ulrich Prokosch

Zusammenfassung

Organisationsübergreifende Zusammenarbeit zwischen Ärzten erlangt zunehmend an Bedeutung. Dabei besitzen Ärzte typischerweise autonome Anwendungssysteme zur Unterstützung von internen Prozessen. Es ist unrealistisch anzunehmen, dass Unterstützung übergreifender kooperativer Abläufe durch eine Art homogenes System erreicht werden könnte, das in jeglichen Praxen und Krankenhäusern vorinstalliert sein müsste. Traditionelle aktivitätsorientierte Modelle zur Prozessunterstützung lösen organisationsübergreifende Herausforderungen an die Systemintegration nicht.

Ziel dieser Arbeit ist es den Informationsaustausch zwischen streng autonomen Organisationen im Gesundheitswesen zu ermöglichen. Den Schwerpunkt bildet die Vermittlung zwischen ambulanter und stationärer Versorgung unter besonderer Berücksichtigung der bestehenden papier-basierten Arbeitspraxis. In Situationen die eine organisationsübergreifende Zusammenarbeit über regionale Grenzen hinaus erfordern ist es notwendig die Systemfunktionalitäten zur Prozessunterstützung von den bestehenden Anwendungssystemen zu entkoppeln. In dieser Arbeit wird das α-*Flow*-System vorgestellt. Es ermöglicht ad-hoc Kooperationen mit Hilfe von aktiven Dokumenten ohne zuvor die lokalen Anwendungssystem zu integrieren. Eines der zugrunde liegenden Prinzipien, das dabei näher untersucht werden wird, ist die Trennung von Inhalten und Entscheidungsfindung von Koordination.

Eine verteilte Fallakte namens α-*Doc* wird dazu eingesetzt zusammenarbeitende Prozessbeteiligte zu koordinieren. Die Verwendung dieser Fallakten benötigt keine vorinstallierten Systembausteine, wodurch Informationsaustausch völlig ad-hoc ermöglicht wird. Die Fallakte beinhaltet sowohl die verteilte Prozessstruktur in Form eines organisationsübergreifenden Therapieplans als auch beliebige elektronische Dokumente mit medizinischem Inhalt, die zwischen den Beteiligten ausgetauscht werden. Zur Veranschaulichung des Konzepts wird ein organisationsübergreifender Anwendungsfall verwendet, dazu wird der kooperative Behandlungsprozess von Brustkrebsfällen analysiert.

Schlagwörter: *Prozessunterstützung, Gesundheitswesen, Fallakten, aktive Dokumente, inter-institutionell, Systemintegration, Dokumenten-orientiert, Versionsverwaltung, verteilte Datenverwaltung, Content-Management, inhaltsorientiert, datengetrieben*

Abstract

Inter-institutional cooperation among physicians becomes increasingly important. Yet, it is unrealistic to assume that cooperation can be supported via a homogeneous system that is pre-installed in every organization. Instead, physicians will typically have their own autonomous systems that support internal processes. Traditional activity-oriented workflow models do not resolve inter-institutional integration challenges.

The goal of the proposed solution is to provide information exchange between strict autonomous healthcare institutions, bridging the gap between primary and secondary care, following traditional paper-based working practice. In large-scale inter-institutional scenarios, it is necessary to decouple cooperation functionality from the existing applications. This thesis presents the *α-Flow* approach for distributed process management, which enables ad hoc cooperation via active electronic documents without the need to integrate local systems. The rationale behind separating content, decision support, and coordination work will be explained.

A distributed case file, the *α-Doc*, is used to coordinate cooperating parties. Using this case file does not require any preinstalled system components, so genuine ad hoc information interchange is enabled. The case file contains both the distributed process schema in form of a collective therapy plan as well as arbitrary content documents that are shared among the cooperating parties. To illustrate the approach an inter-institutional use case is provided by cooperative breast-cancer treatment.

Keywords: *process support, healthcare, case handling, workflow engine, active documents, inter-institutional, system integration, document-oriented, versioning, distributed data management, content management, artefact-centric, data-driven*

Acknowledgements

> “He who has begun
> has half done.”
>
> (Horace, c. 21 BC, Epistles)

I wish to express my gratitude and thanks to Prof. Richard Lenz for the opportunity to work on his staff on many interesting topics during these past few years. He always supported my work and he provided many useful comments on my research papers and my thesis. His ideas and open-minded personality made him an invaluable contributor to my project.

I am very grateful to Prof. Klaus Meyer-Wegener. He holds the chair and provides an inspiring and familial environment to all of us. As a member of the “Studienkommission Informatik”, I had the chance to work with him on various study affairs. I highly esteem his profound knowledge, his fairness, and calm course of action as well as his idealistic engagement for the department and its students.

Many thanks go to all my current and former colleagues at the department of computer science. It was really a pleasure to work with you! Special thanks go to Florian Irmert, I enjoyed lecturing “eBT” very much together with you. Big thanks to Michael Daum, Robert Nagy, Frank Lauterwald, Juliane Blechinger, and Thomas Fischer for several years of companionship and for providing various valuable ideas and suggestions. I would also like to mention Julian Rith and Niko Pollner who provided me with a last-minute review and proofreading of the thesis. Extra thanks go to Dr. Marcus Meyerhöfer, Dr. Bernd Hardung, and PD Dr. Gabriella Kókai as former mentors. In addition, I would like to mention our caring system administrators Roswitha Braun and Ursula Stoyan, who always provided us with a well-running system environment. Finally, I wish Philipp Baumgärtel, Gregor Endler, and Johannes Held success in their projects as they continue with Prof. Lenz on our shared efforts in the healthcare domain.

Furthermore, I am grateful to many former colleagues and mentors at the sd&m AG, today a company of Capgemini. Jakob Boos as a dedicated and wise technologist inspired me as well as Dr. Dirk Gernhardt and Michael Hauser as thorough system architects. Also, I had the honour to be working under some great managers: Andree Sturm and Ulrich Bonfig. Thank you for your encouragement and support, I have tried to direct

my own students by your role model. Thanks also to my co-workers Ute Walther-Maas, Sebastian Thiebes, and Dr. Martin Kronenburg as well as Jochen Weber, Christian Hinken, and Bernd Tophoven. I have valued our exchange of experience and our fruitful discussions very much. During my time at the sd&m all of you made me a better software engineer and prepared me both for the technical and administrative project lead of my many software projects to come at the university.

All my former students are dear to me; they contributed their passion, countless lines of code, and a lot of debugging to my work. Special thanks to Florian Rampp who became a dear friend as well as Anelyia Todorova, Peter Schwab, Florian Wagner, Christian Hunsen, Scott Hady, and Andreas Wahl; both for their contributions to this work and the good relationship. Peter Schwab has recently become a colleague at the institute and I wish him best of luck with his own thesis.

Above all, I will greatly miss Prof. Hartmut Wedekind, his enthusiasm and our discussions. Furthermore, I am grateful for the friendship with Dr. Franz Lankes, Dr. Michael Klemm, Dr. Dominic Schell, and Dr. Szilvia Zvada; with all of them, I had a great number of happy coffee breaks at the department. Big thanks to my friends Georg Blaschke and Bernd Haberstumpf for the many experiences that we shared in our ERA editor project on requirements specification. Special thanks go to Alexander von Gernler for your invited talks in eBT, for educating me in the arts of Scrum, and for your friendship.

In addition, I am much obliged to Dieter Gawlick and Ralf Müller from Oracle for their interest in my work as well as Dr. Volker Stiehl from SAP for his visit and his work about composite application systems. Special thanks go to Dr. Falk Langhammer, the inventor of Ercatons and founder of the Living Pages Research GmbH. Falk, I believe in organic programming and admire your vision for the future of programming platforms.

I am very grateful to Nina Nieschler for your encouragement and love. Your patience with me in the last months of writing this thesis has been a great support. Finally, this work is dedicated to Dr. med. Helmut Neumann who as a gynaecologist explained breast cancer treatment to me and Rita M. Neumann who survived breast cancer and familiarized me with the patient perspective. Coincidentally both are also the best parents in the world! Without your perpetual support none of this would have been possible. Thank you!

Christoph P. Neumann

Bibliographic Notes

Preliminary versions of parts of this work have also been published in the following research papers:

[1] Christoph P. Neumann and Richard Lenz. 'Distributed Ad Hoc Cooperation in Healthcare'. In: *Post-Proceedings of the Joint Int'l Workshops on Process-oriented Information Systems in Healthcare and Knowledge Representation for Healthcare (ProHealth'12 / KR4HC'12) in conjunction with the 10th Int'l Conf on Business Process Management (BPM'12)*. Accepted for publication. Post-Proceedings of ProHealth'12 / KR4HC'12 will be published as part of Lecture Notes in Artificial Intelligence (LNAI) series. Springer, 2012.

[2] Christoph P. Neumann and Richard Lenz. 'The alpha-Flow Approach to Inter-Institutional Process Support in Healthcare'. In: *International Journal of Knowledge-Based Organizations (IJKBO)* 2.4 (2012), pp. 52–68.

[3] Christoph P. Neumann, Scott A. Hady, and Richard Lenz. 'Hydra Version Control System'. In: *Proc of the 10th IEEE Int'l Symposium on Parallel and Distributed Processing with Applications (ISPA-12)*. Madrid, Spain, July 2012.

[4] Christoph P. Neumann, Andreas M. Wahl, and Richard Lenz. 'Adaptive Version Clocks and the OffSync Protocol'. In: *Proc of the 10th IEEE Int'l Symposium on Parallel and Distributed Processing with Applications (ISPA-12)*. Madrid, Spain, July 2012.

[5] Andreas M. Wahl and Christoph P. Neumann. 'alpha-OffSync: An Offline-Capable Synchronization Approach for Distributed Document-Oriented Process Management in Healthcare'. In: *Lecture Notes in Informatics (LNI) Seminars 11 / Informatiktage 2012*. Ed. by Ludger Porada. Gesellschaft für Informatik e.V. (GI). Mar. 2012.

[6] Christoph P. Neumann, Florian Rampp, and Richard Lenz. *DEUS: Distributed Electronic Patient File Update System*. Tech. rep. CS-2012-02. University of Erlangen, Dept. of Computer Science, Mar. 2012.

[7] Christoph P. Neumann, Peter K. Schwab, Andreas M. Wahl, and Richard Lenz. 'alpha-Adaptive: Evolutionary Workflow Metadata in Distributed Document-Oriented Process Management'. In: *Proc of the 4th Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'11) in conjunction with the 9th Int'l Conf on Business Process Management (BPM'11)*. Clermont-Ferrand, FR, Aug. 2011.

[8] Aneliya Todorova and Christoph P. Neumann. 'alpha-Props: A Rule-Based Approach to 'Active Properties' for Document-Oriented Process Support in Inter-Institutional Environments'. In: *Lecture Notes in Informatics (LNI) Seminars 10 / Informatiktage 2011*. Ed. by Ludger Porada. Gesellschaft für Informatik e.V. (GI). Mar. 2011.

[9] Christoph P. Neumann, Thomas Fischer, and Richard Lenz. 'OXDBS – Extension of a native XML Database System with Validation by Consistency Checking of OWL-DL Ontologies'. In: *Proc of the 14th International Database Engineering & Applications Symposium (IDEAS'10)*. Montreal, QC, CA, Aug. 2010.

[10] Christoph P. Neumann and Richard Lenz. 'The alpha-Flow Use-Case of Breast Cancer Treatment – Modeling Inter-Institutional Healthcare Workflows by Active Documents'. In: *Proc of the 8th Int'l Workshop on Agent-based Computing for Enterprise Collaboration (ACEC) at the 19th Int'l Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2010)*. Larissa, GR, June 2010.

[11] Christoph P. Neumann and Richard Lenz. 'alpha-Flow: A Document-based Approach to Inter-Institutional Process Support in Healthcare'. In: *Proc of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'09) in conjunction with the 7th Int'l Conf on Business Process Management (BPM'09)*. Ulm, DE, Sept. 2009.

[12] Christoph P. Neumann and Richard Lenz. 'A Light-Weight System Extension Supporting Document-based Processes in Healthcare'. In: *Proc of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'09) in conjunction with the 7th Int'l Conf on Business Process Management (BPM'09)*. Ulm, DE, Sept. 2009.

[13] Christoph P. Neumann, Stefan Hanisch, Bernhard Schiemann, and Richard Lenz. 'OXDBS – Erweiterung einer nativen XML-Datenbank um die Validierung und Konsistenzprüfung gegen eine OWL-Ontologie'. In: *Tagungsband der 54. GMDS-Jahrestagung*. Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS). Essen, DE, Sept. 2009.

[14] Christoph P. Neumann, Florian Wagner, and Richard Lenz. 'XdsRig – Eine Open-Source IHE XDS Testumgebung'. In: *Tagungsband der 54. GMDS-Jahrestagung*. Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS). Essen, DE, Sept. 2009.

[15] Christoph P. Neumann, Florian Rampp, Michael Daum, and Richard Lenz. 'A Mediated Publish-Subscribe System for Inter-Institutional Process Support in Healthcare'. In: *Proc of the 3rd ACM Int'l Conf on Distributed Event-Based Systems (DEBS 2009)*. Nashville, TN, USA, July 2009.

In cooperation with other members of our research group I published on several other topics. Thus, I am co-author in the following publications but they do not contain parts of this work:

[16] Frank Lauterwald, Christoph P. Neumann, Richard Lenz, Anselm G. Jünemann, Christian Y. Mardin, Klaus Meyer-Wegener, and Folkert K. Horn. *The Erlangen Glaucoma Registry: a Scientific Database for Longitudinal Analysis of Glaucoma*. Tech. rep. CS-2011-02. University of Erlangen, Dept. of Computer Science, Dec. 2011.

[17] Thomas Fischer, Michael Daum, Florian Irmert, Christoph P. Neumann, and Richard Lenz. 'Exploitation of Event-Semantics for Distributed Publish/Subscribe Systems in Massively Multiuser Virtual Environments'. In: *Proc of the 14th Int'l Database Engineering & Applications Symposium (IDEAS'10)*. Montreal, QC, CA, Aug. 2010.

[18] Holger von Jouanne-Diedrich, Juliane Blechinger, Christoph P. Neumann, Stefan Schwarz, and Richard Lenz. 'Integration verteilter und heterogener Configuration-Management-Datenbanken'. In: *Informatik-Spektrum* 33 (4 2010). Ed. by Arndt Bode, pp. 351–362. ISSN: 0170-6012. DOI: 10.1007/s00287-009-0398-6.

[19] Florian Irmert, Frank Lauterwald, Christoph P. Neumann, Michael Daum, Richard Lenz, and Klaus Meyer-Wegener. 'Semantics of a Runtime Adaptable Transaction Manager'. In: *Proc of the 13th Int'l Database Engineering & Applications Symposium (IDEAS'09)*. Cetraro, IT, Sept. 2009.

[20] Florian Irmert, Christoph P. Neumann, Michael Daum, Niko Pollner, and Klaus Meyer-Wegener. 'Technische Grundlagen für eine laufzeitadaptierbare Transaktionsverwaltung'. In: *Tagungsband der 13. Fachtagung Datenbanksysteme für Business, Technologie und Web (BTW'09)*. Münster, DE: Gesellschaft für Informatik e.V. (GI), Köln, Germany, Mar. 2009.

[21] Marcus Meyerhöfer and Christoph Neumann. 'TestEJB – A Measurement Framework for EJBs'. In: *Proc of the 7th Int'l Symposium on Component-Based Software Engineering (CBSE'04) in conjunction with the 26th Int'l Conf on Software Engineering (ICSE'04)*. Ed. by Ivica Crnkovic. Vol. 3054. Lecture Notes in Computer Science. Edinburgh, UK: Springer, Berlin, DE, May 2004, pp. 294–301.

Table of Contents

I

Prologue

1 Introduction

> “Concise, in-depth, and ab initio!”
>
> (Prof. Augustus Van Dusen
> by J. Futrelle and M. Koser)

The patient treatment process is increasingly evolving from isolated treatments towards continuous episodes that incorporate multiple organizationally independent institutions and different professions. One characteristic of this process is that both the order of treatment steps and the number of involved parties are usually not known in advance as they are largely dependent on the preceding course of the treatment.

For example, during post-operative care of a breast cancer case it might occur that a patient gets yellowish skin pigmentation. These symptoms could be caused either by liver metastases or by gallstones. Thus, emergent changes in patient conditions cause ad hoc demand for diagnostic measures. The subsequent therapeutic measures for each diagnosis are utterly different.

Evolutionary workflow approaches are required that enable cooperation and coordination among the participants. It is essential to deal with the semantic and technical heterogeneity of the systems at the participating sites because different information systems and internal workflows are used.

This thesis presents an approach that implements distributed case files that enable ad hoc cooperation. The case files provide distributed document-oriented process management and data distribution. To illustrate the approach an inter-institutional use case is provided by cooperative breast-cancer treatment.

1.1 Fundamentals

This section provides a survey of fundamentals of healthcare. It is intended for computer scientists without prior knowledge about the domain of healthcare as well as for health professionals without prior knowledge about systems integration and workflow management. The survey considerations are a premise for understanding the subsequent

motivation of this work. Readers with suppositional knowledge about healthcare, medical guidelines, and clinical pathways as well as continuity of care may skip to the general motivation in section 1.2 on page 32.

1.1.1 Inadequate Information

The problem of inadequate information in healthcare is well-documented by meta-analyses, for example from the Institute of Medicine (IOM) by Kohn et al. [1, 2]. An exemplary study is a systems analysis of adverse drug events by Leape et al. [3]: 18% of the medical errors were associated with inadequate availability of patient information. Missing information is, for example, prescriptions or the results of laboratory tests. Inadequate availability of information as a major cause for medical errors is aggravated both by the aging of western society and by the increasing number of multiple healthcare providers that are involved in a treatment.

The *aging of western society* affects the public health sector because chronic diseases and multimorbidity—like cancer, diabetes, asthma, and cardiac insufficiency—become the focus of interest. For example, chronic diseases account for three-quarters of U.S. health expenditures with eight out of ten older individuals challenged by one or more chronic diseases [4]. In Germany 43% of all costs are caused by those greater 65 years of age, again with chronic diseases and multimorbidity being the key aspect [5]. In an analysis of multimorbidity and its effect on healthcare costs [6], the prevalence of multimorbidity was two thirds in those greater 50 years of age, and statistical evidence is provided that healthcare costs are significantly increased among patients with multimorbidity.

Chronic diseases and multimorbidity requires an *increasing number of healthcare providers* in comparison to common diseases (cf. [4–6]). In addition, the advance in medicine leads to a growth of specialization [7], which is another cause for the increasing number of involved parties concerning any patient's treatment. This increasing number is also known as *system fragmentation* [8]. The term "system fragmentation" not only means that patients receive care from multiple providers but it also contains the criticism that they rarely coordinate the care they deliver. System fragmentation in healthcare is countered by the concepts of continuity of care and integrated care, which will be explained in section 1.1.6. To improve the treatment quality and to avoid unnecessary costs, an effective information and communication technology is vital for the support of inter-institutional patient treatment (e.g., [9, 10]).

1.1.2 Supply Chains in Healthcare

Supply[1] chains in healthcare are centred on a patient and his or her physicians. Office-based physicians are collectively described as the *primary care*. The *secondary care* adds hospitals, laboratories, pharmacies, and ancillary medical institutions as participants of the medical supply chain. Accompanying participants are the health insurance funds (German "Krankenkassen") and the associations of statutory health insurance physicians (German "Kassenärztliche Vereinigungen"). All participants are outlined in figure 1.1.

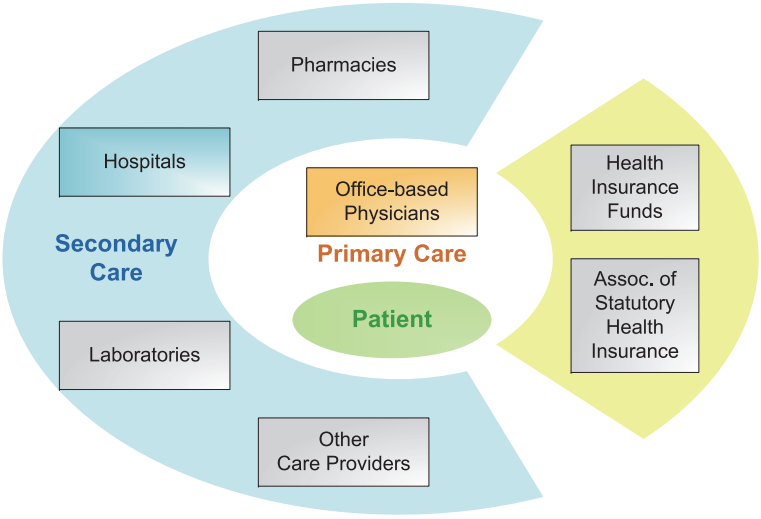

Figure 1.1: Participants in healthcare supply chains (adapted from Sippel [11])

The term "healthcare supply chain" is meant in a comprehensive way: it includes services, manufacturers, and the public sector, from a medical, insurance, fiscal, or industrial perspective, either with a direct, indirect, or transitive involvement. All medical parties as a whole are described as *health service providers* (German "Leistungserbringer"); amongst others these are the doctors, nurses, laboratories, and pharmacists. The insurance parties, amongst others, provide services like billing, accounting, and clearance.

In addition to figure 1.1, the healthcare industry is another important stakeholder. The biggest one is the industry for pharmaceuticals (German "Arzneimittel"). Yet, there are other healthcare industries like the one for assistive equipment (German "Hilfsmittel"), for example, the industry for prosthetic devices.

1 The term "supply" is sometimes misunderstood because in English it is a homonym (one word, two meanings). First meaning stems from economics, as in "supply and demand"; it can be interpreted as an *economic ability to deliver* (German "Angebot"). The second meaning stems from supply chain management, as in "to supply" or "supplier"; it characterizes environments with a set of suppliers and *actual delivery* (German "Lieferung").

In loose analogy to a picture by Kotler [12, p. 32] about factors of influence in marketing management, I have arranged the factors of influence in healthcare supply chains in form of figure 1.2. The patient remains at the centre. The basic measurement types of medical care that patients receive create the second onion layer. The third onion layer comprises all involved parties for operative purpose. Primary care and secondary care represent the health service providers. The accompanying participants appear just as well. The colours that have been used in figure 1.1 are reused for visual continuity. The IT administration that is in charge of the software applications and the technical infrastructure is a forth party that is operatively involved. All parties of the third onion layer interact with each other, which is illustrated by the bidirectional arrows. The fourth layer is the one that provides the rectangular shape. It comprises fractions with strategic influence. The outside frame surrounds the healthcare supply chain with universal factors.

A comprehension of healthcare supply chains is imperative for understanding healthcare processes. If a referral of the patient treatment occurs between primary care and secondary care it generally transfers responsibility for process coordination from one physician to another. The synopsis on general factors is relevant since the factors influence system fragmentation, continuity of care, and integrated care. A patient-individual

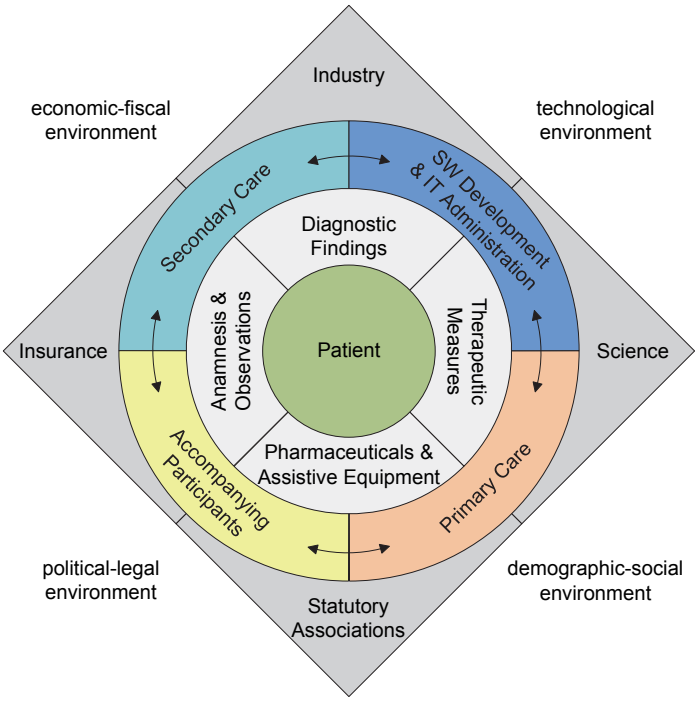

Figure 1.2: Factors of influence in healthcare supply chains

supply chain is not a predefined set of institutions. Healthcare processes result in ad hoc cooperation between independent organizations.

1.1.3 The Diagnostic-Therapeutic Cycle

A basic model to describe medical processes is the traditional diagnostic-therapeutic cycle by Jan van Bemmel [13]. He discerned three stages of all human activities (observation, reasoning, action) and applies them to medicine in form of 1) observation, 2) diagnose, and 3) therapy. An important aspect in medicine is that physicians are not confronted with the issue of solving an abstract or general problem, but are confronted with solving the problems of individual patients; these problems can only partly be generalized. For example, a patient tells his or her history (anamnesis interview), the physician collects more data (e.g., during a physical examination, by laboratory tests, radiology, etc.), comes to a conclusion (hypothesis about the disease) and possibly even a diagnosis (identified disease), and can ideally prescribe a therapy (e.g., a clinical operation or a drug prescription).

In the original diagram from 1997 the patient and the three stages had formed one simple cycle. Yet, the diagnostic and therapeutic activities are often cycled iteratively because it may appear that hypotheses have to be refined or altered. Thus, [14] provides an adopted version of the diagnostic-therapeutic cycle, which is illustrated in figure 1.3. For the purpose of decision making, the principle of Evidence-Based Medicine (EBM) [15] has the goal to make medical knowledge as explicit as possible. EBM will be discussed in the next subsection. However, the medical knowledge that influences decision making is still primarily *tacit knowledge*[2] and for the most part not explicit (e.g., [18]).

Particularly chronic diseases require a long-cyclic exchange of patient information between all involved health service providers from different institutions [19]. At present, this is done by paper documents like referral vouchers and discharge letters. Yet, discharge letters are frequently missing or are insufficient in detail [20], either because they are not written by the physicians at all or because they are not available to all involved parties [21]. Repeated anamnesis interviews are necessary to compensate for missing document interchange between health service providers. Redundantly applied diagnostic methods by each distinct institution are the norm. As simple as order entry and

2 The term “tacit knowledge” has been popularized by Nonaka and Takeuchi since 1991 in [16] and in their well-known book [17]. The latter also described the Socialization, Externalization, Combination, Internalization (SECI) model for organizational learning and for transforming tacit knowledge into explicit knowledge.

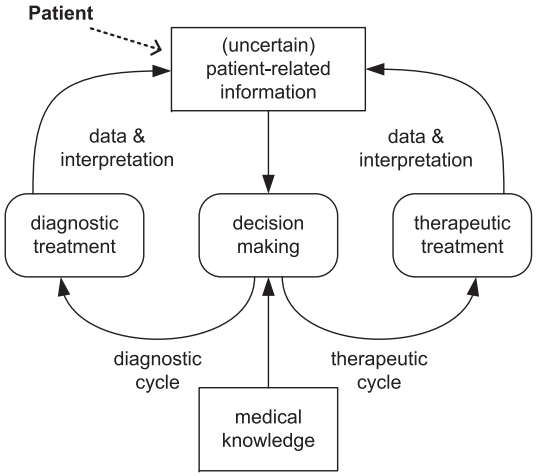

Figure 1.3: The diagnostic-therapeutic cycle (adapted from Lenz [14])

result reporting may seem, they are still one of the most important issues in healthcare information systems (e.g., [22, 23]).

From the perspective of possible IT support for diagnostic-therapeutic process, there are three opportunities: 1) to provide patient-related information at the point-of-care by means of system integration [24], 2) to make explicit knowledge available following the principles of EBM, and ideally 3) to assist to make informed decisions, e.g., by recommendations from artificial intelligence provided by a decision support system.

A comprehension of the therapeutic-diagnostic cycle is imperative to understand workflow support for medical processes. In a nutshell, the therapeutic-diagnostic cycle demonstrates that the next step is not known in advance. Needlessly to say, physicians usually have interim plans for some of the next measures. However, there is a feedback loop. Therapeutic measures can change the patient condition. New symptoms might appear. An interim therapy plan becomes deprecated and new cycles are necessary. Frequent invalidation of plans is characteristic to medical processes.

1.1.4 EBM and Guidelines

The principle of *Evidence-Based Medicine (EBM)* was introduced by Sackett [15]. Expert *guidelines* are a key concept in result to the implementation of EBM. Both are well-known terms in healthcare.

Sackett defined EBM as "the conscientious, explicit, and judicious use of current best evidence in making decisions about the care of individual patients" [15]. EBM can easily be misunderstood twofold: by healthcare professionals and by non-healthcare persons.

Healthcare professionals, for some time, tended to misunderstand EBM as "cookbook medicine". Non-healthcare persons hardly understand EBM until the specific meaning of "evidence" is further clarified.

Medicine is an empirical science. The term "evidence" most often means published clinical studies (e.g., a "clinical trial") both on treatments or on diagnostic tests. EBM requires studies and their publications to provide statistical validity (e.g., by randomised trials), currency, and peer-reviews on the results. In addition, EBM demands studies to assess both the benefits and the risks—because the basic question "Does a treatment more good than harm?" involves many subtleties[3].

Before the stir of EBM, medical decision making did primarily rely on individual clinical expertise. After graduation and approbation, the individual empirical experience from his or her cases constitutes each physicians expertise. A large part of medical knowledge is not explicit but tacit [26]. Thus, in extension to his or her intuitive expertise, EBM emphasises on the necessity for "selective patient-driven searching, appraisal, and incorporation of the best available evidence" [15]. Because EBM demands patient-driven searching, it sometimes was defamed as "cookbook medicine". Yet, Sackett stressed that "external clinical evidence can inform, but can never replace, individual clinical expertise". The exigency for a careful balance between external evidence and individual expertise is summarized by Sackett himself:

> "Good doctors use both individual clinical expertise and the best available external evidence, and neither alone is enough. Without clinical expertise, practice risks becoming tyrannised by evidence, for even excellent external evidence may be inapplicable to or inappropriate for an individual patient. Without current best evidence, practice risks becoming rapidly out of date, to the detriment of patients."

The problem with EBM is obvious: clinicians are busy and reading time is scarce. Thus, EBM ultimately triggered the formation of meta-analyses and systematic reviews of evidence-providing studies and publications. Today, medical experts are selected from authoritative national associations and form panels in order to provide periodically updated references on the available evidence for a dedicated type of disease. The expert panels assess the meta-analyses and surveys in order to provide *(medical) guidelines*. To be effective, a guideline must be easily accessible to the physicians who shall implement the guideline in practice. Ideally, the guideline content should be embedded into clinical work practice and the physicians should not need to explicitly look it up [27].

3 An introduction to the grading of the quality of evidence in healthcare may be gained from Oxman in [25].

In Germany, the authority for guidelines is the "Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften" (AWMF). There are three different levels of "maturity" for guidelines: S1 to S3. The maturity is based on requirements on the methodical quality [28]. In a nutshell, S1 provides only an informal consensus of an expert group, S2 implies a formal method of consensus finding, and S3 implies formal methods of consensus finding and the systematic evaluation of research results. The majority of guidelines are S1. In the United States (U.S.), there are several grading systems from different authorities, ranging from three levels to five levels.

Guidelines distinguish different options for therapy and narrate on them by summarizing related medical publications and providing their citations. Thus, they provide indicators and criteria for qualifying a disease and relevant medical constellations. Then, they advise on diagnostic tests and therapeutic measures that had been proven effective (by evidence) for certain constellations. In conclusion, guidelines provide decision support. They quite literally include lists of medical "if...then..."[4] recommendations. For illustrative purposes, I suggest the S3 guideline for breast cancer treatment by Kreienberg et al. [29][5]. As illustration, a recommendation in sect. B 2.3 of that guideline document states[6]: "If micro calcium carbonate is existent then a stereotactically controlled vacuum biopsy should be applied, preferably. [plus some cite on Nothacker et al. in 2007]".

Notably, the example demonstrates a subtle but important factor in phrasing. It says SHOULD and this is not by coincidence, hardly any recommendation in the entire guideline document ever says MUST. Thus, all "if...then..." expressions are recommendations and they are not strict rules. Even S3 guidelines still represent a consensus, only. Not even all the involved experts must have agreed, actually. Hence, is a guideline legally binding? Not by definition, not even S3 guidelines. However, are S3 guidelines used in court and by court-appointed appraisers as a reference? Naturally. Still, from the perspective of computer science, if guideline recommendations should be considered as rules then only with a tacitly implied fuzziness. An explicit quantification of the underlying probabilities is sometimes available from the referenced publications, for example, in terms of incidence or prevalence. However, it is not necessarily extracted from the publications. In case of the German breast cancer guideline, the quantified probabilities are not directly available within the scope of the guideline document.

4 For example (but not limited to), recommendations like "If you already know this and that factor, then there is yonder evidence [cite here, cite there] that says that your treatment shall or should involve this and that."

5 e.g., `http://www.leitlinien.de/mdb/downloads/dkrebsg/mammakarzinom_lang.pdf`

6 German original wording: "Bei Vorliegen von Mikrokalk soll vorzugsweise die stereotaktisch gesteuerte Vakuumbiopsie eingesetzt werden."

A comprehension of EBM and guidelines is imperative to understand the influence of decision support on workflow support for medical processes. Guidelines list options for activities. They do not describe the sequence of the activities. Usually, several necessary activities would be indicated in parallel. Guidelines do not describe physicians or roles. They tacitly assume that the reader knows who would be required for an upper abdomen sonography or a bone scintigram. The guidelines do not tell whether a measure can be applied with the available resources at the local institution or whether a referral is necessary. As an illustration for the absence of many workflow relevant aspects in medical guidelines sect. C 4.2 of the breast cancer guideline might be read. Guidelines are a recommendation for medical decisions; they do not coordinate a workflow.

1.1.5 Clinical Pathways

Clinical pathways are a key concept to traditional workflow support in clinical environments. Process support in healthcare has to manage both medical processes (MedProcs) and organizational processes (OrgProcs) [14, 27].

OrgProcs concern the planning, preparation, and appointing of procedures and not the execution of medical procedures. OrgProcs are basically agnostic to medical knowledge. They involve the patient admission and discharge, the order preparation and arranging of visits by physicians or nurses, or the physical transport of patients. OrgProcs also include transliteration processes from dictates to reports, which require secretaries that have been trained in the necessary terminology and frequently used abbreviations. Other well-established OrgProcs appear in association with imaging encounters or lab tests. A final example would be the scheduling processes for operation rooms in a hospital, which can be quite complex and commonly require at least one dedicated administrator. In any clinical scenario, the IT challenge for OrgProcs is up to cross-departmental system integration.

MedProcs concern the decision about medical procedures and their actual execution with patient-individual adjustment. The medical guidelines are primarily concerned with medical knowledge and suggestions for decision making within MedProcs. In a nutshell, guidelines are the “What?, Why? & With what?” whereas clinical pathways are also concerned with “When?, Where?, Who? & With whom?”[7]. Thus, a clinical pathway extends the MedProc by its implied OrgProc for a specific setting.

Clinical pathways are generally modelled within a single institution and include site-specific resource and time requirements. Thus, each clinical pathway is an organization-

7 If automation is intended then workflows require also a formal model of the “How?”.

specific consensus between process participants. It mediates between physician concerns and nursing concerns and it mediates between different departments. Figure 1.4 outlines the association between guidelines and clinical pathways.

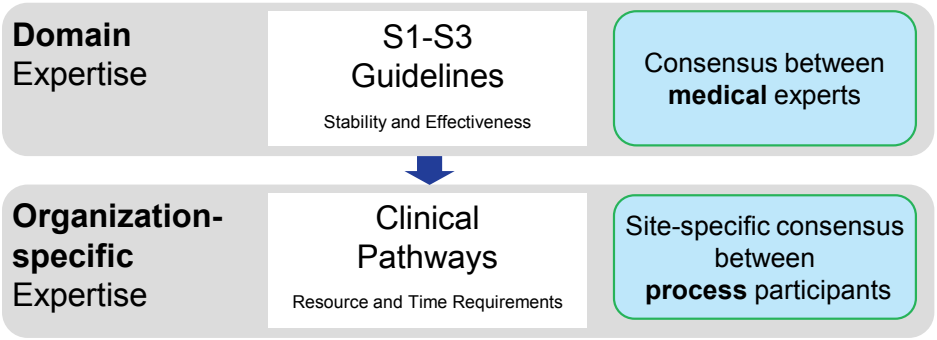

Figure 1.4: From guidelines to clinical pathways (adapted from Lenz [14])

From the perspective of workflow modelling, a major problem is that the many medical options result in a combinatorial explosion. Thus, a clinical pathway not necessarily includes all guideline recommendations but models a coarse-grained schema of the medical activities or uses a selected treatment option to illustrate the implied organizational activities by reference example. Such a reference example is like a generic "workflow intention" that is used for graphical visualization purposes. For instance, a clinical pathway would include the activity "biopsy" in its workflow schema but usually not the guideline-based specialization "stereotactically controlled vacuum biopsy", explicitly. Needlessly to say, if a clinical path would incorporate all relevant guideline recommendations then it could maximize guideline compliance. A comprehensive work on incorporating knowledge management facilities into workflow management platforms has been done at our institute by Sascha Müller in his PhD thesis [30]. In addition, his thesis includes a good introduction on the distinction between workflow instance, workflow schema, and workflow language.

A clinical pathway models a clinical process based on site-specific best practice. Aside from combinatorial problems, a clinical pathway does not necessarily cover the whole guideline because some measures might not be applicable in a particular setting, for example, with the locally available technical devices or human expertise. Thus, an actual clinical path is commonly only concerned with a subset of the guideline complexity. The extend of the subset varies from institution to institution. In conclusion, there commonly is a deviation between guidelines and clinical pathways.

The individual patient treatment will deviate from a clinical path by two reasons. The first reason had already been outlined by the EBM discussion: "individual clinical expertise always overrules external clinical evidence" because medical problems can only

partly be generalized and a cookbook medicine is inappropriate. The clinical pathway may be customized for a patient due to his or her condition as well as to his or her individual chances and risks. This customization results in an individual treatment plan. From a computer science perspective, such individualization is still in parts an open issue and is not generally available in commercial workflow management platforms.

The second deviation takes place between the individual treatment plan and the actual process. The patient condition changes and the originally individualized treatment plan must sometimes be adapted due to emergent contraindications and alerts. Lenz provides examples for the deviation of the individual treatment plan from the actual treatment process in [31]. A first example is patient vitals that drop out of bound. Another example is a patient who is scheduled for Magnetic Resonance Imaging (MRI) and has an implanted pacemaker. Such MRI-preventing factor is not necessarily checked until MRI preparation by a nurse who prepares the patient and runs through a checklist. In conclusion, Lenz distinguishes between four levels of healthcare processes: the guidelines, the clinical pathways, the individual patient treatment plan, and the actual patient treatment process. An illustration of these four levels is provided by figure 1.5.

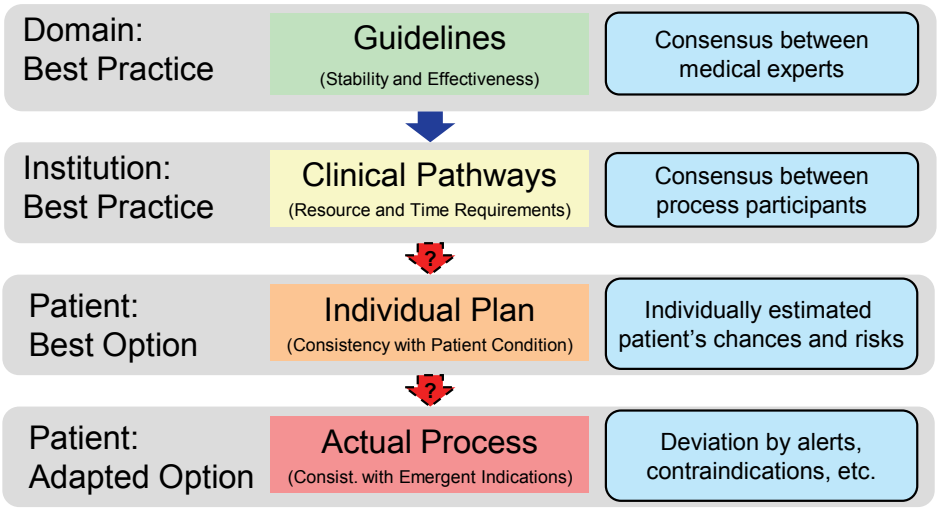

Figure 1.5: Workflow refinement in healthcare (adapted from Lenz [14])

A comprehension of clinical pathways is imperative to understand current limitations of workflow support for medical processes. A workflow schema that models a clinical pathway does only consider the generic best practice. The frequent invalidation of plans that are known from the diagnostic-therapeutic cycle render them insufficient for patient-individual treatments. The key for success of workflow management is not to support the predicted processes but the real ones.

1.1.6 Towards Inter-Institutional Scenarios: Continuity of Care and Integrated Care

The *continuity of care* can mean different things to different types of caregivers. It is generally concerned with the degree the care is coherent and linked; it primarily focuses on the personal relationships between patients and providers that connects care over time and bridges discontinuous events. Fletcher et al. [32] provided an overview of the different conceptions in 1984; with different components being, for example, number of providers and visits, distribution of visits among providers, and episodes of illness.

Based on the total number of providers and visits, Fletcher defines *continuous care* simply as the per cent of visits to the primary physician; as an index number, it is a baseline. In addition, *coordinated care* is introduced; it depends on three conditions: a) written evidence that the other physician was aware of the primary physician's involvement, and that b) the primary physician arranged visit to the other physician or knew about it beforehand; or c) the primary physician was aware of the patient's visit to the other physician after the visit. In overall, coordinated care is the per cent with the presence of (a) and (b or c). Finally, Fletcher introduces *integrated care* and formally defines it as the per cent with continuity or coordination present.

From the perspective of IT support for integrated care there are two opportunities: once again 1) to provide patient-related information by means of system integration, but also 2) to provide coordination information by means of participant management, treatment planning, and treatment history.

1.2 Motivation

This section provides current problems and emerging requirements of information exchange and process support in healthcare. Readers with suppositional knowledge about systems integration, workflow management, and case handling in healthcare may skip to the consolidated problem statement in section 1.3 on page 41.

1.2.1 Unsolved System Integration

In order to foster continuous and coordinated care, the inter-institutional cooperation needs to bridge the current information gap between institutions of the primary and secondary care [33]. Considering the comprehensive medical supply chain, integration

between autonomous information systems of multiple participants is still unsolved in diagnostic-therapeutic processes (e.g., [34, 35]).

Such effort must not apply regional standards, as it is done in a Regional Healthcare Information Network (RHIN) [36], but transregional standards. Many sites have their own Electronic Medical Record (EMR) [37] for storing patient related information, which typically can be extracted on demand. Yet, it is unclear how these systems scale and how direct communication between institutions can be effectively supported in large-scale scenarios. Independent (personal) Electronical Health Record (EHR) [37] or (lifelong) Personal Health Record (PHR)[8] [41] are discussed as a basis for inter-institutional cooperation. Yet, despite of existing standards like openEHR, reality is far from the vision of seamless record exchange, and IT-support for inter-organizational patient treatment processes is an open issue.

Still, an IT infrastructure for healthcare networks needs to cope with the heterogeneity of the systems at different sites. It is required to preserve inter-institutional data consistency. It needs to encourage semantic agreements among the communicating parties. Ideally (but not necessarily), such an agreement could be based on standard domain ontologies, which contribute to semantic compatibility.

With no available common standard that is sufficiently established for wide-area networks in environments with initially unknown participants, an a priori semantic agreement cannot be assumed. Thus, an IT infrastructure needs to apply a "pay-as-you-go" approach in form of a principle known as *dataspaces* by Franklin et al. [42] since 2005. In the dataspace method, in contrast to traditional database methods, the data participants do not pursue data integration but *data co-existence*. The objective of data co-existence is to provide basic functionality, in form of a Dataspace Support Platform (DSSP), over a whole set of distributed data sources irrespective of the degree of data integration. Further data integration, in order to provide advanced functionality, can and should be applied in a demand-driven manner. Another "pay-as-you-go" approach is known as "Data First, Structure Later. Maybe." It is a motto for favouring unstructured content models. It was popularized by Java Content Repository (JCR) specification leader David

8 The Medical Records Institute (Newton, MA, USA) distinguishes five stages of development for electronic records in medcine and healthcare. The classification is based on Waegemann [38] in 1996. The original publication is hardly available, thus, the German audience is best advised to consult Lehmann's handbook of medical informatics [39, sect. 12.3.1]. EMR and EHR are the most prominent stages, whereupon EMR is stage three and the EHR is stage five. As of late, the PHR is considered as a sixth stage (cf. [40]). It implicates lifelong documentation. In addition, all five stages by Waegemann implicate that the records are maintained by health service provider organisations, in contrast, PHRs are considered to be maintained by individuals. Notably, the five stage classification does not provide a notion for case files.

Nuescheler as his "Rule #1" since 2007 in [43][9]. The dataspace approach is concerned with distributed systems and favours deferred data integration, in contrast, Nuescheler's principle favours deferred data schematization, in general.

1.2.2 Inter-Institutional Problems

Accomplishing information exchange in distributed healthcare scenarios requires the integration of heterogeneous systems and must respect the strict autonomy of the pre-existing systems in different institutions. The message-based approach to system integration that is favoured in traditional EMR and EHR environments is unsuitable for ad hoc cooperation between independent organizations because it needs pre-established message standards. A platform is required that favours local autonomy over central hegemony. In particular, strict autonomy of the institutions requires the abdication of central infrastructure like joint databases, transaction monitors, and central context managers. Closely integrated data management with terminological standards for contents is unrealistic in this scenario.

A conceptual change from messages to documents is provided by Health Level 7 (HL7) v3 Clinical Document Architecture (CDA). CDA provides a framework for XML-structured medical documents. An overview on the application of CDA for cross-institutional data exchange is, for example, provided by Müller et al. [33].

Electronic documents, in contrast to messages, can be stored and interpreted independently from the originating system. Document content specifications like the Continuity of Care Document (CCD) have been developed, a U.S.-specific standard, which is based on HL7 and focuses on document-oriented medical content types.

Such content structure standards, however, do not consider process history or coordination information. Their primary function is standardized semantic tagging of information within electronic documents—not the infrastructure for managing such documents.

1.2.3 From Bilateral Information Exchange to Information Distribution

Information interchange by letters is the way of traditional cooperation. In terms of liability, a referral delegates responsibility from one institution to another. Yet, establishing and managing dynamic teams of cooperating specialists becomes more and

9 For his motto, David Nueschler in [43] references Stefano Mazzocchi's online article [44] in 2005.

more important [45]. For some years now, in Germany, the treatment of breast cancer is organized by accredited in-station breast cancer treatment centres cooperating with manifold accredited partners like oncologists, radiologists, and the post-operative care [46].

In cooperation, participants are organizationally independent and the degree of work-product interdependence remains low, although the information exchange is based on a mutual benefit. In contrast, collaborative scenarios require a team to achieve *collective results* that the participants would be incapable of accomplishing when working alone. Although collaborating partners remain organizationally independent, the degree of work-product interdependence is substantial.

Cooperative treatment scenarios in healthcare can be considered as distributed medical treatment processes for treating complex chronic diseases and multimorbidity. Collaborative treatment scenarios, in contrast, can be described as physician teams from different institutions that physically or virtually meet. Typical examples for collaboration can be found in the field of telemedicine. Prokosch et al. [47] provide a survey on exemplary scenarios like teleconsultation/teletherapy, teleradiology, telepathology, or telesurgery. All of these are based on synchronous communication between clinicians. Telemedicine is still rarely used (cf. [47]).

Current inter-institutional healthcare scenarios are primarily cooperative ones and not collaborative ones (e.g., [48, p. 23] or [49, p. 3]). Most of breast cancer treatment is based on cooperation (e.g., [50]). Yet, the tumour conference during primary therapy is an example for true collaboration. Both cooperative and collaborative scenarios change the requirements for the availability of patient information and necessitate advanced information exchange models.

1.2.4 Cases as Workflows in Healthcare

There is no agreed definition for the term “case”. Nevertheless, it has been used in the context of insurance, law court, or healthcare for some time. *Connie Moore*, the vice president of Forrester Research, Inc., provided a description (as it is portrayed by Alison Clarke during a presentation in 2010 [51]):

> “A pattern of work that’s highly dynamic, in which a group of people systematically collaborate in structured and ad hoc way on a case folder using business process management, document management and collaboration tools.”

And *David Roe*, a journalist and staff reporter of the web magazine *CMS Wire*, provided another description in 2009 (cf. [52]):

> "A case is basically a grouping of information that a user works on. It could be a legal brief, a customer, a location or a query. Essentially, the user is pulling together all the information they need into a single location to work on—or manage—this 'case'.
> Case Management is generally a collaborative process with a number of contributing users, and a single overall manager. Tasks, data objects, documents and even processes can be added at any time, depending on a change in the status of the case in question, all of which need to be traced and tracked if a successful resolution is to be achieved, with a strong emphasis on information sharing."

In healthcare, a patient with a health problem who visits healthcare professionals at different sites spans a distributed workflow—with activities that include all kinds of diagnostic or therapeutic treatments. Such a workflow is considered as a case [53], and process management in cooperative healthcare scenarios requires to handle these cases. In comparison with the four levels of workflow refinement in healthcare (cf. sect. 1.1.5, fig. 1.5), the case implies the forth level, the actual treatment process. Recently, Swenson introduced the term Adaptive Case Management (ACM) in his well-received book [54]. ACM also emphasizes the need to provide support for the real/unpredictable process instead of an ideal/predicted process when knowledge workers, of any domain, deal with cases. In conclusion, cases are a conceptual counterpart to clinical pathways, from a certain perspective.

Medical cases and insurance cases must be strictly distinguished. This thesis will focus on medical cases derived from the diagnostic-therapeutic cycle. In comparison to the overall healthcare supply chain, the case participants form only a subset. Medical cases include only the health service providers. Notably, this still does not include only *directly* involved parties like doctors, nurses, and pharmacists but also *indirectly* involved parties like laboratories. Some doctors can also be involved indirectly without direct patient contact, for example pathologists and in breast cancer cases also some oncologists who participate in a consultative role.

1.2.5 Traditional Workflow Approaches and Unsolved Issues

The dominant approach for formal workflow models in computer science is the activity-oriented workflow paradigm, e.g., Petri Nets with states and transitions or Business Process Modelling Notation with actors and activities. The characterization of tasks or

actions by pre-conditions, post-conditions, and possible exceptions is dominant. The state-of-the-art section 3.2 will discuss activity-oriented workflow approaches in more detail.

To put workflows into practice, two distinct levels need to be distinguished: specification and automation. For specification purposes, prospective conceptualization of the activities is necessary in form of predefined semantically rich workflow schemas. For task automation purposes, a system service implementation must be provided. Enactment engines provide process control and monitoring in order to direct firmly rather than to assist wisely. Because workflow systems are server-centric, decentralized workflows are still a challenge. Ad hoc workflows are not traditionally considered, and initially unknown sets of actors, states, and transitions are not supported.

1.2.6 Content-Oriented Workflow Paradigms

The content-oriented workflow paradigms implement coordination based on state-changes of an artefact life-cycle model. The goal of content-oriented workflow models is to articulate workflow progression by the presence of content units (like data-records/objects/resources). Some content-oriented workflow approaches provide a life-cycle model for content units, such that workflow progression can be qualified by conditions on the state of the units. Most approaches are research and work in progress and the content models and life-cycle models are more or less formalized.

The term "*content-oriented workflows*" is an umbrella term for several scientific workflow approaches. The current set of approaches consists namely of "data-driven", "resource-driven", "artifact-centric", and "object-aware" research. The state-of-the-art section 3.3 will discuss these approaches in more detail. In the final analysis, the meaning of "*content*" ranges from basic data attributes to coarse-grained documents. A general term, independent from a specific approach, is necessary to contrast the content-oriented modelling principle with traditional activity-oriented workflow models where a workflow is driven by a control flow and where the content production perspective is neglected or even missing.

As a disclaimer, it should be pointed out that the term "content-oriented workflows" is my own term and it does not appear in scientific publications until 2009, in one of my publications [55]. I did choose the term "content" to subsume the different levels in granularity of the *content units* in the respective workflow models. Furthermore, it was chosen to make a conscious association between the fields of *content management* and *workflow management*. Otherwise, both fields are only fractionally associated (e.g., [56, 57]). Both terms "artifact-centric" and "data-driven" would also be good candidates

for an umbrella term, however, each is closely related to a specific approach of a single working group. The group behind the artifact-centric approach (i.e. IBM Research) has generalized the characteristics of their approach and has used "information-centric" as an umbrella term in [58]. Yet, the term "information" is too unspecific in the overall context of computer science, thus, "content-oriented workflows" is considered as a good compromise[10]. In the end, this thesis will not provide an exclusive definition of the content-oriented workflow paradigm, however, a taxonomy to systematically characterize content-oriented workflow models will be derived, empirically, from the embraced approaches in section 3.4.5.

The object-aware workflow systems, the artifact-centric approach, and the data-driven process structures all represent advanced approaches to the content-oriented workflow paradigm. A basic example for content-orientation in processes is the write-and-review scenario: characterizations of content states are provided by editing states like "draft", "submitted", "revised", and "final" in a publishing scenario. Coordination actions can be triggered based on state-changes in the content life-cycle. In all approaches, the object, respectively the business artifact or data needs a structured and predefined content schema and content state model.

Just as the activity-oriented workflow approaches, the content-oriented workflow approaches require predefined semantically rich schemas, just of the content models instead of the control flows. Again, ad hoc workflows are not traditionally considered by content-oriented workflows, and initially unknown sets of actors, states, and transitions are not supported.

1.2.7 Problem-Oriented Medical Records

Recognizing healthcare processes in a content-oriented fashion has been considered particularly in clinical environments for about forty years. A corresponding conception is the Problem-Oriented Medical Record (POMR) [59], which is a method of organizing medical records that was introduced by Lawrence Weed in 1968. In order to understand the problem-oriented order, it is necessary to describe former ordering schemes. Hippocrates of Kos is credited with creating the first medical records, using a diary-style journal that records observations in chronological order, which is called a Time-Oriented

10 I later found out that the developers of the open-source Content Management System (CMS) Alfresco have also used the term "content-oriented workflows" since 2007 to paraphrase its basic write-and-review workflows (cf. sect. 3.3.1) in their system documentation. In conclusion, the term is used by Alfresco intuitively to generalize the characteristics of its workflow conception. I consider this earlier occurrence of the term (even if it is outside of the scope of scientific publication) as further evidence for its general fitness to characterize the underlying paradigm.

Medical Record (TOMR). The TOMR was the dominant style until the 19th century, which is documented by Musen and van Bemmel in "the history of the patient record" [13, pp. 100]. An electronic application of TOMR as the ordering scheme for an EMR is described by Fries in [60].

In the 19th century, as science and technology developed, diagnostic methods were refined, consequently, the fragmentation into different medical disciplines was amplified, and hence, cross-sectional treatments occurred more frequently. Accordingly, the Source-Oriented Medical Record (SOMR) became an alternative method for organizing the medical records. A SOMR orders the records according to the method by which they were obtained. For example, notes of visits, X-ray, blood tests, and other data become separate sections in the patient record. Commonly, the chronological order remains as the second sorting criterion in a SOMR.

In 1968, Weed revised the primary and secondary ordering of medical records, forming his POMR style. Weed argued to use a *problem list* as first-order structure, which contains a patient's current (i.e. *active*) and past (i.e. *inactive*) problems. A problem can be anything: a symptom, a physical abnormality, or any sign that has been observed of the patient. In addition to medical problems, the problem list may include social, psychiatric, or demographic problems. For example, a problem might be "chest pain", which might be diagnosed as pneumonia, initially, and later found to be a myocardial infarction (cf. [61]). To formulate a list of all the patient's problems is the initial challenge for a physician. The POMR problem list has significant model implications because it allows to follow the course of a problem in parallel with the patient's other problems or as a separate problem (cf. [62]). Consequently, some patient-related information will properly belong under more than one problem. The three TOMR, SOMR, and POMR styles of ordering are exemplified by Musen and van Bemmel [13, pp. 100]. The POMR style is well accepted in the scientific community (e.g., [63, 64]). However, many actual EMR systems remain to be organized in SOMR style by resembling the departmental organization of the institution because of accounting requirements (cf. [65]).

Following his pioneer publication in 1968, Weed refined his proposal in 1971 [66]. For each problem in the problem list there is a list of *SOAP notes*. SOAP in the context of POMR is an abbreviation for subjective, objective, assessment, and plan. The *subjective* is the patient's own complaints as reported by the anamnesis. The *objective* is an uninterpreted finding, like a physician's observation or the results of a diagnostic measure. The *assessment* is the interpretation, i.e. the physician's understanding of the problem. The *plan* is a follow-up advice, in form of general treatment goals or specific actions. These four categories become the basic sections of clinical notes, thus, they are the constitutive scheme of the SOAP format for clinical notes. SOAP-formatted notes of a POMR-ordered patient record are known as *progress notes*. The progress notes, as the

second order entries of an EMR, are expected to deal conceptually with problems and to display the clinical reasoning used.

In contrast to the majority of entries in a TOMR and SOMR patient record, the SOAP notes of POMR are intentionally interpretative notes. The SOAP principle emphasises on providing summaries of the most relevant issues instead of offering large collections of raw data records. The progress notes allow medical personnel "to act as a guidance system and follow the course of each problem, collecting more data base, reformulating and updating problems and respecifying the plans, each action dependent upon the course of the patient's problems" [62].

The POMR explicitly issues follow-up documentation and eases access to relevant information. In conclusion, the progress notes intend to directly reflect the successive measures indicated by the diagnostic-therapeutic cycle. Bayegan and Nytro in [67] argue that the POMR style fosters continuity of care in general. In a distributed healthcare scenario, when multiple health service providers are involved, the nature of the POMR becomes convenient because it allows new participants to follow the medical story of an overall treatment based on the filed content units. In conclusion, the POMR list of SOAP-formatted progress notes is a list of content units that articulate a medical process in a content-oriented fashion.

1.2.8 Case Handling

Case handling is a new paradigm for process support. It stems from Pallas Athena, Inc. for the FLOWer system [68]. In computer science it has been picked up by Reijers, Ritger, and van der Aalst [53]. Unlike traditional workflow management it is aimed at supporting a team of cooperating process participants in their decisions—rather than predefining process steps.

The core features that are defined by van der Aalst in [69] for the case handling paradigm are: (a) provide all information available, i.e. present the case as a whole rather than showing bits and pieces, (b) decide about activities on the basis of the information available rather than the activities already executed, (c) separate work distribution from authorization and allow for additional types of roles, not just the execute role, and (d) allow workers to view and add/modify data before or after the corresponding activities have been executed, e.g., information can be registered the moment it becomes available.

On a paradigm level, case handling explicitly argues the lack of usability of contemporary workflow management systems for healthcare processes. Yet, contemporary case handling systems focus on single institution scenarios, in healthcare mainly hospitals, and

technologically on centralized case handling system. Case handling for inter-institutional processes is an open issue.

1.3 Problem Statement & Objectives

When we started the project, my supervisor and I had the premise: "There is no adequate infrastructure to support an emergent willingness to cooperate in inter-institutional scenarios." This is true for healthcare and any other domain. Thus, two primal scientific questions have been the driver of this thesis: 1) "Can we develop an infrastructure that allows us to enable emergent ad hoc processes?" and 2) "To which degree can we establish information exchange without prior system integration?". Such an infrastructure must provide functionality beyond e-mail, being adapted to typical needs of cooperation.

In order to minimize the initial effort for establishing an information exchange between healthcare professionals willing to cooperate, we are looking for an evolutionary and decentralized approach. The traditional approach to manage such healthcare processes is based on paper documents with a dedicated semantics, such as a referral or a discharge letter. We adopt and extend this paper-based interaction paradigm to support more complex cooperation scenarios.

In order to foster the continuity of care, the inter-institutional cooperation needs to bridge the current gap between institutions of the primary and secondary care. Organizational independence requires adhering to the strict autonomy of the involved IT systems. Although comprehensive IT support for closely meshed treatment scenarios in general has unsolved legal boundary conditions, our approach outlines an architecture for distributed case files as a technical foundation. It is based on digital information units that are yielded from institutional EHR into a distributed publish-subscribe system that synchronizes the case files between participants.

A design principle is to aim for minimal standards in order to yield minimal requirements to the participating systems. Favouring local autonomy over central hegemony requires, for example, that distribution of information will not be enforced, but is voluntary and process participation can be supplemented on demand. Platform independence and the avoidance of vendor lock-ins require that the basic architecture is decoupled from any specifically instrumented middleware and off-the-shelf components. Loose coupling becomes an important system quality and will be discussed as a system design property in the methods chapter. Since autonomous systems are inevitable in heterogeneous environments, they must be coupled in adequate ways such that their autonomy is preserved. For example, it should be possible to add and remove participants without

any modification of other participants. Thus, without previously interconnecting two participants it should be possible to interchange information[11].

With *α-Flow*, the constitutive project of this thesis, I mean to provide case handling in distributed environments with an emphasis on document-oriented systems integration. I subsume the characteristic requirements, methods, and objectives of this intent under the term distributed Document-oriented Process Management (dDPM). Support for dDPM answers both primal questions. Thus, it means 1) to enable knowledge-driven ad hoc processes with an initially unknown set of activities and actors, and 2) to offer data exchange in inter-institutional environments in spite of prevalent system heterogeneity without prior system integration. *α-Flow* provides a prototypical implementation for dDPM.

There are several system design objectives of the proposed solution. The first technical objective is the abdication of any central server, like joint databases, transaction monitors, and central context managers, as adherence to the strict autonomy of the institutions. The second objective is the application of document-oriented integration with lightweight interfaces instead of service-oriented integration with semantically rich interfaces. Document-orientation favours local autonomy by adhering to the design goal of loose coupling. The third objective is to provide process status information that consists of a shared therapy plan and the process history. The forth objective is about participant management and to supply information about the pre-treatment or mutual treatment providers—this patient-related information is often missing and is considered one of the most important issues in inter-institutional healthcare scenarios [22]. Finally, a fifth objective is to provide process templating facilities for template creation with import/export of process structure and process-required roles.

1.4 Thesis Outline

The following section provides the scientific contribution. Then, a scope disclaimer will discuss important aspects that are not addressed by the remainder of this thesis. At the end, the structure of the thesis as a whole is outlined in graphical form.

11 Excluding considerations for a federated, large-scale *security* infrastructure that might still impose coupling on certain levels.

1.4.1 Scientific Contribution

This thesis provides results on several levels. In the domain of healthcare, it concerns the physician as a knowledge worker and medicine itself in form of modelling breast cancer treatment. In the domain of computer science, it concerns workflow management, distributed systems, content management, and adaptive-evolutionary systems.

The human-machine interaction perspective: In terms of user experience, the idea of active documents is reinterpreted in a specific shape. Similar to paper-based interaction, the active documents in this thesis embed all facilities for direct interaction with itself and the embedded pieces of information. An active document does not depend on another application, but is a lightweight application itself. It is a file on the user's desktop, can be replicated like common files, and is active only when opened by the user.

The medical perspective: In terms of treatment, the breast-cancer case is modelled. It is organized by accredited in-station breast cancer centres cooperating with partners like oncologists, radiologists, and the post-operative care. The challenge is to apply a document-oriented perspective on structuring the comprehensive process. In the end, a method to describe therapy plans results from this perspective such that their descriptions can be understood by both the doctors and the patients.

The workflow management perspective: In terms of process formalization, the idea of content-oriented workflows is explored. The challenge is to articulate workflow progression by the presence of content units and by conditions on the state of the units. In the end, the content-oriented workflow paradigm is combined with case handling. A process model is provided for knowledge-driven ad hoc processes in inter-institutional environments.

The distributed systems perspective: In terms of overlay networks, the active documents represent nodes in a peer-to-peer network for synchronization. All nodes will have an offline characteristic and usually no two peers are online at the same time. The challenge is to provide data synchronization that is offline-capable such that human workers are not entirely blocked in their work by selective conflicts. In the end, a protocol is provided that locally conducts synchronization operations such that eventual global consistency is achieved across distributed replicas.

The content management perspective: Healthcare processes are paper-based and each healthcare treatment step implicates a logical unit of electronic documents. Each document unit belongs to a organizational unit, thus, it is necessary to maintains an independent history for data provenance purposes. The autonomy for each unit must be ensured within an overall treatment episode's context. From a technical perspective,

multiple independent version histories are required within a single content repository. Furthermore, the necessity to differentiate versions based on their validity characteristics will be motivated. In the end, a version control system is implemented that provides multi-module histories and validity-based navigation.

The evolutionary information systems perspective: In terms of adaptiveness, an evolutionary information system[12] is required to provide support not only for planned change but also for emergent change. The challenge is to support process naturalism, i.e. to embrace the real/unpredictable course of action instead of any ideal/predictable. In the end, the entity-attribute-value data design approach and prototype-based programming concepts are applied. For example, to provide an adaptive-evolutionary status attribute model for document-oriented processes.

1.4.2 Scope Disclaimer

Several topics are not targeted by this thesis. The first domain is security both with a secure data exchange between systems and a security architecture of each system. Security is a major issue in IT infrastructures for healthcare. A brief but adequate tribute to security is mandatory. The second domain that will not be discussed beyond this section is human actor identification and classification. This includes both the patient identification as well as the organizational structures of healthcare institutions and physicians. Again, these are very important issues at large but these topics have their own extensive pool of available research, specifications, and products.

Security

The whole interaction and cooperation in healthcare is liable to many technical, organizational, economic, and legal factors. The legal boundary conditions are critical for the information provision and availability because warranty of data protection is essential for patient-related data. The research does not focus on Public Key Infrastructure (PKI), but relies on existing PKIs like the German "Elektronische Gesundheitskarte" (eGK).

The approach that will be discussed in this thesis will result in a distributed data architecture. The German boundary conditions in regard to security for distributed

12 An information system is defined by Heinrich [70] as being composed as a "Mensch-Aufgabe-Techniksystem", i.e. a medley of humans, assignments, and IT. Thus, an information system must not to be understood solely as a technical system but as a social-organizational-technical system. An evolutionary information system is defined by Lenz [71], essentially enabling the advancement of an organization with a well-defined purpose driven by organizational learning.

electronic data transfer in healthcare are described in the specifications of the eGK project: [72, p. 54] and [73, pp. 51–53]. Relevant protection targets are confidentiality, integrity, authenticity, non-repudiation, and availability. These concepts are described in the appendix (sect. A.1).

In the context of security, it is important to notice that several considerations do not arise with digital systems but occur with people and paper-based records. Thomas Rindfleisch describes breaches of confidentiality in healthcare (cf. [74]): accidental disclosure, insider curiosity, insider subordination, uncontrolled secondary usage, and outsider intrusion. Health information must be very carefully controlled. Technological systems introduce additional security risks and at the same time, technological interventions can improve security.

In contrast to distributed data architectures, there are other approaches with central storage. Ückert and Prokosch provide an overview of security-related concerns for a web-based EHR system in [75]. In the United States, several vendors are creating centralized solutions for inter-institutional purpose, mostly in web-based form, like Microsoft HealthVault[13], Dossia[14], PatientsLikeMe[15], and the hyped but already discontinued Google Health[16]. Instrumenting a central content storage in large-scale inter-institutional environments is risking an information leak that potentially involves all patients. Such is not comparable to any possible abuse scenario in today's paper-based infrastructure. No current healthcare institution stores information about as many patients as it is intended by any of these centralized platforms. In contrast, the distributed approach mirrors the current state in paper-based working practice and provides *information locality*: The patient information is available only to the directly involved healthcare systems. As a result, the consequences of a system security breach are limited by applying a distributed data architecture.

Besides eGK, there are competing PKI standards in healthcare, like PaDok [76] or the Audit Trail and Node Authentication (ATNA) platform [77] from the Integrating the Healthcare Enterprise (IHE) initiative. To secure the messaging with arbitrary PKI technology, for the synchronization of distributed patient files or case files, generalized PKI component integration would be required that is independent of the various PKI specifications. Such a PKI component remains an open issue. As a tribute to the security of electronic communication, message encryption based on OpenPGP [78] is integrated into the transfer facilities of the prototype to this thesis.

13 http://www.microsoft.com/en-us/healthvault/
14 http://www.dossia.org/
15 http://www.patientslikeme.com/
16 http://www.google.com/health

Beyond secure messaging and PKI considerations, the aspect of security in information systems and applications is much broader. A *security architecture*, cf. [79] is a detailed description of all aspects of a system that relate to security in regard to its implementation, along with a set of principles to guide the design. Security permeates different functional aspects, from file system handling and input/output to memory management and process control. A security architecture, for example, requires error handling strategies to be aware of security related exceptions because security checks may introduce new failures. For example, there might appear a new kind of system failure when a system is incapable to read a file even though the system recognizes that the file exists. Another common issue in security-related system design is securing temporary files in a scratchpad directory in multi-user environments.

The prototype to this thesis does not provide a security architecture. This stems from a pragmatic observation: the undergraduate students that are participating in prototype development are nearly overwhelmed with the complexity of the many functional aspects of a distributed healthcare-, content management-, and workflow-related system design. However, Morrie Gasser warns in [79] that it is fatuous to believe in "build it first, secure it later" because it usually involves great costs to retrofit security. Thus, the least thing that is required are the mentioned *principles to guide the design*. Gasser continues in her explanations that one important factor in achieving "a high confidence in the security of a system" is to "minimize and isolate security controls" by "minimizing the size and complexity of the security-relevant parts of the internal design". She motivates this with Saltzer and Schoeders principle of *economy of mechanism*, in their publication about "the protection of information in computer systems" [80]. In these terms, the prototype implementation to this thesis attempts compliance to this principle 1) by a high degree in modular system design, 2) by reducing the number of different types of security enforcement so that no proliferation of redundant security mechanisms emerges, and 3) by a coarse and hence intuitive granularity of access controls. Still, the remainder of this thesis will not testimony on security architecture concerns.

Identification and Classification of Human Actors

Two types of actors are involved in basic medical scenarios, the patient and the physicians. Patient identification concerns matching, tracking, merging, and de-duplicating patient identifiers. It is a big issue in medical informatics because of its impact on uncontrolled duplicate records or missing medical records. A related IT infrastructure is called Master Patient Index (MPI). There are several MPI specifications and vendors. For example, the IHE Patient Identifier Cross-referencing (PIX) specification [77] or the Object Management Group (OMG) Person Identification Service (PIDS) specification

[81]. Vendors are, for example, Orcle[17] and German promedtheus[18]. Any of these instrument hierarchical federation with central system nodes and are not applicable in genuinely distributed environments.

I supervised a master thesis (cf. [82]) that evaluated common MPI systems, in search of distributed peer-to-peer approaches. In conclusion, for large-scale scenarios, a loosely-coupled distributed patient identification service for inter-institutional purpose remains an open issue. The eGK probably "solves" the problem, in the national scope of Germany, by introducing a life-long national patient ID. In the prototype to this thesis, the patient ID is a placeholder that is filled by a Universally Unique IDentifier (UUID) [83, 84].

The second type of human actor is the knowledge worker as workflow participant, i.e. the physicians in healthcare. A comprehensive work on organization models for representing institutions, roles, and actors has been done at our institute by Christoph Bussler in his PhD thesis [85]. Modelling organizations can involve a great complexity; especially when internal processes are modelled and automatized it is often required to model a fine-grained representation of the specific organization structure. For inter-institutional purposes, we require a coarse-grained model. My references for contact information are address book standards like vCard [86] in which the institution is represented as a common string. My references for role information are paper-based referral vouchers; paper-based referrals include the required medical role again as a plain string. Besides the flat data type for institution and role, both data fields could either be used unstructured in free-form, or a controlled vocabulary could be applied to such a string field. The latter is preferable.

In order to support distributed actor identification as well as the semi-structuring of institution fields or role fields, I supervised the construction of a distributed institution management in form of a meta-data repository. An experimental system was implemented in the context of a bachelor thesis (cf. [87]). It applied graph structures to provide a basis for controlled vocabularies on both data fields. After the student's completion of his thesis, I put the approach on hold, to not diversify the combined efforts. My recommendation, today, would be the application of a W3C standard for basic taxonomies: Simple Knowledge Organization System (SKOS) [88]. In conclusion, the prototype to this thesis does not provide support for controlled vocabularies on institution or role fields but allows free-form usage. Placeholders for institution IDs and actor IDs are filled by UUIDs.

17 Sun MPI: `http://docs.oracle.com/cd/E19509-01/820-3377/ghbdc/index.html` which has not been renamed into Oracle MPI, yet.

18 promedtheus MPI: `http://promedtheus.de/`

1.4.3 Structure of the Thesis

The thesis is structured in four parts. An overview is illustrated in figure 1.6. The first part comprises the introduction with a survey on fundamentals, the outline on the project motivation, and the problem statement. It also contains the second chapter about methods, which explains both the project procedure and applied methods.

The second part is about inter-institutional processes and active documents. It first provides a survey on state of the art in chapter 3. This chapter will discuss healthcare standards for integration, workflow management approaches, and active documents. Two variations of workflow approaches will be discussed, the traditional activity-oriented approaches and recent content-oriented approaches. As a result, a taxonomy of characteristics will be devised for content-oriented workflows and active documents. Ideas from those domains are constitutive to dDPM. Accordingly, the dDPM conception is defined in two chapters. In chapter 4, a user story exemplifies the operative embedding and emphasises on the active document characteristics. In chapter 5, the content-oriented characteristics of inter-institutional treatment processes will be analysed in detail.

The third part is about the *α-Flow* system as the pilot implementation of the dDPM concept. Chapter 6 provides an architectural overview on implementing a distributed case file and case handling engine in form of an active document. Various methodical and technical aspects are discussed in chapter 7. The *α-Flow* implementation is of considerable size, thus, only issues of general interest will be illustrated. The third part concludes with chapter 8, which provides a technical evaluation of the design and implementation of *α-Flow*.

The fourth part concludes the thesis. It first provides a conceptual evaluation of *α-Flow* in chapter 9, which contains a comparative analysis with related approaches, summarizes the fitness for use, and provides a discussion of open issues and future work. The last chapter 10 finally gives a résumé on the findings and the contribution.

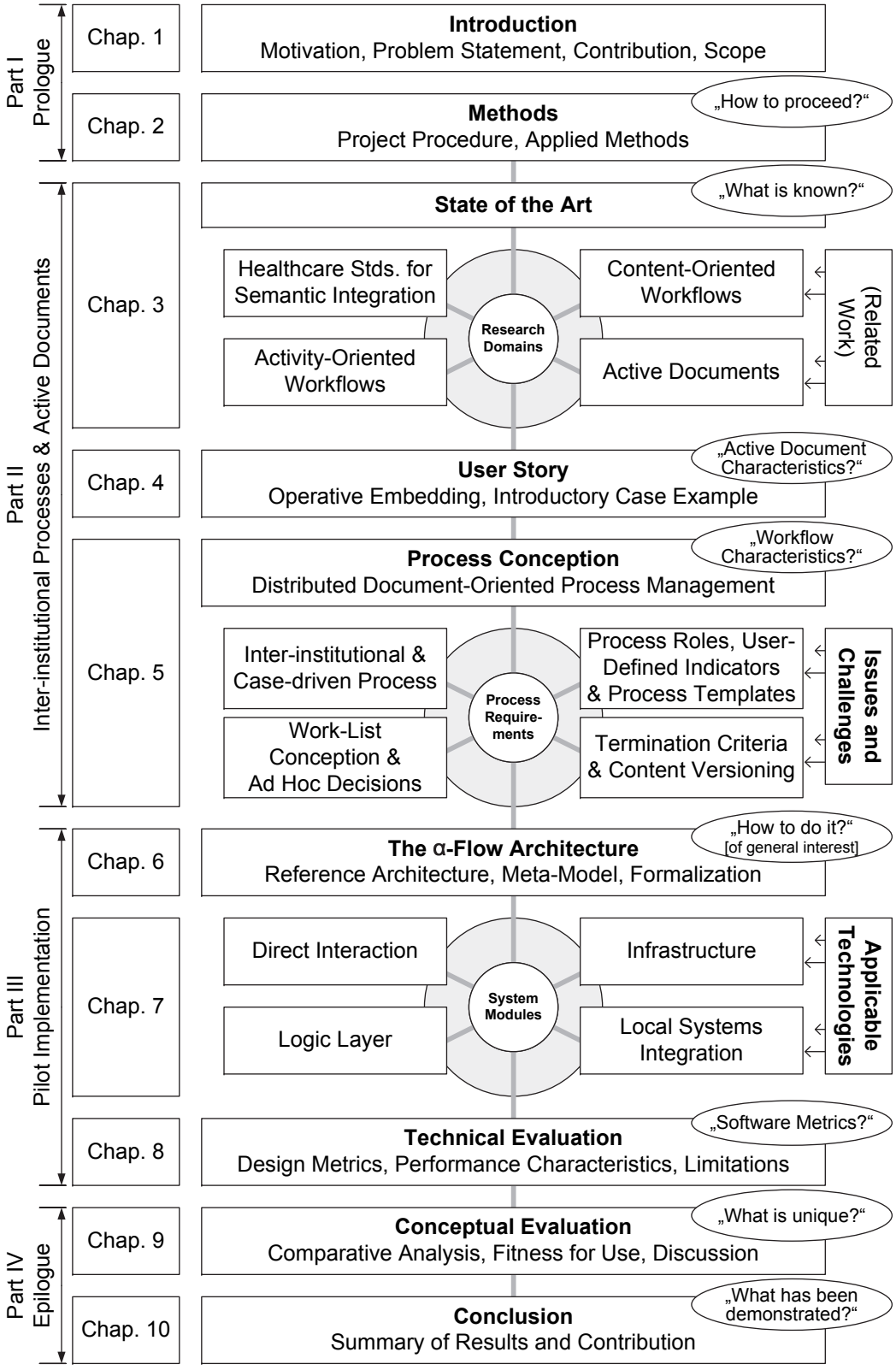

Figure 1.6: The structure of the thesis

2 | Methods

> “The science of planning lies in anticipating the hassles of execution.”
>
> (Luc Clapier de Vauvenargues)

This chapter is split into two separate parts. First, the project procedure will be outlined. It encompasses three major project phases and one interlude. The phases lead from the initial evaluation of existing standards, over two intermediate approaches and an interlude about the German “Elektronische Gesundheitskarte”, to the final *α-Flow* approach.

The second part describes scientific methods that have been applied to the system design. It discusses several aspects of integration, deferred design, prototype-based programming, loose coupling, sender-push vs. receiver-pull, separation of the concerns content/coordination/decision, cards-that-represent-tasks, and active documents. The last section concludes with an anticipatory outline of distributed case files via active documents. The structure of this chapter is outlined in figure 2.1.

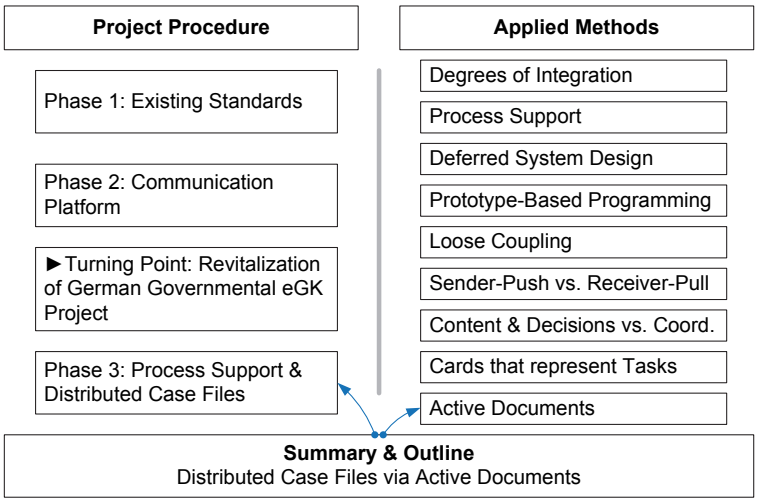

Figure 2.1: The structure of the methods chapter

2.1 Project Procedure

The project started with a working title "ProMed"[1] and with the objective to enable process support in heterogeneous and inter-institutional healthcare scenarios. Such scenarios concern adaptive-evolutionary information systems, which means that there are integration considerations implied that are explained in later sections in more detail. In summary, a system integration style that is based on electronic documents was favoured early on. The general idea was to take medical documents and somehow extend them with process information. An important design objective was to aim for minimal standards in order to yield minimal requirements to the participating systems.

2.1.1 First ProMed Phase: Existing Standards

The first ProMed efforts focused on semantic system integration and on evaluating existing standards for document interchange. For example, in healthcare there are the Health Level 7 (HL7) standards for messaging as well as the Cross Enterprise Document Sharing (XDS) standard from the Integrating the Healthcare Enterprise (IHE) initiative for document interchange. I will provide a more detailed discussion on available standards in sections 2.2.2 and 3.1.2. My initial conception about semantic integration was strongly influenced by my supervisor, Prof. Richard Lenz, in form of his habilitation treatise [71] and seminal article [89].

Eventually, in a student research project with Florian Wagner [90], we constructed the XdsRig that is an open-source IHE XDS test stand environment. The XdsRig was assembled with components from a National Institute of Standards and Technology (NIST) implementation of the IHE XDS Repository and Registry as well as with components from the Eclipse OHF projects for IHE XDS client actors. For test stand purposes we implemented a graphical XDS user client. I presented the XdsRig on the 54th annual German "Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie" (GMDS) conference [91].

Another consideration during this phase was about the content and its medical consistency. I did cooperate with the institute for artificial intelligence, i.e. Bernhard Schiemann, and our efforts resulted in the open-source Ontological XML Database System (OXDBS) system, based on the open-source database eXist-db[2]. I presented the

1 ProMed is an abbreviation for the German title "Prozessunterstützung von adaptiv-evolutionären Informationssystemen in der Medizin" that translates to "process support for adaptive-evolutionary information systems in healthcare".

2 http://exist.sourceforge.net/

OXDBS also on the 54th annual German GMDS conference [92] and later published it in some more detail (cf. [93]). In the OXDBS project, we extended a native XML database system with validation by consistency checking of OWL Description Language (OWL-DL) ontologies. In another project during this phase, I supervised a master thesis on the evaluation of Master Patient Index (MPI) systems (cf. [82]).

During this initial ProMed phase, the insight I gained in existing standards was that, unfortunately, any available standard in healthcare provides only a centralized integration architecture. There is no standard available that provides peer-to-peer document exchange for distributed inter-institutional healthcare environments. The next course of action, thus, was to create our own transfer platform for document interchange.

2.1.2 Second ProMed Phase: Communication Platform

The second phase of ProMed consists of two parallel efforts in order to enable inter-institutional document exchange: DMPS and DEUS. Both were student research projects; the former with Igor Engel [90] and the latter with Florian Rampp [94].

In the project Distributed Medical Process Support (DMPS) we constructed a peer to peer platform for order-entry and result reporting. The decision to begin with such an approach was to some extent influenced by a former study of my supervisor that identifies support for order-entry and result reporting, amongst others, as a key issue in healthcare processes [22]. I published the DMPS approach in [95].

The applied exchange protocol is deduced from the traditional diagnostic-therapeutic cycle and is adopted for distributed environments, thus, I named it the pandiagnostic-pantherapeutic protocol. The DMPS takes the *counterpart identity availability* into consideration: the counterpart can be known at the time of document shipping (addressed communication) or the counterpart can yet be unknown (unaddressed communication). The latter is the common case in German healthcare because unaddressed referrals only state the necessary medical specialty and the patient freely decides which specific doctor to consult. Thus, in this project we took special efforts to integrate not only an online approach of communication but also an offline transfer, for example by means of flash drives. The DMPS implementation is based on the RESTful paradigm [96]; it applies HTTP as its transfer protocol for addressed online communications. In addition, not only a direct transfer is considered but also a mediated approach in form of a composite of directed communications. The DMPS communication styles are outlined in figure 2.2.

In the whole process, each participating healthcare system is free to delegate diagnostic-therapeutic treatments to other institutions, which are its downstream institutions. The DMPS acts as a system extension. It creates a document that contains the global

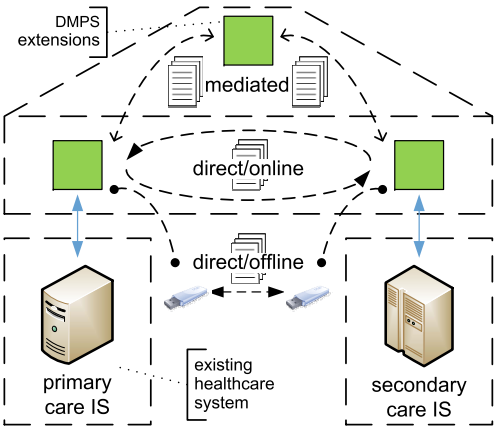

Figure 2.2: The DMPS communication styles

process information, or propagates this process document to the downstream DMPS correspondents. The process status information consists of a shared process ID and the process history with any involved institutions. The process history provides information about the pretreatment or mutual treatment providers. The downstream and upstream relationships between institutions are outlined in figure 2.3.

There had been two prime objectives of the DMPS approach. The first was the abdication of any central server, like joint databases, transaction monitors, and central context managers, as adherence to the strict autonomy of the institutions. The second objective was the application of document-oriented integration with lightweight interfaces instead of service-oriented integration with semantically rich interfaces.

The DMPS approach focused on uni- or bi-directional communication for traditional healthcare supply chains. Yet, from the onset, ProMed did not only want to support traditional supply chains with strict downstream/upstream relationships but also closely cooperating dynamic teams. Such requires mechanisms for team publication. Thus, in parallel to the DMPS efforts, I focused on multi- or omni-directional communication with the project Distributed Electronic Patient File Update System (DEUS).

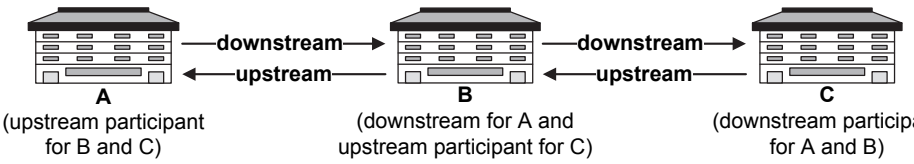

Figure 2.3: The downstream and upstream relationships between institutions

DEUS implements distributed electronic patient files and focuses on trans-sectional, life-long, and patient-centered healthcare documentation; I first published it in [97]. There are two distinguishable features of DEUS: to apply document-orientation as instrument of inter-institutional integration and to allow patients to control information distribution. The latter objective was basically motivated by the underlying intent of "Patient Empowerment", a term that is a general issue in eHealth, for example it had been addressed by the EU in form of a ministerial declaration [98]. A recent study on European citizens and their level of empowerment has been provided by Santana et al. [99].

DEUS applies the document-oriented idea in form of Digital Cards (DCs) that are self-contained units of information. Unfortunately, the preferred term *information card* was patented in 1972 by Paul P. Castrucci, US patent 3702464. The DC metaphor had been inspired by the Higgins project [100] with its *I-Cards* as foundation of an open source identity framework. A DC is authored by an information provider (first part of its ID, e.g., a physician) and it concerns a subject (second part of its ID, e.g., a patient). DEUS puts a *mediated publish-subscribe architecture* into practice; one emphasis lies on its systematic distinction of actors, roles, phases, and responsibilities in the distribution scenario. A sophisticated system modularization had also been a major design goal; the DEUS architecture is described in a technical report [101].

The basic DEUS scenario is outlined in figure 2.4. The patient ("Alice") has recently visited a healthcare professional ("Dr. Higgins"[3]) and the obtained information has to be shared to other involved parties (inter alia "Prof. Bob").

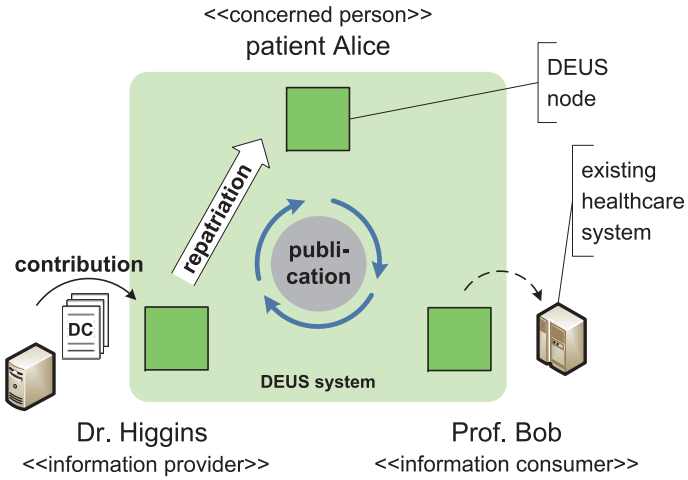

Figure 2.4: The DEUS scenario as mediated publish-subscribe system

3 The exemplary actor "Dr. Higgins", as actual contributor of DCs in the DEUS scenario is a homage to the Higgins project that inspired some of the DEUS concepts.

The local Healthcare Information System (HCIS) of Dr. Higgins, the author of the obtained information, bundles the information into a DC. This DC is electronically signed by its contributor and becomes the subject of information distribution. Subsequently, it is contributed into the node's local DEUS system extension. The exported DC is subsequently transferred to the account of patient Alice who is the person being concerned by the medical information. The patient as sovereign of information distribution decides whether the information is accepted into the pool of DCs that builds his or her *personal patient file*. The process of transferring a DC from the contributing DEUS system to the patient's DEUS system together with the patient's decision about the acceptance of the DC is named *repatriation*. Subsequently, the DC is published to any subscriber DEUS systems, like Prof. Bob. He will consume the information later, for example when Alice is visiting next time.

Each DEUS participant owns a DEUS account. A DEUS node is a healthcare information system with an installed DEUS extension. Platform independence and the avoidance of vendor lock-ins required that the basic architecture was decoupled from any specifically instrumented middleware and components off-the-shelf. A DEUS node can host multiple DEUS accounts and the applied data architecture implements multitenancy. For the mediated publish-subscribe interactions, it is transparent whether an account resides on the same or on another DEUS node.

Another term that describes DEUS in regard to its patient-centred approach is Personal Health Record (PHR), which is an old term that appeared first in 1978 [102]. The term has gained new momentum; Tang et al. provide a survey about the recent PHR conception in [41]. However, solutions that are currently marketed as a PHR are Web-based and centralized; Sittig provides an overview in [103]. Well-known examples are systems that I mentioned during the introduction: Dossia[4], PatientsLikeMe, Microsoft HealthVault[5], and the discontinued Google Health. Whereas the original PHR concept naturally emphasizes a paper perspective (like we do in DEUS with document-oriented integration), the emerging Web-based PHR systems, however, are intrinsically database-oriented with semantically rich data models. Rich data models imply integration efforts in a proprietary way and in a fragmented fashion. PHR systems do not necessarily provide an export; mature products at least provide an import for particular subsets[6] of the Continuity of Care Records (CCR) format.

The IHE started to attend issues of PHR integration by adopting it under the profile XPHR into the Patient Care Coordination (PCC) efforts [105]. Yet, PHR integration currently

4 Dossia is only available to employees of the few signee companies.
5 Microsoft HealthVault is not available in Germany
6 The CCR-subset that had been supported by Google Health is documented in [104].

depends on the range of offered import/export facilities and ultimately remains an open issue. Finally, PHRs risk to become "yet another data silo" or even a "data tomb": When Google closed the Google Health service, the users got a database dump, amongst others in a subset of the CCR format. Such an export was better than expected but it remains insufficient because many other PHR vendors have incompatible data models. A survey on the problems of contemporary PHR implementations is provided by Schwarze et al. [106]. In addition, Schwarze et al. describe their own "akteonline extendable" (AOX) implementation of a PHR. The interesting aspect about AOX is its emphasis on CDA-based document-oriented data integration. With the collection of electronic documents that constitute such a patient file, it is guaranteed that it is possible to export its contents in a valuable format at any time. DEUS shares key concepts with AOX, still, AOX is a centralized web-based platform and DEUS is a distributed pub/sub platform.

A follow-up bachelor thesis, which I initiated during this phase, was a project about evaluating technologies for distributed metadata repositories (cf. [87]). It focussed on Extensible Resource Identifier (XRI) [107] and XRI Data Interchange (XDI) [108] because both technologies are associated with the Higgins project. I had been in personal contact with Markus Sabadello, an XRI/XDI-pundit [109], who extends these base technologies for the purpose of Vendor Relationship Management (VRM) and Personal Data Store (PDS) solutions. VRM provides customers with IT means to bear their share of the relationship burden in contact to vendors and other organizations in terms of data maintenance. Thus, VRM is similar to patient-doctor-relationships in which the patient takes initiative in data management, similar to the DEUS scenario. A PDS is a generalized form of a VRM. It is a semantic database in form of an experimental peer-to-peer data architecture: a PDS should provide means to manage personal data that remains under personal control and means to selectively share personal data with organizations. The PDS infrastructure is based on web servers as peers. The user accesses his or her PDS with a web browser. In contrast, DEUS nodes allow local access to patient files without an online connection. I did not continue the project after the student's completion of his thesis. Still, our experience with XRI/XDI was a successful one. XRI/XDI it is a hybrid solution that combines web-based access and presentation with a distributed pub/sub server infrastructure. In conclusion, VRM and its generalized PDS approach both seem promising and they are concerned with key problems in another domain that are shared by healthcare scenarios.

Finally, at that time, it was the plan to converge both DMPS and DEUS characteristics into a unified platform. In its final form, DEUS still lacked an adequate user interface and is just a transfer platform; several conceptually unsolved issues remained. For example, there is a specific drawback about patient-based mediation: if the patient lacks the abilities to sovereign his or her healthcare information, it would be necessary to delegate this role to a legitimate proxy person or institution, possibly a general practitioner.

Proxy delegation had not been implemented into DEUS. Then, from the physician's point of view, in trans-sectional, lifelong patient files sophisticated facilities for content filtering are necessary in order to extract case-relevant information such that a physician is not overwhelmed with the entirety of the patient's history; DEUS lacks content filtering due to its content-agnostic distribution paradigm. Besides, neither DMPS nor DEUS are perfectly suited for ad hoc processes because both systems had been designed as system extensions and require a minimized but still significant amount of software installation and administration at the participating sites. In addition, the DEUS system did not contain any kind of process semantics, i.e. neither a case identifier had been integrated nor is any assistance provided for managing shared therapy plans.

In conclusion, DEUS has a strong potential as a transfer platform for providing trans-sectional, lifelong, and patient-centred healthcare documentation. Yet, we discontinued DEUS and its distributed patient files in favour for distributed case files based on active documents, in order to support ad hoc processes without prior integration of any systems and in order to provide process semantics for shared therapy plans. Still, DEUS had a strong conceptual influence on *α-Flow*, its final successor.

2.1.3 Turning Point: Revitalization of the eGK Project

The German governmental project "Elektronische Gesundheitskarte" (eGK) was initiated after the German parliament passed a bill, the "GKV-Modernisierungsgesetz", to modernize health insurance cards in 2003. Since 1995, chip cards had been used in German healthcare in form of electronic health insurance cards ("Krankenversicherungskarte"), and since 1999 the forum "Telematik im Gesundheitswesen" had discussed telematics for German healthcare. Thus, the objective of the 2003 modernization plans was a telematics infrastructure for interconnecting healthcare facilities [110] with some new electronic health smartcards as its cryptographic foundation. The 2003 bill defined the first January 2006 as the latest start date for national deployment.

A survey of the eGK project history can best be gained by Prof. Peter Mertens analysis [111]. The *bit4health* consortium (e.g., with IBM, T-Systems, SAP, and Siemens as members) was founded in 2003 to answer for the overarching project. A first concept should be developed by *Protego.net*, its solution was considered as too complex. The *Fraunhofer Institute for Software and Systems Engineering* (ISST) in Berlin was consulted; accompanied by bit4health, the ISST provided a restructured and revised specification.

I will provide a short overview of the resulting eGK overall concept: The primary eGK function is a cryptographic Public Key Infrastructure (PKI) with smartcards and card readers as well as connectors for online communication. On top of this basic communication platform, the *mandatory eGK applications* comprise:

1. health insurance master data
   [eGK: “Versichertenstammdatendienst” (VSDD)]
2. electronic prescriptions
   [eGK: “Verordnungsdatendienst” (VODD)]

In addition, so called *optional applications* had been described in the official documentation [112]. It comprised a requirement analysis for the following applications:

- ⋆ a patient's pharmaceuticals history
  [eGK: “Arzneimitteldokumentationsdienst” (AMDD)]
- ⋆ the patient's emergency health data
  [eGK: “Notfalldatendienst” (NFDD)]
- ⋆ an infrastructure for physicians' result reporting
  [eGK: “Arztbriefdienst” (ABD)]
- ⋆ an infrastructure for physician-provided electronic patient files [eGK: “Elektronischer Patientenaktendienst” (EPAD)]
- ⋆ an infrastructure for patient-provided information to his/her electronic file [eGK: “Patientendatendienst” (PDD)]

In contrast to the mandatory eGK applications, no technical system specification exists for any of these optional applications. In eGK terms, my ProMed system DMPS represents a solution to the ABD; and DEUS represents a combined solution to EPAD and PDD.

Back to the timeline. In 2005, the *gematik mbH* was founded in Berlin to implement and operate the eGK infrastructure and to supersede previous organizations. All direct eGK project efforts concerned only the primary PKI infrastructure as well as the mandatory applications VSDD & VODD. Lab tests and field tests with ten thousand participants were scheduled for 2005. Technical difficulties occurred and non-compliance to security standards was alarming. The social-technical embedding into the doctors' office systems and dispensary systems was criticized. The introduction of the eGK became a political controversy. General project blame was broadly discussed in the media. In 2007, the German physician congregation (“Deutscher Ärztetag”) voted against the health card in its current form. Some believed the eGK as dead.

A turning point were the government elections (“Bundestagswahl”) in 2009, after the coalition of the political parties Union and FDP replaced the former grand coalition. After long controversies, new field tests were initiated with improved cards and card readers. After the eGK project gained new momentum, the promises of cost reductions by electronic prescriptions was challenged, and the NFDD with its emergency health data became discussed to replace VODD in the project's priorities, changing from *optional* to *quasi-mandatory* and vice versa. Then, the extent of NFDD emergency data and its access conditions became yet another unsolved political controversy. Today, there is an

ongoing roll-out of a preliminary version of the eGK ("eGK der 1. Generation"); the only actual functional surplus value to the chip cards since 1995 are a nation-wide life-long patient identifier and a patient's photographic image. Both the PKI and the emergency data (i.e. NFDD) are still work in progress. In 2011, the federal mister of health at that time, Philipp Rösler from the FDP, declared work on electronic prescriptions (i.e. VODD) and electronic patient files (i.e. EPAD) stopped for the time being.

For my ProMed project, it was crucial that in 2009, in the wake of the revitalized eGK project, there appeared pilot projects for ABD and EPAD with governmental support, like the *ProspeGKT*[7] project. We had just finished initial DMPS and DEUS prototypes, and I had just written the first according publications in April 2009, to be published in July and September 2009. After the government change and the coalition agreement from October 2009, we expected governmental projects for ABD, EPAD, and PDD to outrun our own efforts. Thus, Prof. Lenz and I decided to discontinue DMPS and DEUS and to re-focus my ongoing research on the process and workflow aspects, in the hope that the eGK project would ultimately provide a platform of information exchange and therefore solve the issue of technical integration in German healthcare. Thus, our final approach to distributed case files ever since relies on the eGK project to sometime provide a PKI with secure, trusted, and guaranteed delivery for inter-institutional store-and-forward communication.

2.1.4 Third ProMed Phase: Process Support and Distributed Case Files

With my final ProMed project, the *α-Flow* approach, I mean to provide case handling in distributed environments with an emphasis on document-oriented systems integration. *α-Flow* provides a prototypical implementation 1) that enables knowledge-driven ad hoc processes with an initially unknown set of activities and actors in inter-institutional environments and 2) that offers process support in spite of prevalent system heterogeneity without prior system integration. Thus, *α-Flow* implements distributed Document-oriented Process Management (dDPM).

The traditional paper-based interaction paradigm, which uses signed forms for communication, is imitated. In DEUS a communication scenario was implemented that has no counterpart in current working practice, howsoever simple, effective, or desirable its intention is. In *α-Flow*, the basic principle of a patient with some referral and his paper-based case dossier gathered within a ring binder who is walking from doctor to

7 Project information, e.g., available at: `http://www.prosper-netz.de/PROSPER/DE/02_Navi/01_prosper_proGesund/04_elek_karte/ePA.pdf`

doctor is taken as literally as possible. From the data synchronization perspective, this metaphor is extended to exploit the potential of electronic communication. From the workflow perspective, it is extended to enable shared process planning as well as managing data about any case or team members; in healthcare terms, this is shared diagnostic-therapeutic planning and managing the involved physicians. Even if the focus is on healthcare, when I explain dDPM in later sections it will become clear that the case-driven characteristics can also be valid to many other domains, for instance sales & acquisition, IT project management, scientific processes, and the law system.

The *α-Doc* is the primary *α-Flow* item and is my notion of a distributed case file that contains all case related information to be shared among multiple participants. The "α" stems from "active", in analogy to the underlying concept of active documents with active properties. A supplemental abstract about the *α-Doc* conception is provided at the ending of this chapter (sect. 2.3) as a summary.

2.2 Applied Methods

The following section describes principles that have determined the overall system design. Detached disciplines will merge into the overall system solution, like rule-based systems, adaptive-evolutionary system design, data synchronization, version control, and workflow notations (cf. sect. 1.4.1). From these fields additional methods have been applied that are not described in this chapter. These facet methods are later discussed in the context of the individual subsystems of the *α-Flow* prototype.

2.2.1 Degrees of Integration

Integration efforts are caused by semantic heterogeneity. Batini et al. in [113] as well as Heiler in [114] declare that schema integration cannot be automated in general. Experiences from federated database systems demonstrate that there can be situations in which a consistent interpretation of heterogeneous sources is impossible (e.g., [115]). The autonomy of system applications and integrity of data and/or interoperability of functions are conflicting goals.

Lenz in [89] provides two dimensions of integration, integration layers and integration scopes. The layer dimension distinguishes *data integration* and *functional integra-*

tion[8]. The scope dimension distinguishes *technical integration* from *semantic integration*; whereas technical integration is also referred to as syntactic integration. Data integration is further refined by distinguishing it into three levels: the *instance level*, the *type level*, and the *context level*; whereas the context level refers to relationships between data objects. An overview of this classification scheme is illustrated in figure 2.5.

Data integration achieves data compatibility either by common standards or by data transformation. The purpose of technical data integration is to translate between syntactic frameworks. The purpose of semantic data integration is "to create a unique reference for commonly used data and to ensure data consistency" [89]. There are standards on instance level primarily for end-users as a semantic reference at run-time. There are also standards on type level for system implementers at design-time. Data integration will be discussed from the perspective of document-orientation in the following section (→ 2.2.2).

Functional integration achieves interoperability between applications. Syntactic functional integration, for example, translates between different Interface Description Languages (IDLs) of middleware frameworks. Semantic functional integration, for example, has to resolve situations in which systems overlap in their functionality. Functional

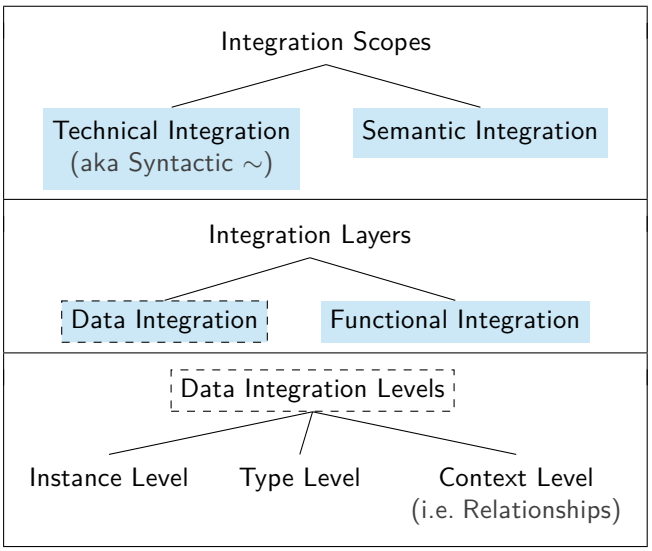

Figure 2.5: Classification scheme for application integration (adapted from Lenz [89])

8 Originally, the integration layer dimension also distinguished the presentation integration, or desktop integration. Yet, it was decided that it is an integration complex that actually concerns both functional and data aspects. Thus, the resulting classification matrix did not list desktop integration separately.

integration will also be discussed from the perspective of document-orientation in one of the following sections (→ 2.2.3).

In addition to the two dimensions, more aspects about integration can be identified, which are illustrated in figure 2.6. For example, Lenz discusses desktop integration in [89] and process integration in [10]. Both may be subsumed under the term "*integration complexes*".

Desktop integration concerns the presentation layer of a system architecture and involves both functional integration and data integration [89]. The purpose of basic desktop integration, for example, is to unify the layout of the applications or their concepts of user interaction. A further degree of desktop integration may unify application access by means of a "single sign-on" mechanism. Another form of desktop integration is desktop synchronization, which is a concept to have multiple windows of disparate applications that share a common operational context. Desktop synchronization provides a protocol such that if the user changes the operational context in one application the interlinked applications are synchronized accordingly. An exemplary healthcare standard for desktop synchronization, i.e. context management at the user-interface level, is the HL7 Clinical Context Object Workgroup (CCOW) specification (e.g., [116]).

Process integration is related to user acceptance and system evolution. Methods to foster process integration are participatory design and continuous evolution. These concepts are further discussed in one of the following sections (→ 2.2.4).

The final aspect about integration concerns different integrator roles (cf. fig. 2.6). In [117], Lenz distinguishes the application designer and end-user from an architect for general system functionalities. For example, general system functionality stores newly defined

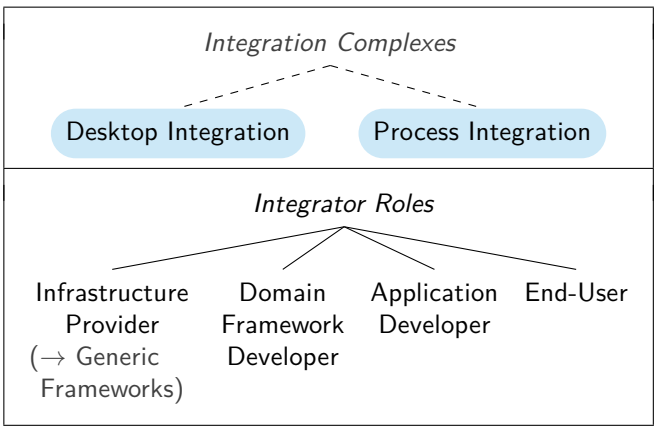

Figure 2.6: Additional aspects of application integration (adapted from Lenz [10, 89])

data units within a joint content repository, or provides means for workflow specification (cf. [117]). The purpose of an according system architecture is to reduce system complexity and to provide reusable system services for higher layers. In [89], the roles are refined in four different responsibilities on four layers of system design by further distinguishing application-independent domain functionality from domain-independent generic functionality. These four layers are illustrated in figure 2.7.

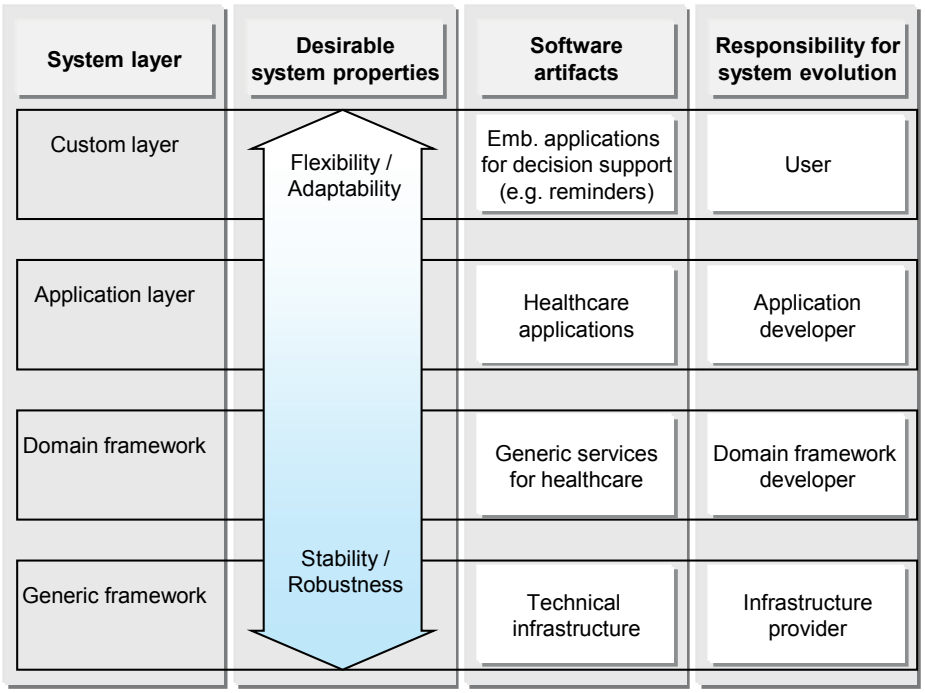

Figure 2.7: Multi-level software engineering in healthcare (adapted from Lenz [10, 89])

The approach to multi-level software engineering in healthcare is expected to permit different degrees of integration. The purpose is to bring software development as close to the end-user as possible. Thus, multi-level software engineering can also be a method to achieve a participatory design and to foster continuous evolution (→ 2.2.4). Finally, a notable implication of taking the user into responsibility for the top system layer concerns data integration: flexible support can mean that semantic data integration is not ensured on the application layer but is deferred to the end-users at run-time. For inter-institutional environments, the same argument is maintained by the pay-as-you-go approaches and the dataspace principle (cf. sect. 1.2.1).

2.2.2 Data Integration: Records or Documents

Exchange of information among healthcare institutions requires data compatibility in various ways. Semantic data integration for medical processes requires standards for medical terminology. Such standards have to deal with volatile medical concepts [118]. Over the intervening years numerous standards for medical ontologies have been created on type level and on instance level. The state-of-the-art section 3.1.1 will discuss available data specifications on the different levels in more detail.

Electronic Medical Record (EMR) and Electronical Health Record (EHR) (e.g., [37]) subsume any system that provide institutional or regionally federated access to a patient's longitudinal collection of health data, mostly including a whole range of data in comprehensive form which is not directly suitable for sharing it inter-institutionally. They typically contain data that can be extracted on demand. Yet, it is unclear how these systems scale and how direct communication between institutions can be effectively supported in large-scale scenarios. EMRs and EHRs fit in the notion of our approach in that they are the natural technological source of medical information in the local institutions. They already support printing for paper-based working practice, thus, using for example a freely available Portable Document Format (PDF) printer driver it is possible to extract the inter-institutionally relevant paper-based information into an electronic equivalent.

A conceptual change from messages and records to electronic documents is provided by the HL7 v3 Clinical Document Architecture (CDA). In section 2.2.6, deferred system design will be explained, which is supported by CDA. Any new standards should respect the ones already in practice for backwards-compatibility and to achieve and to maximize acceptance. In conclusion, the increasing importance of HL7 v3 CDA motivates to accordingly exchange documents within a distributed case file. Thus, EMRs and EHRs are preferably required to support data export based on specifications like CDA for the reason of the highest possible degree in semantic data integration. However, even if CDA is the preferred document format for the exchange between the healthcare participants, the proposed case handling infrastructure must not depend on CDA because of the uncertain system qualifications in large-scale inter-institutional environments. Methodologically, a distributed case handling system must be agnostic to syntactic and semantic content standards.

2.2.3 System Integration: Interfaces or Documents

System integration in healthcare is traditionally based on interface-orientation. Three-tier network-based architectures with remote procedure calls are yet the dominant archi-

tectural style for information systems. A modularization is often based on a component-oriented system design. Szyperski gives an overview on component-orientation in [119]. The Unified Modeling Language (UML) provides the component diagram as a special diagram type for component-oriented system design with an emphasis on exported and imported interfaces. Today, several component frameworks support the implementation of a component-oriented system design. The most prominent ones are those for the Java programming environment and for the Microsoft environment. For the Java environment there is, for example, the *Enterprise JavaBeans (EJBs)* component model of the Java Platform, Enterprise Edition (JEE) framework[9], the *Spring Bean* component model of the Spring framework[10], and the Open Services Gateway Initiative (OSGi) *Bundle* component model of the OSGi framework[11]. For the Microsoft environment there are, for example, the Component Object Model (COM) with its extensions COM+ and DCOM as well as the System.ComponentModel namespace in Microsoft .NET. For distributed environments and inter-institutional scenarios, an infrastructure for remote invocations is mandatory. Today, the most prominent type of remote invocations are Web Services. They are based on Simple Object Access Protocol (SOAP), as a language-independent protocol, in combination with Web Services Description Language (WSDL), as a language-independent interface description language. Web Services can optionally be augmented by various available WS-* extensions. Erl provides an overview of Web Service technologies and WS-* extensions in [120]. The first-order element of component-orientation and remote invocation is the explicit notion of a system interface.

The *interface-oriented integration* focuses on available functionality, and the integration method affects semantically rich interfaces. An invocation uses parameters to detail its synchronous service request to a target system. In interface-oriented integration the information being passed is not necessarily viable on its own but often in the context of the service request only. Typically, the target system must interpret messages in order to assimilate their contents. This way of integrating systems requires a high a priori effort for semantic data integration. The same characteristics and integration challenges apply to record-oriented EMRs or EHRs in healthcare.

Even if a service is triggered event-oriented using asynchronous messaging, as it is done in HL7 v2-based systems, such parameters or messages essentially represent transient fine-grained information that is assimilated by the targeted system. The three main problems in information integration projects, including healthcare systems, are insufficient synchronization of redundant data, problems with data consistency, and functional overlapping [121]. Hence, interface-oriented and message-oriented integration between

9 http://docs.oracle.com/javaee/
10 http://www.springsource.org/spring-core
11 http://www.osgi.org/Specifications/

distinct institutions is complex and custom-designed. The state-of-the-art for related system integration in healthcare environments will be discussed in section 3.1.2 in more detail.

In contrast, documents are coarse-grained, self-contained, and viable. 'Viability' for computer-related artefacts is the ability to independently exist, indicated by an artefact 'life-cycle' that is represented by intrinsic context information that is independent of the local system environment. Changes are not propagated by update information, but by creating an updated document that replaces its predecessor.

The *document-oriented integration* focuses on available information, not on available functionality, and the integration method affects the semantic scalability of document models, using standardized and minimal interfaces for hand-over. Self-contained documents are not updated in place like tuples in a database; instead, document versioning and variant management solutions are sufficient. Likewise are traditional data consistency checks confined to the scope of the document, inconsistencies between documents represent logical errors or divergence in opinion on such semantically high level that a conflict can only be detected or solved by specialized decision support systems or humans. A comparison between both integration styles is outlined in figure 2.8. The term "document" will further be reflected upon in the context of active documents in section 2.2.12.

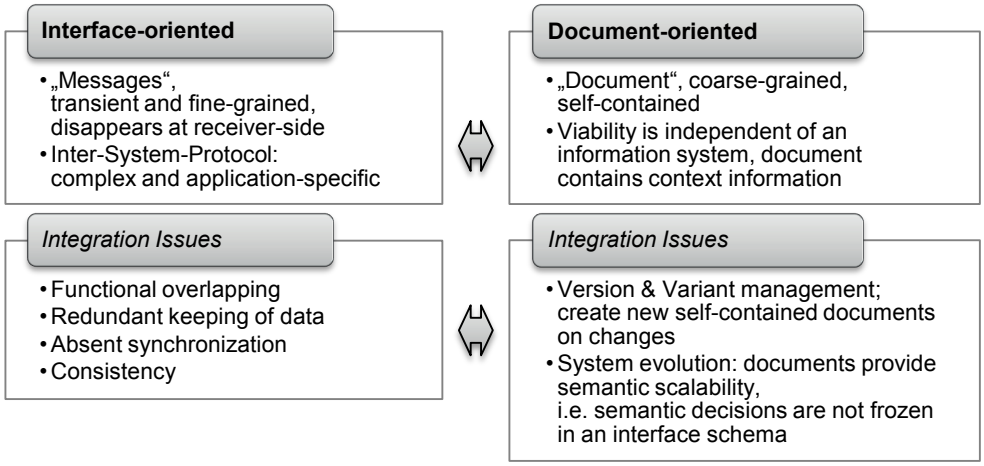

Figure 2.8: Integration styles: interface-oriented versus document-oriented

2.2.4 Process Integration and Software Evolution

The term "process integration" is used in different contexts. It is sometimes used in the context of Business-to-Business (B2B) scenarios (e.g., [122]) as an aspect of Enterprise

Application Integration (EAI) (e.g., [123]) in the form of "process-oriented (system) integration" (e.g., [124]). In a similar context, the term "business process integration" is used as an equivalent to the implementation of Supply Chain Management (SCM) systems (e.g., [125, 126]). The related term "business/IT alignment" [127] implies that over-arching concerns like contractual, legal, regulatory, and strategic aspects extend the mere aspects of system integration. However, the dividing line between B2B process integration and mere system service interoperability and/or data integration is not always sharp.

It is not enough to consider only technical aspects of integration, nor is it enough to consider contractual, regulatory, or strategic business aspects. In order to gain benefits from information technology, the socio-technical aspects are imperative. It is a well-known phenomenon that the most sophisticated system architecture still provides no guarantee that the users actually use the software tools. User acceptance must be a socio-technical objective of integration. This can be considered as user/IT alignment (U/I alignment) as a complement to business/IT alignment.

Accordingly, Lenz defines "process integration" in [10] as "all techniques that help to seamlessly embed information technology into routine work processes". Insofar as it should be necessary to contrast this concept from "process-oriented (system) integration" it may also be referred to as "acceptance-oriented process integration".

Process integration can be improved, for example, by involving the end-user more closely and rapidly in the software development process. Methodical support for obtaining necessary feedback is provided, for example, by approaches like Rapid Application Development (RAD)[12] and other iterative or agile software development methodologies. However, concluding a software development project with a tailor-made application that is based on a participatory system design is only half the battle won. In [10] it is pointed out that necessary adaptations of an application to the actual needs of end-users is not a one-time effort. Process integration cannot be finished with a single terminable software project but requires continuous evolution. A theoretical foundation for the inevitable need for software evolution is provided by Lehman and Belady [129–131].

Lehmann introduced his SPE-classification of software in [129]. He distinguishes S-type ("structured"), P-type ("problem-solving"), and E-type ("embedded") software. *S-type software* solves formally specified problems. Lehmann defines it such that "the only criterion of success in its creation is equivalence to its specification" and S-type software

12 In 1991, James Martin consolidated various methodologies for rapid prototyping of software applications in his well-known book [128]. One key aspect to RAD is the purpose to "rapidly" provide the end-users, i.e. as early as possible in the project, with a Graphical User Interface (GUI) for feedback purposes.

is "always *provably* correct" [130]. The definition of *P-type software* has changed from 1980 [129] to 2006 [132]. The most recent definition of the P-type software is that it lacks a formal specification, instead, a successive approximation is used to produce a working solution. The conception of the problem changes over time, which requires the implementation to be adapted to its changing specification. Both the problem understanding and the system specification may be subject to evolution. Exactly for P-type software projects, iterative and agile methods are recommended. The crucial software type is the last one: *E-type software* by definition "becomes a part of the world it models" [129]. Thus, any analysis of the application "involves extrapolation and prediction of the consequences of system introduction [..]", which "must inevitably involve opinion and judgement" [129]. Furthermore, once the software is completed and being used "questions of correctness, appropriateness, and satisfaction arise [..] and inevitably lead to additional pressure for change" [129]. In conclusion, E-type software implicates an intrinsic feedback loop between changes in the environment and changes to the software, both affecting each other, and software evolution is inevitable. The evolutionary behaviour of large E-type software systems has been studied further by Lehman (e.g., [133]), amongst others it has resulted in the theory of *eight laws of software evolution* in [134].

Both participatory design and continuous evolution are necessary to achieve process integration. Healthcare information systems are E-type software (cf. [10]). A sustainable U/I alignment in an E-type system requires an architectural approach that is flexible enough to support demand-driven system evolution.

2.2.5 Process Support

The discussion about process integration has already implied that processes[13] have something to do with work. From the perspective of *division of labour* [135], the term "process" can be defined as an "organizational form that encapsulates the interdependence of tasks, roles, people, departments, and functions" [136]. Further sub-classifications can be made, for example, Medina-Mora et al. distinguish material process, information processes, and business processes [137]. From the perspective of information technology, the common terminology of Workflow Management Systems (WfMSs) distinguishes three

13 The term "process", in its broadest sense, is used in different contexts like chemistry, mathematics, thermodynamics, biology, or social psychology; in each case with quite different semantics. In the context of computer science, the term is either used in the context of operating systems (program instances), software engineering (development life-cycle), or in the context of workflow management (division of labour).

degrees of process conception: process description, workflow specification, and workflow automation (cf. [138]).

The *process description* is the result of a modelling phase in which informal or semi-formal languages with general and easy-to-understand notations are used. Commonly, this is called Business Process Modelling (BPM). The main purpose of BPM is to enable domain experts to validate, optimize, and re-engineer the processes (cf. [139, 140]).

The *workflow specification* is the result of workflow modelling, which uses a workflow language to formally capture the process. It is not unusual to use different languages for BPM and workflow modelling (cf. [141]), however, such requires to translate the constructs of a BPM language into constructs of a workflow language. It would be deceptive to assume that all BPM concepts are retained as workflow concepts. In fact, a BPM process description usually contains over-arching concerns that are lost by the transformation. At the same time, in order to gain a workflow specification, the semi-formal description must be enhanced with information that is necessary for the controlled enactment of the workflow by a machine. There are different types of workflow languages. For example, Weske and Vossen in [141] distinguish between graph-based languages, net-based languages, and workflow programming languages. In the state-of-the-art chapter (sect. 3.2) an overview of workflow languages is given.

Workflow automation is also called workflow implementation (e.g., [138]), workflow execution (e.g., [141]), or orchestration (e.g., [142]). Georgakopoulos et al. [138] point out that workflow implementation does not necessarily involve an WfMS engine but can be achieved by software engineers that implement the workflow specification tailor-made in form of a customary software application. Yet, the term "workflow execution" implicates that an WfMS engine is used. Accordingly, additional automation-related modelling efforts are necessary to refine the (abstract) workflow specification into an "application-aligned workflow specification" that is computationally complete. Application-alignment means to map individual workflow elements with software applications, database systems, and technical infrastructure. Orchestration is a variant of workflow execution. The term is particularly popular in the context of web services, i.e. web service technology is used for syntactic integration.

The Workflow Management Coalition (WfMC) provides a terminology & glossary document [143] as a reference for activity-oriented workflow approaches. Some basic terms and concepts are illustrated in the appendix (sect. A.3). The WfMC does not consider processes in the broadest sense, instead, its view is particularly restricted to "*business processes*" [143, p. 10]. These are defined as "A set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships." In the same context, the WfMC regards a workflow solely as the part of a business process

that can be executed automatically. The literal WfMC definition of "*workflow*" is "The automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules" [143, p. 8].

This is a somewhat restrictive perspective in the overall context of process support. Rusinkiewicz and Sheth in [144] provide a more general definition:

> "*Workflows* are activities involving the coordinated execution of multiple tasks performed by different processing entities. A *task* defines some work to be done and can be specified in a number of ways, including a textual description in a file or an email, a form, a message, or a computer program. A *processing entity* that performs the tasks may be a person or a software system."

Furthermore, McCready distinguishes three types of workflows in [145]: ad hoc workflows, administrative workflows, and production workflows. In additon, Georgakopoulos et al. provide a classification of workflows in [138] and use system-oriented workflows and human-oriented workflows as antipodes.

All things considered, the dividing line between process support in the broadest sense and workflow management in the narrowest sense is not always sharp. The dDPM approach addresses human-oriented workflows. Process support is provided by managing a shared work-list and by routing and/or synchronizing document artefacts. The process support of dDPM will comprise the following basic aspects:

- to capture *process participants* (Who?)
  at different *institutions* (Where?)
- to capture *work-items* as process steps (What?) in a document-oriented style
- to capture the process plan in form of a prioritized *work-list* (approx. When?)
- to support consensus finding about the process plan by synchronizing the work-list (in a document-oriented style) between different sites
- to share the work-item results (in a document-oriented style)
- to capture process-related status for each work-item
- to manage a process history by versioning the work-list, -items, and results
- to have no intention of workflow automation
- to have no, or not necessarily, knowledge of the content being processed
- to supplement process participation on demand

In conclusion, the work-list is a basic articulation of the workflow in terms of "What?", "When?", "Where?", and "Who?". Georgakopoulos et al. explain in [138] that most workflow languages consist of elements similar to "workflows", "tasks", "manipulated

objects", "roles", and "agents"[14]. The *α-Flow* implementation of dDPM will provide a basic workflow language with according concepts in a document-oriented style.

2.2.6 Deferred System Design and Semantic Scalability

The Deferred System Design (DSD) is a principle for evolutionary systems (cf. [147] and [71, sect. 4.5.2]). It requires semantic decisions not to be frozen in an interface schema because they are hard to revise. Instead, certain schema decisions are deferred from design-time to deploy- or run-time. Applying a document-oriented approach improves the adaptability of an information system because it allows for DSD and for semantic scalability [10].

HL7 v3 CDA provides semantic scalability for healthcare documents, both because this has been an inherent feature of the underlying Reference Information Model (RIM) and because CDA is particularly structured in three levels of semantic abstraction: In figure 2.9 a basic outline of the three CDA levels is provided as illustration. CDA level 1 is the unconstrained CDA specification. CDA level 2 applies section-level templates. CDA level 3 applies entry-level templates. For example, CDA level 1 simply ensures the ability to display a document like a PDF file.

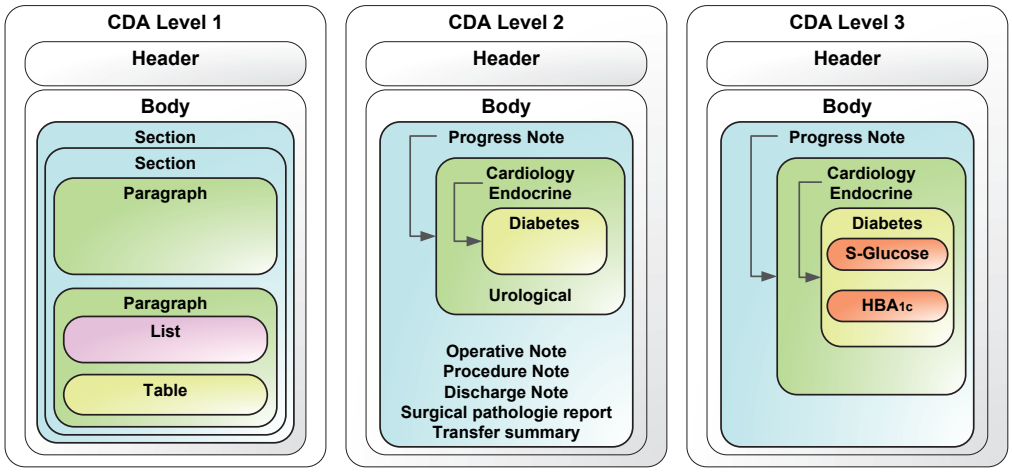

Figure 2.9: HL7 v3 CDA structure outline for levels 1, 2, and 3 (adapted from Sippel [11] and Alschuler [148])

14 Agents is meant as a mere linguistic abstraction of humans or information systems that fill roles and perform tasks in the context of a workflow. No association with multi-agent systems [146] is implied.

DSD in this case means that any CDA document can be accepted without immediate support for processing. Advanced semantic processing support of CDA level 2 or 3 can be added to the system, seamlessly enhancing the information value of already stored CDA documents. Thus, HL7 v3 CDA is favoured as the document type for the artefacts in a distributed healthcare case file, not only because of its document-orientation but also for its specific capability for DSD and semantic scalability.

2.2.7 Prototype-Based Programming

Application systems are constructed by different kinds of high-level programming languages. Amongst others, the capacity for DSD depends on the typing concept of the language that is used for software construction. Popular programming languages are object-oriented languages. Concerning the typing concept, there are two variations of the object-oriented paradigm: the class-based approach and the prototype-based approach. The capability for software evolution depends on that characteristic.

In class-based languages, objects are instantiated from a class with a `new` operator. Inheritance relationships are specified between classes. Classes are the first-order elements of system specification and objects are only second-order elements as run-time derivations from classes. Neither the data schema of an object nor its method behaviour can be adapted, instead changes must be made to the classes.

The programming language that introduced the term "object-oriented programming" was Smalltalk. It belongs to the class-based subtype, as do C++ and Java. Only few class-based languages allow classes to be altered at run-time, Smalltalk is actually one of these few. The first representative of the prototype-based paradigm is the Self language by Unger and Smith for Sun Microsystems. Today, the most prominent prototype-based language is JavaScript[15] by Brendan Eich.

Dony et al. provide an overview on prototype-based object-oriented programming in [151]. Prototype-based languages are classless and apply objects as first-order elements, directly. Objects are either created *ex nihilo* ("from nothing") or by cloning from an existing object. Ex nihilo creation concepts are rather rare, instead, a root object is commonly provided by the language natively, e.g., called `Object`. Thus, other objects can be cloned from the root object, initially. Structural and behavioural changes and extensions to objects are possible, any time. Any custom-made object can become a prototype object

15 JavaScript started out as Mocha language by Eich for the Netscape Communications Corporation in May 1995, then it became LiveScript, and then it became JavaScript when Netscape and Sun Microsystems did a license agreement in December 1995 (cf. [149]). International standardization of the JavaScript language is provided by the ECMA-262 specification [150].

just by cloning another object from it. The clone commonly remembers its prototype as its clonebase. Class-based inheritance concepts are substituted by delegations to its clonebase. The clone can overwrite any attributes or methods' implementation and the clone can be extended with new attributes or methods. Prototype-based programming inherently supports run-time adaptiveness. If the changed object is a clonebase to other objects, its changes can be propagated to its clones at run-time.

In conclusion, the concepts from prototype-based programming languages are better suited for deferred system design than class-based concepts. Unfortunately, there is no JavaScript run-time environment for implementing native desktop applications. Run-time environments for other prototype-based languages are not widely spread. Still, it is possible to adopt prototype-based concepts with a class-based programming language to some extent with programming patterns (cf. [152]). However, the necessary cloning and delegation facilities must be implemented by oneself as well as necessary facilities or substitutes for run-time adoption, in the case that the class-based language supports it not natively. Finally, implementing prototype-based concepts in a class-based language is a challenge but improves the potential for system evolution.

2.2.8 Loose Coupling

Coupling[16] is an old term in computer science, originally introduced by Meyers and Constantine [153], with coupling and cohesion as antipodes of software metrics. In this context of software metrics, one does not speak "loose" but "low" as a terminus technicus: low coupling versus high cohesion (cf. [154]). Coupling concerns the dependencies between software modules and the mechanisms by which the control flow of software routines is managed. Thus, the term is sometimes refined as *code coupling* or *module coupling*. For the same conceptual level, the Institute of Electrical and Electronics Engineers (IEEE) later provided a similar but different classification in its standard glossary [155].

Another form of coupling in computer science appeared within the hardware domain of computer architecture. The computer architecture scope is rather unrelated to the software design scope; however, this scope should not be omitted in a discussion about loose coupling because its community has an equally early claim to the term. This form of coupling concerns the type of shared-memory abstraction in multi-processor architectures (e.g., [156]). Tightly-coupled multiprocessing implies several Central Processing Units

16 Computer scientists should be aware that "coupling" is, unfortunately, associated in non-technical context with biology and sexual intercourse. In German, fortunately, this problem does not exist with the translated term "Kopplung".

(CPUs) connected at the bus level. The "multi-cores" represent the tightest-coupled multiprocessing, with multiple CPUs integrated on a single chip. Loosely-coupled multiprocessing refers to different kinds of cluster computing.

Today loose coupling is discussed in the context of distributed systems. It is not a formally defined concept; rather it is seen as helping to reduce the overall complexity of an information system architecture. Krafzig et al. [157, p. 47] originally provided an exemplary table with system properties of loose coupling. On purpose, it mixes different levels of abstraction and it ultimately spans dimensions from the physical to the logical, from data to function, from syntax to semantics, and from the technical to the organizational.

Josuttis modified and extended the table in [158, p. 36]. A compact description of both property sets is provided by Stiehl in [159, pp. 88-95]. Table 2.1 provides a joint overview. The original properties are unmarked. As a refinement and extension of the "communication style" property, I want to add "computation timeliness" as well as "node availability"; both are marked with a triangle '△'. The properties by Josuttis are marked by a diamond '⋄'. Josuttis replaced Krafzig's original "physical coupling" with characteristics from a transfer protocol perspective; I decided to untangle and retain both aspects, thus, introducing "transfer protocol" for Josuttis; marked with a braced diamond '(⋄)'. Then, "concurrency control" is actually discussed by Josuttis but it was not included in his table; it is re-included and marked with another braced diamond '(⋄)'. Finally, I extend the table by appending three properties of loose coupling for evolutionary systems described by Lenz in [10] as well as the deferred system design property; these are marked with a star '⋆'.

An architectural style that implies minor requirements to be supported by participating systems is the Representational State Transfer (REST) architectural style. It is the generalization of the architecture of the web, proposed by Fielding [96], the co-author of the original Hypertext Transfer Protocol (HTTP) with Berners-Lee. REST provides a paradigm for decentralizing applications in which applications are decomposed into resources with various representations and links between them. The RESTful approach does not require an additional marshalling layer as do interface-oriented remote invocation approaches like SOAP. Instead, REST emphasizes the *explicit* modelling of the representation; in the interface-oriented approach the representation is often generated implicitly by vendor-specific tools, which can lead to incompatibilities in the encoding (e.g., [160]).

We applied a RESTful system design to our DMPS approach during an early ProMed phase (cf. sect. 2.1.2). The benefits of a REST architecture are its minimal requirements and its coarse-grained resource/representation approach, which allows for a document-oriented architecture and which compels loose-coupling in terms of data model, type system,

	Tight Coupling	Loose Coupling
Physical Coupling	Direct physical link	Physical intermediary
Transfer Protocol (◇)	Point-to-Point	Via mediator
Communication Style	Synchronous	Asynchronous
Computation Timeliness △	Online processing	Batch processing
Node Availability △	Continuously running nodes	Sporadically running nodes
Data Model ◇	Complex common types	Simple common types only
Type System	Strong type system (e.g., interface semantics)	Weak type system (e.g., payload semantics)
Interaction Patterns	OO-style navigating of complex object trees	Via data-centric, self-contained messages
Platform Dependencies	Strong OS and programming language dependencies	OS- and programming language independent
Discovery and Binding	Statically bound	Dynamically bound
Control of Process Logic	Central control of process logic	Decentralized control of process logic
Concurrency Control (◇)	Pessimistic, i.e. blocking	Optimistic, i.e. non-blocking
Decentralized Transactional Behaviour ◇	Distributed 2PC (Two-Phase-Commit) and its relatives	Compensating Transactions
Deployment ◇	Simultaneous	At different times
Versioning ◇	Explicit upgrades	Implicit upgrades
Application Extensibility ⋆	Adding applications requires the modification of the present applications	Add applications without the need to modify other applications
Application Privation ⋆	Removing applications requires the modification of the present applications	Remove applications without the need to modify other applications
Up-Front Integration ⋆	Any data transfer requires a priori integration efforts	Data transfer is possible without previously interconnecting some systems explicitly
Deferred System Design ⋆	Design-time dependencies	Run-time dependencies

Table 2.1: Different kinds of loose coupling (adapted from Krafzig et al. [157], Josuttis [158], Stiehl [159], and Lenz [10])

interaction patterns, and platform dependencies. Yet, the basic REST approach does not anticipate network nodes that may be offline for a significant period. Offline nodes require an underlying store-and-forward network with *persistent queues* and *guaranteed delivery* [161, p. 122].

In conclusion, it is beneficial for any distributed system when coupling is reduced because maintainability increases and system evolution is facilitated. However, for inter-institutional system environments it becomes imperative because the applications at the distributed sites cannot be changed by a single authority. Especially, the traits *deferred system design* and *up-front integration* become a major challenge. For large-scale open-world scenarios, as in healthcare, a data exchange model needs to allow for data transfers without previously interconnecting two systems explicitly.

2.2.9 Request for Transmission: Sender-Push or Receiver-Pull

The debate around sender-push versus receiver-pull is an old one. In the advent of computer science, *polling* as a form of receiver-pull with busy-waiting was predominate for input/output handling by operating systems [162, p. 62]. It was superseded in the 60s by *interrupts*[17] as a form of sender-push in the context of I/O operations. During the advent of the Internet, sender-push technologies like Simple Mail Transfer Protocol (SMTP) were still favoured. With the impact of the World-Wide Web (WWW), receiver-pull architectures based on HTTP became popular, later generalized as RESTful architectures. Recently, the debate has been resurrected, in the context of the WWW, by the advent of Extensible Messaging and Presence Protocol (XMPP) and Asynchronous JavaScript and XML (AJAX), both being sender-push web technologies. In the context of Content Management Systems (CMSs) the topic has also been debated, for example by Cummings [163] who contrasts Pull CMS with Push CMS and who favours push technology. In conclusion, Duan provides a survey in [164] on the various and subtle variations of receiver-pull and sender-push.

In medical care, the availability of information at the right time and at the right location (the "point-of-care") is crucial [24]. A pull-based approach would require a healthcare information system to query a patient account ad hoc when information is needed. Since a local replicate of the electronic patient file is absent, there are several disadvantages: The absence of a local copy requires the continuous availability of the patient file host for information provision, which is not necessarily guaranteed. Notably, in the context of the WWW architecture, the pull-based model has sometimes been referenced as *lightweight*

17 One of the earliest interrupt-enabled computer systems was the Electrologica X1, which was subject to Edsger Dijkstras PhD thesis.

access to information without prior relationship establishment between server and client; yet, this benefit is only available for general information that is advertised freely. If it is necessary to establish trust relationships in a multi-peer environment, both the pull-based and push-based approaches have the same difficulties. Even with high internet bandwidths and fast HTTP experience, the push-based approach still has the benefit of reduced response times of the local system because a remote call is avoided, which elevates end-user acceptance.

Trying to set-up administrable access restrictions for decentralized pull-based query facilities is prone to security flaws and data leaks. Even mature standards for distributed access control, like eXtensible Access Control Markup Language (XACML), suffer from inherent semantic complexity [165]. Paper-based working practice is based on sender-push postal delivery. It is easy to control receiver sets at message dispatch time. In conclusion, sender-push technology is more effective and efficient than pull-based solutions in large-scale environments.

2.2.10 Separation of Concerns: Content and Decision Support versus Coordination

Another form of loose coupling is necessary for distributed case handling in healthcare on a high degree of conceptual level: The decoupling of content work and coordination work from decision support.

The basic dDPM assumption for inter-institutional processes will be that human or computer supported decisions may always be represented as a demand for further information. In healthcare, this is well known as the diagnostic-therapeutic cycle (cf. sect. 1.1.3). A treatment episode ends when no further information is required for a particular goal.

Local system environments are dominated by social-technical considerations that require tight integration of data-, knowledge- and process support, as it is argued by the careflow solutions [166]. Such mingling of decision support and workflow causes misunderstandings in inter-institutional scenarios. Decision support systems depend heavily on a formal semantic interpretation of medical content from patient records [167]. This requires a canonical content standard for medical information, which can only be consolidated in closed local or regional environments (cf. sect. 1.2.1). The development of a common semantic reference as a basis for decision support is a continuous consensus process, which should be separated from the basic infrastructure that enables cooperation in principle. In order to support heterogeneous systems, we need to decouple collaboration functionality from the local applications. Thus, in large-scale inter-institutional processes, it is necessary to separate decision support from the cooperative workflow

and the necessary coordination work. This can be a challenge. Figure 2.10 outlines the relationship between content, decision support, and coordination.

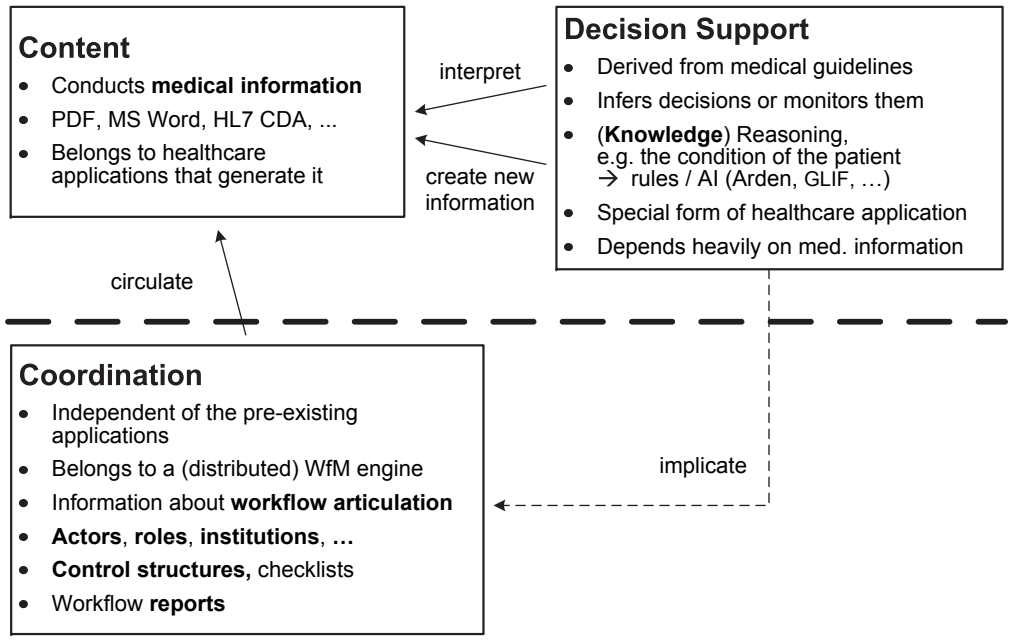

Figure 2.10: Relationship between content, decision support, and coordination

Content documents contain medical information and belong to the healthcare applications that generate them. Ideally, they are well-structured Continuity of Care Document (CCD) documents in HL7 v3 CDA format; however they are typically Microsoft Word or Adobe PDF files. The coordination documents are independent of the local application systems and belong to the distributed workflow. They are required to manage information about actors, roles, and institutions, as well as system topology information, and control structures like checklists. Medical decisions can be represented by the creation of a record keeping document artefact and workflow decision can be represented by adding placeholders ("descriptors") for future artefacts into a shared to-do-list.

A taxonomy of Ortner in [168] provides further evidence for applying a fundamental separation of concerns in an overall system design. The taxonomy is independent of healthcare or any domain. Ortner distinguishes basic systems like Database Management System (DBMS), Decision Support System (DSS), or WfMS and applies a "language-critical reconstruction" of generic functions. The reconstruction results in a terminological reference for the different concepts, figure 2.11 provides an overview. The first three columns in the classification table finally describe universal dissimilarities between content, coordination, and decision.

System Type / Characteristic	Database Management Systems	Workflow Management Systems	Decision Support Systems	Communication Management Systems	...	Consideration Management Systems
Prevalent Language Action Typ	utter/ understand	initiate/ control	ask/ answer	agree/ decline		judge/ provide
Level Pair *)	Schema/ Instances	Control/ Execution	Premises/ Conclusions	Transformation/ Interaction		Meta Language Level Object Language Level
Primary Modelling Item	Subject Entities	Occurrences	Reasoning	Understanding		Considerations
Logic as the Basis of System Development	Standard Logic (Relational Calculus)	Modal Logic (Imperative Logic)	Syllogism (Question Logic)	Interaction Logic (Dialogical Logic)		Metatheory (Metalogic, etc.)
⋮						
Fields of Application	Production and Administration	Management and Controlling	Analysis and Decision-Making	Translation and Imparting of Knowledge		Construction and Invention

*) Note: The level pair "schema/instances" is orthogonal to the levels "control/execution", "premises/conclusion", "transformation/interaction", and "meta language level/object language level".

Figure 2.11: Fundamentals of basic systems: a language-logically reconstruction of generic functions (adopted from Ortner [168])

2.2.11 Cards that represent Tasks

Agile software methods like Kanban [169], Scrum [170], or Scrumban [171] have emphasized on the paradigm to manage tasks by cards. By applying these methods, a set of cards becomes representative for the work-list of a team.

In Scrum, there are development cycles that are measured in weeks, the *sprints*. Some software features or requirements, the *user stories*, are written on *story cards*. User stories are dissected into tasks that are written on *task cards*. Story cards focus on the user experience, whereas task cards focus on the necessary system changes. During a Sprint, the story cards and task cards change their workflow status, for example from unstarted, to ongoing, to completed. To illustrate each stage of work, groups working in the same space often use paper cards on a pinboard or post-it notes on a whiteboard. The board is segmented into state-related areas and the cards are physically moved from one area to the next. A user story is completed when all its task cards are completed.

Using paper cards, task cards remain of the same colour and only change position at the pinboard; colours are only used to distinguish between story cards and task cards. Today there are software tools for Scrum in which colours are also used to indicate task card status instead of using positional semantics. Figure 2.12 illustrates a Scrum task board with both colour and position as indicators for task card status.

In fact, there are other card types, for example *epics* for coarse-grained and still vague requirements as well as *defects* for software errors. There are also more than the three

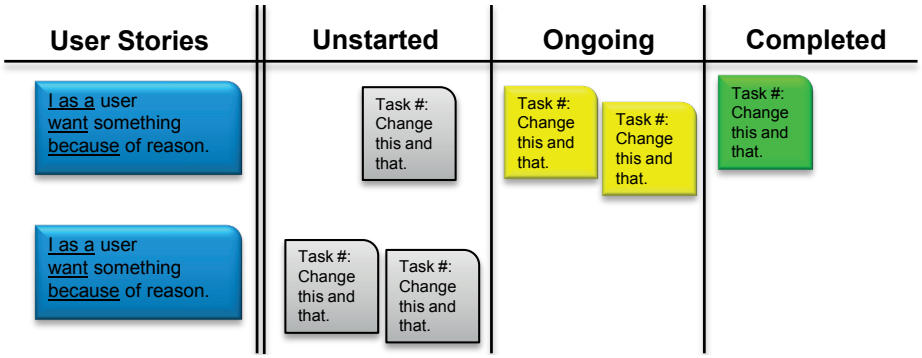

Figure 2.12: An illustration of a Scrum task-board as a card-based work-list

basic task card states, for example there is also deferred that indicates that some kind of external impediment exists that currently prevents task completion. Scrum does not define reference semantics neither on card location nor on card colors. Scrum encourages projects to adopt their own best practice. There are Scrum projects that use a rich taxonomy on task types, like writing code, creating test data, writing unit test code, or designing a user interface wireframe. The reason for such a task-card-taxonomy is not necessarily related to measuring the workflow progress but to improve task-to-person matching. In conclusion, there are many possible process-related attributes of a card.

Not each attribute must necessarily be expressed by colour or location. In fact, Scrum can be applied with only a single card colour and only a few board segments. In that case, most card status attributes will be expressed by adornments. Adornments[18] are textual or graphical markers on the cards, written with flipchart markers on the card itself or using small coloured markers that are pinned on top of task cards. Examples for fixed-and-written adornments are the card type and a numerical score for its costs estimation. Examples for changing-and-pinned adornments are markers for each developer who assigns him- or herself to a task as well as warning-markers for impediments that currently defer task execution.

In conclusion, several Scrum paradigms will become constitutive to the α-*Flow* meta-model. These paradigms are i) to represent tasks as cards, ii) to measure workflow progress by changing card status, and iii) to indicate card status by color, position, or adornments.

There are many more aspects to the agile software development method Scrum than just the cards-represent-tasks metaphor. However, other Scrum mechanisms are not directly

18 The term "adornment" is also used in the UML: a UML adornment adds to the meaning and/or semantics of the element to which it pertains and has a textual and graphical representation; for example the diamond shaped indicators for composition or aggregation are UML adornments.

relevant to knowledge-driven ad hoc processes in general. Still, the following quotations from Schwaber and Sutherland in well-known Scrum publications provide an impression on the Scrum purpose as an agile method:

1. "The sprint phase is an empirical process. Many of the processes in the sprint phase are unidentified or uncontrolled." [172]
2. "Sprints are nonlinear and flexible. Where available, explicit process knowledge is used; otherwise tacit knowledge and trial and error is used." [172]
3. "The project remains open to environmental complexity, including competitive, time, quality, and financial pressures, throughout these phases." [172]
4. "This aspect of self-organization is now understood as a type of Set-Based Concurrent Engineering (SBCE) practiced at Toyota [173]. Developers consider sets of possible solutions and gradually narrow the set of possibilities to converge on a final solution." [174]

Kanban and Scrum have been successfully applied in concurrent engineering and software engineering. Its characteristics match well with the ones we experience from the diagnostic-therapeutic cycle (empirical; tacit knowledge; trial and error), healthcare supply chains (environmental complexity), and medical guidelines (nonlinear; sets of possible solutions).

Finally, the progress documentation by cards matches well with the clinical Problem-Oriented Medical Record (POMR) and its SOAP-formatted progress notes in secondary care (cf. sect. 1.2.7). The Kanban and Scrum methods enhance the POMR conception by articulating the *prospective* therapy plan not only with a narrative "P"-section, i.e. the plan section within a SOAP note, but by creating cards for planned treatment steps. Each prospective step can be represented as an unfilled card that can be considered as a placeholder for a future POMR progress note. Using prospective cards to articulate a plan results in a backlog of cards that makes the future process one degree more structured and explicit than a narrative. The plan can be rearranged by a re-prioritization, addition, or deletion of prospective cards.

2.2.12 Active Documents

The semantics of the term "document" has changed during the centuries. Brügemann-Klein in [175] describes that up to the 19th century the purpose of legal proof was dominant. She quotes a definition from a German lexicon in 1894 – its translation would be "In the broadest sense an artefact that is destined to affirm the truth of a

fact that needs to be proven, especially one that is substantial to a legal position"[19]. Brügemann-Klein further describes that during the 20th century the judicial purpose became subordinate and was replaced by the more general-purpose of an information carrier. Accordingly, Levy in [176] characterizes documents with three constituent properties: 1) *communicative artifacts*, i.e. made by humans to serve communicative functions, 2) *external and public*, i.e. they are separate from their creators and can be made available to others, and 3) *stable or (relatively) permanent*. In the context of this thesis, document always implies digital document or electronic document. Digitalization provides IT systems with basic means to take hold on paper-based document contents. Electronic documents implicate that their document format is designed to be used by IT systems. Advantages of electronic documents over paper-based documents are described by Schamber in [177]. Using electronic documents methodically requires the separation of content, structure, and presentation. The latter has been discussed extensively since the 70s in the context of Standard Generalized Markup Language (SGML) and Extended Markup Language (XML) as well as their stylesheet languages Document Style Semantics and Specification Language (DSSSL) and Cascading Style Sheets (CSS) respectively.

The term "active" stems both from Latin *activus* (i.e. practical; in opposition to contemplativa) and Latin *agere* (i.e. to put in motion, move, lead, drive, tend, conduct). Dourish et al. first described the idea of an *'active' document* in [178][20] in the context of their Placeless documents project at Xerox Palo Alto Research Center (PARC). We will discuss the project again in the state-of-the-art chapter (cf. 3.5) together with the few other existing active document approaches.

Later, the Placeless documents concept of "active properties for dynamic document management system configuration" was patented in the U.S. by Xerox Corporation [179]. It is a middleware likewise to a distributed file system, as an abstraction from existing document- and file-management interfaces. The patent cites several such interfaces like the Document Management Alliance (DMA) [180] interface by the Association for Information and Image Management (AIIM) as well as Network File System (NFS) [181, 182] and Web-based Distributed Authoring and Versioning (WebDAV) [183]. The core of the original approach is about its filing structure and categorization scheme. It contrasts itself to hierarchical tree file systems, instead it applies non-hierarchical document IDs for primary storage—seemingly similar to Tuple Identifiers (TIDs) in database storage.

19 The German original: "Im weiteren Sinne jeder Gegenstand, welcher dazu dient, die Wahrheit einer zu erweisenden Thatsache, besonders einer für ein Rechtsverhältnis erheblichen Thatsache, zu bestätigen".

20 The original paper was submitted in 1999. Other publications in 1999 from the same research group reference it (i.e. "Extending document management systems with user-specific active properties") in its submitted form as the original source of the concept. The primal paper was publicly available not until 2000, later than some of the group's other papers.

The file's hierarchy position is rather managed within static attributes. The patent describes how an NFS, i.e. hierarchical storage, abstraction layer can be provided on top of the middleware: "appropriately formatted directory names are interpreted as queries, which appear to 'contain' the documents returned by the query". The attribute categorization scheme is extendible, allowing different virtual filing structures, and the attributes are explicitly user-specific.

To my own knowledge this characteristic had already become famous since 1996 by Giampaolo and Meurillon [184] in form of the Be File System (BFS) for the Be Operating System (BeOS): BFS provided extended file attributes with indexing and querying characteristics to provide functionality by a file system similar to that of a relational database.

In addition to its relational distributed file system concept, the Xerox PARC approach considers not only informative properties (i.e. static properties) but also *active properties*. Like static properties, active properties can be assigned by users and added to documents; they augment document functionality. In [179], the active properties are described to "exploit knowledge about the external state of the world, documents can, in effect, become 'situationally aware' and be responsive to changes". The examples from the original publication [178] are: a *backup* property that can contain code that causes the document to be written to tape, a *summarize* property that can cause a summary (text or thumbnails) to be generated whenever the document content is changed, and a *logged* property that can cause all document accesses to be recorded.

The definition of the Placeless group relates to active properties. The authors circumscribed the resulting file unit as *'active' document* without formally defining the term. Similar active document conceptualizations had been adopted during 2000 and 2001 by Heinrich and Maurer [185], Chang and Znati [186], and Werle et al. [187]. A survey on these early approaches is available in form of a bachelor thesis by Steffen Idler [188] that I supervised.

Today, the term "active document" is used only for a small set of approaches, which range from the original file system concept to a user interface concept from Microsoft related to its Object Linking and Embedding (OLE). In conclusion, the original active document description is too narrow. In some way this is similar to "object-orientation": the original notion by Alan Kay in [189] was a 6-point list of design principles behind Smalltalk; a generalized definition of object-orientation needed to supersede its inventor[21]. Thus, I want to provide a *generalized definition* of the *"active documents" metaphor*:

21 The common definition of object-orientation is currently based on polymorphism, encapsulation, and inheritance, e.g., [190, sect. 18.1].

① An electronic document
② that allows direct interaction and
③ has active properties.

Comment on 1: the *document* can be atomic in form of a single file or it can be a *molecular document* in form of multiple files, for example (but not limited to) a file set that is bundled according to the *cabinet metaphor* (e.g., Mirosoft CAB files).

Comment on 2: the *direct interaction* implies some form of human-machine interaction, for example (but not limited to) an embedded GUI.

Comment on 3: the *active properties* imply some form of executable logic; the essential aspect is that the logic is part of the document and not part of an external application; by its active properties an active document merges lightweight application characteristics into itself.

From the perspective of the original Placeless document approach, the direct interaction is provided by basic file system operations like copy or move as well as drag-and-drop triggers. The active properties that are triggered by the operations are implemented in form of a scripting language. The programmed action can be of any sort. Various examples for direct interactions and for active properties will be exemplified in section 3.5. The actions of an active document have the ability to autonomously change their life-cycle state based on intrinsic logic.

This kind of *reactivity* is similar to software agents. However, active documents are not (necessarily) software agents just by providing reactivity. Wooldridge and Jennings provide a well-accepted definition of software agents in [191]. The definition requires, besides reactivity and amongst others, *agent autonomy* such that agents "operate *without the direct interaction* of humans or others" and *pro-activeness* such that agents "are able to exhibit *goal-directed behaviour* by taking the initiative". Still, special types of active documents could implement active properties in a way that fulfils the software agent definition but there is currently no such approach.

Finally, *α-Flow* intends to provide ad hoc process support without to require a system installation of a workflow tool prior to process participation. The approach is based on the idea of active documents with the *α-Doc* as its primary item. The symbol "*α-Doc*" essentially can be read[22] as "active document".

22 The "α" symbolically implies the term "active".

2.3 Outline: Case Files via Active Documents

As a synthesis of the sections on project procedure and on applied methods, this section concludes the chapter with an outlook on α-*Doc*s. It is an anticipatory outline about the idea to represent distributed case files in form of active documents. At this point, only a technological outline is provided. Readers with an immediate interested in the benefit for the user may skip to chapter 4 on page 153, which provides a user story and explains the physician's side of view.

Each active document in α-*Flow* carries the workflow context in addition to the domain content and provides autonomous coordination logic in form of a rule-based action library. The purpose is to allow access, viewing, and editing of the original content documents through common editors in the local information system without corrupting the process semantics of the distributed case engine.

The α-*Flow* idea is to form the collective case dossier into a single molecular self-managing file unit, the α-*Doc*. Then it can be handled as passive files like a PDF or a Word file. Still, it contains both the case data and the dDPM enactment engine. α-*Doc*s have the ability to autonomously change their life-cycle state.

In contrast to the original Placeless documents approach, the α-*Doc* does not depend on a special file system middleware. It is a common file on the user's desktop and can be replicated with its active properties like common files across any file system.

From an operative embedding perspective, the α-*Flow* approach minimizes the initial work for establishing an information exchange between different process participants. From a technological perspective, no pre-installed system components are required to interact with an α-*Doc*. Thus, the α-*Doc* is an instantly available tool that needs no administration.

One appeal of the α-*Flow* approach is as follows: If we provide a technical platform for such eccentric case files each human actor becomes participant by handing him or her a copy of the α-*Doc*. In healthcare, this is the same interaction as making them participants by handing over referral vouchers.

For the end user, an α-*Doc* embeds a functional fusion of 1) group-based instant messaging or E-Mail 2) with a shared work-list editor 3) with version control. Trying to provide a Microsoft product metaphor, one could attempt to say that α-*Doc*s are like a self-contained distributed mini-Outlook upgraded by versioning capabilities; or like a self-contained distributed mini-SharePoint. Yet, comparing an α-*Doc* with MS Outlook or similar groupware products, one should keep in mind that these products generally

predefine the data structures that model task entries in form of a fixed schema; in contrast, α-*Flow* integrates a run-time adaptive attribute model for its task entries.

Furthermore, the α-*Doc* embeds a rule engine that guards status changes and executes actions as the kernel of the active document. It currently provides platform rules for versioning and access restriction as well as publication and distributed synchronisation. Workflow benefits are process planning, process history, and participant management as well as template creation with import/export for process structure and process-required roles.

II

Inter-Institutional Processes and Active Documents

3 State of the Art

> “[A workflow system] is built around the concept of waiting.”
>
> (Matt Cumberlidge)

This chapter is split into four sections. First, available standards for semantic integration will be outlined. The following sections provide an overview of activity-oriented workflow approaches and of content-oriented workflow approaches. The last section identifies available approaches for active documents.

3.1 Healthcare Standards for Semantic Integration

Many publications are available on the issue of integration in healthcare. Publications with high reputation are, for example, from Eichelberg et al. [192], Lenz et al. [89], and Lahteenmaki et al. [193]. I will provide an overview of available standards for data integration and for functional integration. The classification of the different types of integration has been discussed in section 2.2.1. Some of the standards and organizations have also been mentioned in section 2.2.2 and 2.2.3. Still, the frequency of names and abbreviations at hand will inevitably be taxing. For both fields, available approaches to document-oriented integration will be highlighted.

3.1.1 Healthcare Standards for Data Integration

At data instance level, standards exist that unremittingly evolve over time. Examples are the International Statistical Classification of Diseases and Related Health Problems (ICD) [194], Systematized Nomenclature of Medicine (SNOMED) [195], and Logical Observation Identifiers Names and Codes (LOINC) [196]. Dealing with inherent volatility of reference terminologies is an unsolved scientific issue. Despite many attempts, there is no stable unique and comprehensive ontology of the medical domain in sight. A *Canonical Data Model*, as it is described as enterprise integration pattern in [161], is not available.

Effective information systems (e.g., [197]) are based on a discreet selection of available specifications, however, data integration is thereby limited to compatible systems.

The Health Level 7 (HL7)[1] v2 is a well-established standard for clinical message specification [198]. It is a standard on type level, and incorporates coding schemes and terminologies on instance level. The HL7 v2 standard allows for the specification of self-defined messages, which has led to incompatibilities.

In parallel to HL7 v2 there are specifications like the Continuity of Care Records (CCR) [199] by the American Society for Testing and Materials (ASTM) for standardization purposes in the United States (U.S.). The initial version of the CCR has its strengths as a lightweight, easily implemented approach, and it was intended primarily for the exchange of health summaries. The integration of CCR with HL7 v2 systems requires a translation via mapping rules (e.g., [200]).

The HL7 v3 standard is based on its core specification, the Reference Information Model (RIM). Thus, HL7 v3 is radically different from the v2 standard: It is based on Extended Markup Language (XML) and allows new types to be derived from a limited number of core classes, enabling RIM-based systems to handle even unknown message-types in a generic way.

HL7 is primarily a standard for hospital environments. There are only a few messaging standards for primary care environments. For German primary care there are a set of text-based "xDT" standards. xDT enfolds four separate specifications; for accounting ("ADT"), treatment data ("BDT"), device data ("GDT"), and laboratory data ("LDT"). The xDT standards were established since 1987, first with ADT. The Standardized Communication of Information Systems in Physician Offices and Hospitals using XML (SCIPHOX) [201] project was founded in 2000 to act as a broker between primary and secondary care requirements, i.e. to bridge between xDT und HL7 standards.

Document-oriented Standards for Data Integration

HL7 v3 supports both messages and documents, the latter are specified by HL7 v3 Clinical Document Architecture (CDA) [202]. CDA provides a framework for XML-structured medical documents. For illustration, an XML listing of a CDA document is provided in the appendix as lst. A.1. Standards like Continuity of Care Document (CCD) and the SCIPHOX specifications are based on CDA. The CCD specification (e.g., [203]) is a U.S.-specific standard that is a constraint on CDA and focuses on document-oriented medical

1 Health Level 7, http://www.hl7.org

content types. CCD is often regarded[2] as the successor of CCR, even if the organizations behind both standards are quite different. For German healthcare, the working group SCIPHOX developed specific document content types based on CDA. For example, referral vouchers and discharge letters [204]. Notable SCIPHOX specifications are the *eArztbrief SCIPHOX CDA R1* and its advancement *eArztbrief VHitG CDA R2* [205]. Recently, the SCIPHOX group has merged into the technical committee of the German HL7 user group and its interoperability forum. Notably, the CCD and SCIPHOX standards do not consider workflow history or workflow coordination information.

3.1.2 Healthcare Standards for Functional Integration

Protocol standards for information exchange in distributed healthcare scenarios mainly focus on hospitals of the secondary care. They are driven by the complexity of a major hospital and its need for inter-sectional information exchange. Available standards, like Digital Imaging and Communications in Medicine (DICOM) [206] or the Cross Enterprise Document Sharing (XDS) [77] standard from Integrating the Healthcare Enterprise (IHE) [207], focus on the information exchange between a Hospital Information System (HIS) that cooperates with ancillary systems like Radiology Information System (RIS), cardiology, and pathology systems, or Laboratory Information Management System (LIMS).

Centralized system functionality is commonly instrumented for institutional or regional integration purposes even with distributed participants: Most of the tailor-made integration efforts are based on a central database system with distributed transaction systems and diverse communication middleware. Even wide-area Regional Healthcare Information Network (RHIN) architectures like HYGEIAnet [208] on the island Crete require a federated database schema [209]. These architectures are tightly-coupled by their complex infrastructure being inadequate for transregional scaling.

In RHINs, several hospitals and ancillary institutions develop a shared set of IT services for information exchange. In Germany, the governmental telematics project "Elektronische Gesundheitskarte" (eGK) has not yet provided a large-scale solution. Thus, no universal exchange protocol and format exists, not even for the interchange of referral vouchers or discharge letters.

Neutral organizations like IHE try to establish interaction standards in format and protocol, little by little, based on standards like DICOM or HL7. Lenz has placed concurring standards into a classifying matrix of integration [89]. In conclusion, a "semantic gap"

2 The history of HL7, CCR, and CCD is described by Ferranti et al. [203].

is revealed. The gap primarily concerns process-related information that rapidly evolves over time. Thus, it is not subject to standardization. The classifying matrix and the semantic gap are outlined in figure 3.1.

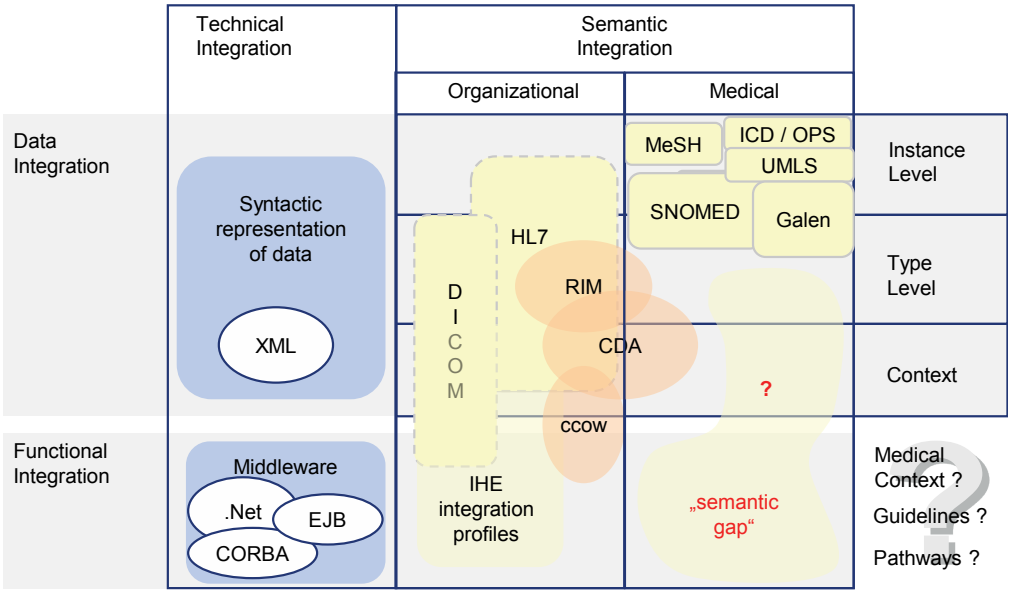

Figure 3.1: Standards for different degrees of integration (adopted from Lenz [89])

Document-oriented Standards for Functional Integration

Solving the information exchange in healthcare in a document-oriented fashion seems to be targeted by IHE XDS [77] specification. For gaining experience with XDS we implemented *XdsRig*, already mentioned in section 2.1.1. XDS allows for distributed document repositories and access delegation. The IHE defines system components and system interactions in IHE specifications as so-called *actors* and *transitions*. The XDS actors and transitions are outlined in figure 3.2.

In order to find documents in such a repository, a single central document registry is specified, reusing Electronic Business using eXtensible Markup Language (ebXML) registry infrastructure to provide a centralized method of indexing documents. The central registry is a global system node that allows queries and that delegates the access to referenced documents to the original document repositories. Such architecture targets complex hospitals with associated ancillary systems and is even applicable to regional integration efforts but fails for wide-area application due to its centralized approach. Other standardized document-oriented infrastructures are not available from authoritative organizations or standardization groups like HL7 or IHE.

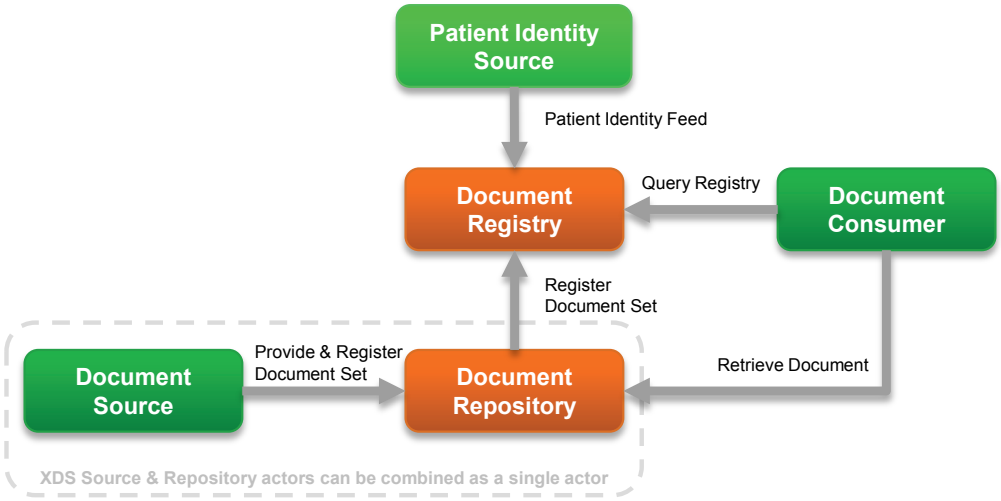

Figure 3.2: IHE XDS actors and transitions (adapted from [77])

3.1.3 Healthcare Standards for Shareable Representations of Clinical Guidelines

Guideline-based decision support systems require a significant effort of integration in healthcare. However, it is important to distinguish between the knowledge representation itself and the application of the knowledge (i.e. rules) to actual patient-related data (i.e. facts). The former implies a shareable representation of guidelines and requires format standards. This is *knowledge integration* and concerns rule codification. It improves the dissemination of a guideline's formalization between institutions and may accordingly reduce collective efforts. However, most of the guideline-based integration efforts are not concerned with knowledge integration. Instead, most integration efforts occur locally for connecting the Decision Support System (DSS), i.e. explicit knowledge, with the Electronic Medical Records (EMRs), i.e. facts, which is customary *data integration*. The comprehensive demand for IT support (of DSS recommendations being dependent on EMR-administered information) has been articulated by Shahar in 2002 [210].

> "There is a clear need for effective guideline-support tools at the point of care and at the point of critiquing, which will relieve the current information overload on both care providers and administrators. To be effective, these tools need to be grounded in the patient's record, must use standard medical vocabularies, should have clear semantics, must facilitate knowledge maintenance and sharing, and need to be sufficiently expressive to explicitly capture the design rational (process and outcome intentions) of the guideline's author, while leaving flexibility at application time to the attending physician and their own preferred methods."

Shahar wrote this statement within a briefing paper in the context of the foundation of an initiative that is known as "OpenClinical". It is a "not-for-profit organisation created and maintained as a public service"[3]. At that time, several DSS approaches were already available. OpenClinical is an online-community that maintains an extensive archive of artificial intelligence systems that are used in routine clinical working practice or that are current research. For example, well-known healthcare approaches to DSS are the Arden Syntax, GUIDE, PRO*forma*, Asbru, EON, or PRODIGY. OpenClinical gathers DSS-related products and research and provides a regularly updated and comprehensive overview that can be accessed online[4]. The online overview provides all necessary scientific references for each approach.

Paul de Clercq et al. [211] summarize four guideline-related challenges. These are guideline acquisition, guideline modelling and representation, guideline verification and test, and guideline execution. At the outset of guideline support, the guideline acquisition is a major problem because it requires to translate narrative guideline documents into computer-interpretable form (cf. sect. 1.1.4). In its final step, guideline execution results in a conceptual overlapping with workflow enactment by Workflow Management Systems (WfMSs). The diagnostic-therapeutic cycle (cf. sect. 1.1.3) universally illustrates that each decision making implies a workflow consequence in form of a subsequent diagnostic or therapeutic measurement. Thus, there is an intrinsic terminological overlapping between DSS concepts and WfMS concepts. From the perspective of DSS, this overlapping is illustrated by Mor Peleg et al. in [212]. Her comparative analysis provides an overview on terms that are used by guideline modelling methods. The overview is reproduced in table 3.1 for illustrative purposes.

In a nutshell, the commonality between the different approaches is that they consider the overall patient-individual therapy as a *plan* of recommended *actions*. In other words, the outcome of DSS is a therapy plan that can be considered as a workflow structure, which is represented as a work-list being devised ad hoc. The primary function of DSS is to automatize the ad hoc composition of a recommended therapy plan. The pre-eminent problem of automatized medical guideline support, in extension to traditional knowledge-based approaches, is that it requires para-consistent logics [213] that provide techniques "for reasoning in the face of uncertainty or ambiguity" [214].

In the context of the foundation of OpenClinical, Panzarasa and Stefanelli coined the well-known term "careflow systems" in 2002 [215], so to speak as part of the community-building intention of OpenClinical. One of their later publications has been titled "Workflow management systems for guideline implementation" [216], which can be taken

3 cf. `http://www.openclinical.org/about.html`
4 cf. `http://www.openclinical.org/gmmsummaries.html`

		Plan component					
Model	Plan	Branching	Action	Decision	Scenario	Special	Subplan
Asbru	Plan	Plan type	Plan	Plan precondition			Recursive plan
EON	Management Guideline	Branch Synchronization	Action	Decision	Scenario		Subguideline step
	Consultation Guideline	Consultation—branch	Consultation—action				Consultation guideline part of scenario
GLIF	Guideline, Macro	Branch Synchronization	Action	Decision	Patient-state		Guideline or Macro called in Action or Decision steps
GUIDE	Guideline	Synch-&, Synch-Or	Task	Deterministic decision, nondeterministic decision		Wait Monitor	Any task can be decomposed
PRODIGY	Decision/ Management map		Action		Scenario		Subguideline Step or called in Action step
	Consultation Template	Consultation—branch	Consultation—action				Consultation template part of scenario
PRO*forma*	Plan	Action, Enquiry, Decision	Action, Enquiry	Decision			Plan task

Table 3.1: Terms used by guideline modelling methods (adopted from Peleg et al. [212])

literal for the overall careflow approach. In a nutshell, the careflow conception favours a technological system blend of decision support and workflow enactment. The dominant perspective is the one from a single clinical environment. From such perspective, the careflow demands are justified and to achieve the necessary degree in system integration may be expensive but a successful accomplishment seems realistic. However, the problems that are caused by trying to implement a careflow system are still generally unsolved, as it is summarized by Miller and MacCaull in [214].

From an inter-institutional perspective, a careflow approach remains impractical as it depends on unsolved inter-institutional data integration and functional integration. Besides this limitation, additional challenges arise from inter-institutional knowledge integration. Both already mentioned publications from de Clercq [211] and Peleg [212] are well suited surveys on available representations of clinical guidelines. Standard formats for guideline representation are a prerequisite of guideline implementation. Additional publications on sharing and integrating knowledge representations can be found by Boxwala et al. [217], Wang et al. [218], and Peleg et al. [219, 220]. Lenz summarizes integration-related problems of guideline representations in [65]: "the integration problems to be solved are a matter of semantics rather than format". Imam and MacCaull provide an overview on the semantic problems of merging ontologies in [221, 222].

In conclusion, system support for clinical guidelines concerns multiple levels. Inter-institutional knowledge integration remains an unsolved issue. Finally, overall system integration challenges are not solved by DSS. Local system integration is just achieved by tailor-made software development efforts. Finally, tightly-coupled careflow approaches increase integration challenges, both in the context of local sites and particularly in inter-institutional scenarios.

3.1.4 Conclusion

In distributed environments, the architectural style of asynchronous messaging or synchronous interface-oriented invocations is commonly applied to achieve data integration and functional integration. In medical informatics, the history of messaging frameworks and interface standardization efforts by HL7, IHE, and others reflects this general trend. Document-orientated system integration styles are a more recent trend. In healthcare, from the perspective of data integration, HL7 CDA provides a platform for canonical data models based on document-orientated methods. CDA has already been successfully used as a foundation for specialized content standards like CCD and the SCIPHOX specifications. Process-related status information is not targeted by the standardization efforts for these document formats. From the perspective of functional integration, standards for document exchange like XDS are scarce in healthcare; decentralized ones are missing.

From the perspective of knowledge integration, the automation of decision support is feasible in local and seamlessly pre-integrated system environments. Infrastructures that are suitable for nation-wide healthcare scenarios with non-centralized large-scale requirements are an open issue.

3.2 Activity-Oriented Workflow Approaches

Basic activity-oriented workflow terminology and concepts have already been discussed in the methods chapter (cf. sect. 2.2.5). This section provides an overview of the capabilities and limits of contemporary approaches.

Business Process Modelling (BPM) became an established field of computer science in the 90s. One of the pioneers in process description is Prof. Scheer. In 1992, his research group invented the notation of "Ereignisgesteuerte Prozesskette" (EPK), in English Event-driven Process Chain (EPC). EPC is a semi-formal process description model. It is part of the ARIS™ methodology and creating EPC diagrams was soon supported by the ARIS™ toolset. EPC diagrams became quite popular because ARIS™ became the foundation for several BPM tools and was also adopted by SAP.

In order to formalize processes descriptions into workflow schemas, general-purpose methods for formal system modelling, like statecharts or Petri nets, became popular. A pioneer in formal workflow modelling is Prof. van der Aalst, who initially used Petri nets and later created Yet Another Workflow Language (YAWL) in 2004. YAWL is based on Petri nets but provides a semantically rich set of workflow constructs.

In 1993, the Workflow Management Coalition (WfMC) was founded. Its first standard was Workflow Process Definition Language (WPDL) in 1998, which had an XML-based successor in 2002, the XML Process Definition Language (XPDL). Today, there are still several competing workflow languages. The Unified Modeling Language (UML) also provided a platform for workflow modelling, either with UML state machine diagrams or with UML activity diagrams. In UML 1.x the activity diagram type was actually based on state-machine semantics but since UML 2.x it is based on Petri net semantics.

For workflow execution, the best-known standard is Web Services Business Process Execution Language (WS-BPEL). As a workflow language, WS-BPEL is not graph-based but block-structured. In block-structured workflow languages, control flow is defined similar to programming languages by using constructs like *if* or *while*. Kopp et al. describe the implications of using block-structured instead of graph-based languages [223]. WS-BPEL stems from IBM and Microsoft who combined their WSFL and XLANG efforts in 2002.

Finally, similar to the UML activity diagram specification is the Business Process Model and Notation (BPMN) specification. BPMN was originally developed by the Business Process Management Initiative (BPMI) but is currently maintained by the Object Management Group (OMG)[5] who also maintains UML. Since the release of BPMN version 2.0 as of January 2011, BPMN has become the most prominent standard for activity-oriented business process modelling, workflow modelling, and workflow execution.

3.2.1 Outline of Activity-Oriented Modelling with BPMN

The initial intention of BPMN has been to provide business analysts with means to illustrate business process models with a semi-formal graphically notation. Since BPMN 2.0, the semantic standardization of elements has been extended such that technical developers are now enabled to refine the diagrams into executable workflow schemas, which can still be illustrated in graphical BPMN notation. In conclusion, the scope of mainstream activity-oriented BPM and workflow languages can be illustrated by the example of BPMN. It provides a rich set of elements that is distinguished by specification [224] into different categories: 1) flow objects, 2) connecting objects, 3) swim lanes, and 4) artefacts.

The first category for flow objects contains event types, gateway types, and activity types. For illustrative purpose, a basic overview of this category is provided by figure 3.3. The figure is comprehensive and lists all elements of the first category but the BPMN repertory of event, gateway, and activity types will not be further explained in this context. However, it should be highlighted that BPMN supports nested sub-processes (e.g. "collapsed sub-process" in figure 3.3). Another notable aspect concerns workflow element instantiation. In workflow approaches like state charts and Petri nets, workflow elements are unique within a workflow schema and are only instantiated once for each workflow instance. BPMN explicitly allows for the instantiation of multiple activities of the same element type at run-time within a workflow instance (cf. "multiple instance" in figure 3.3)[6]. Concepts like composition and instantiation lead to object-oriented programming considerations. For the sake of completeness, it should be mentioned that BPMN does not support inheritance, e.g., conceivable between activities or between sub-processes. Inheritance is generally unsupported by workflow languages and a discussion about the overall potential for inheritance of dynamic behaviour within the context of workflow modelling is available by van der Aalst in [225].

5 BPMI and OMG merged in 2005.

6 For example, XPDL also allows for multiple instantiation of elements. The UML activity diagrams support it, too, on the basis of the "Expansion Region" concept.

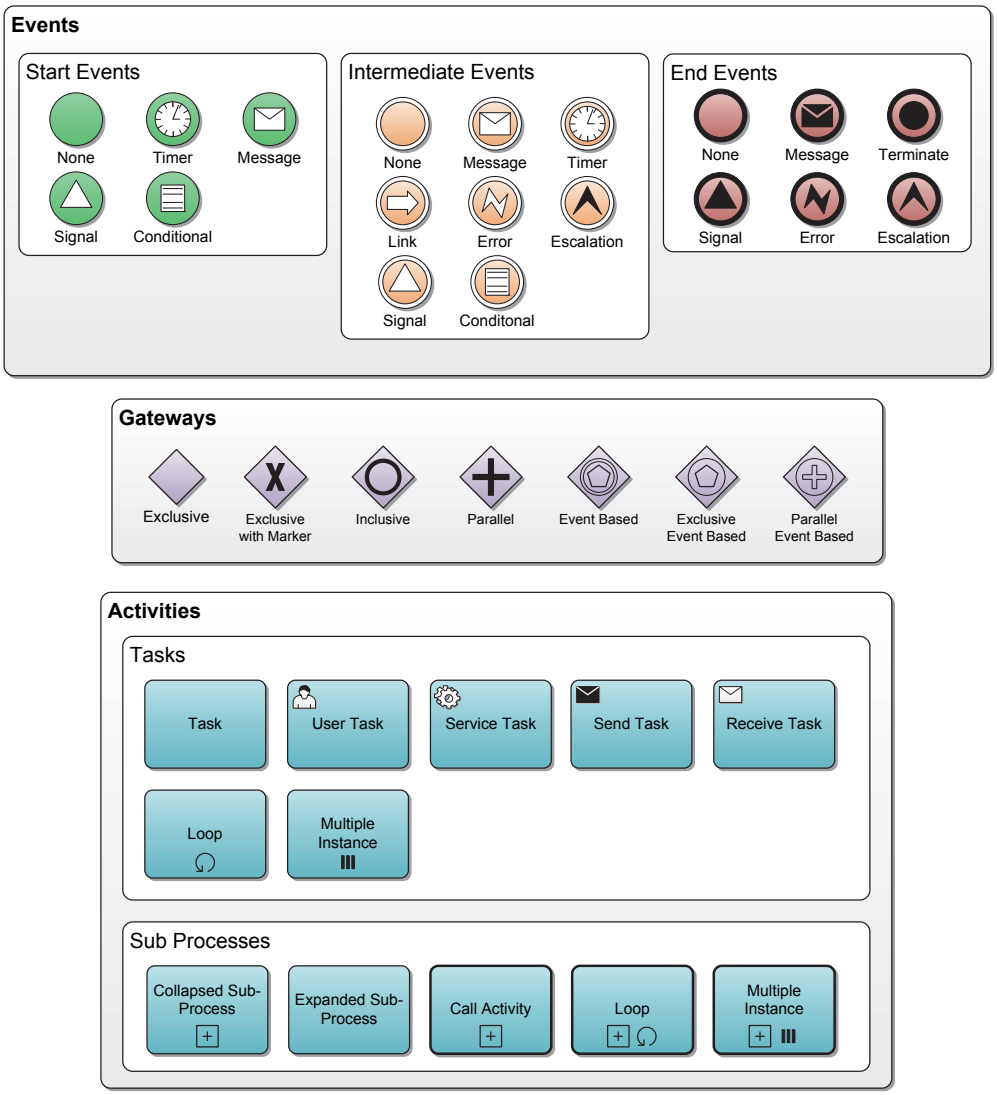

Figure 3.3: BPMN language: the category *flow objects* and its model elements

The second category for connecting objects includes different types for sequence flow, message flow, and associations. For the message flow, this category includes the element type for message. (Message is not related to data object, an element type that will be discussed soon.) The Message can be named, it is unstructured, and it is always associated to a message flow.

The third category for swim lanes includes both the pool and the lane. Pools are process participants. Both pools and lanes can represent responsibilities for activities and can be an organization, a role, or a system. Lanes subdivide pools or other lanes hierarchically. The most important distinction between pools and lanes is provided by a restriction:

only pools can communicate via message flow, lanes within the same pool are forbidden to connect via messages.

The fourth category is known as artefacts. Yet, the term "artefact" just implies ancillary elements that provide additional process documentation without effecting its semantics. In the BPMN 1.1 specification, the category included group, annotation, and data object. Group is used to arrange different activities visually without effecting workflow semantics. The annotation is used for free text comments. The data object represents any information that flows through the process, such as method parameters, database records, XML-structured business documents, or binary letters. That data objects had been considered non-effective on workflow semantics signifies on the inferior role of data in traditional workflow approaches.

Since BPMN 2.0 there is a fifth category of elements: 5) data. The data category includes four elements: data objects[7], data inputs, data outputs, and data stores (it does not include element type message for message flows). Data objects are primarily put in association to the sequence flow in order to optionally annotate data flow anywhere within a process. Data objects still represent any kind of data type and BPMN does not itself provide a built-in model for describing structure of data. The data inputs and outputs define data requirements for activities, but not every activity type is allowed to use these elements. The data store represents data that persists beyond the scope of the Process but BPMN does not itself support any expression language for querying data. In conclusion, the neglect of data structures by activity-oriented workflow models is still prevalent even if we can observe some convergence.

For further reading, the specification itself [224] is a definitive source, which provides several examples. BPMN is a 300+ pages specification. It is elusive to think that even dedicated business analysts will be able to master all these concepts. Books on BPMN 2.0 with high reputation are, for example, from Silver [226] as well as from White and Miers [227]. A German book on BPMN 2.0 is provided by Allweyer [228]. Silver comments in [229] from experience "BPMN has a lot of attributes put in there just for BPEL generation, and these are generally ignored". Michael zur Muehlen has run a survey of the most used constructs in BPMN and his conclusion was that about 25 constructs are routinely used [230, slide 24]. However, it is unclear if these experiences concern only usage for BPM by business analysts or also usage for workflow automation and execution.

7 After introducing the data category and moving data objects into it, the artefacts category in BPMN 2.0 just contains group and annotation. Still, BPMN vendors are invited to extend the artefacts category by further elements.

Illustrative Example of a BPMN Diagram

As complement to the former abstract description on all the various elements of BPMN, I want to provide a short example diagram that uses basic elements from all the available categories. The initial treatment episode of breast cancer is outlined in figure 3.4. The process involves four participants; an ambulant gynaecologist, an ambulant radiologist, a clinical gynaecologist, and a pathologist. If there is evidence for breast cancer in the end, then the process is a succession of a manual examination, sonography, mammography, biopsy, and histology.

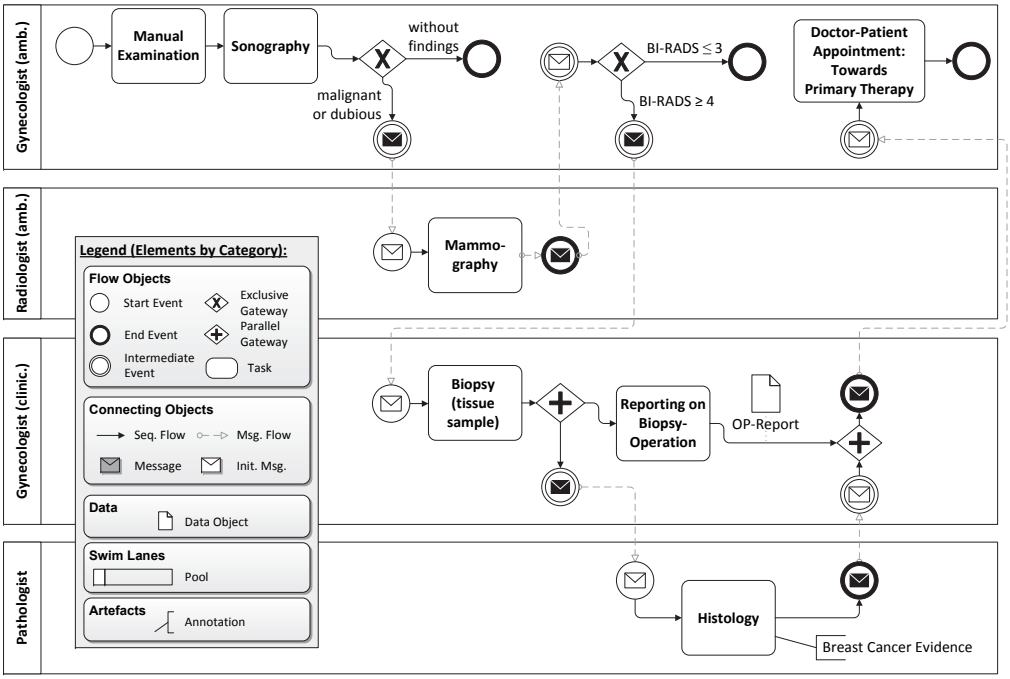

Figure 3.4: BPMN example (initial breast cancer treatment episode)

A comprehensive legend provides the names of the used element types as well as an explicit correlation to their BPMN category. I will not explain the detail of the process at this point but use it as an illustrating example for the graphical notation of BPMN. For example, in the context of the reporting on the biopsy operation, the sequence flow is annotated with a data object that represents the respective report.

In a student research project with Benedikt Lempetzeder [231], we modelled the initial breast cancer treatment episode with several BPM and workflow languages. Amongst others, with EPC, Petri nets, YAWL, and UML activity diagrams. His thesis includes an equivalent diagram for each notation, an evaluation of their individual specifics, and an analysis of their data flow capabilities. I will revisit data flow in section 3.3 for providing

a brief comparison of the various data flow capabilities both of the activity-oriented and content-oriented workflow approaches.

For enacting the BPMN diagram by workflow technology, it must be refined into an executable BPMN schema. Furthermore, each of the four participating sites would be required to install and maintain a BPMN workflow engine and to establish the necessary messaging channels. The number of primary care offices with an IT infrastructure that supports BPMN (or any other workflow language) is not known but it can be assumed negligibly low. In clinical scenarios, the application of BPMN, for example for clinical pathways, is easier because all participants share a single organizational context and commonly a centrally administrated IT infrastructure. Thus, a multi-participant workflow can be managed by a single central BPMN workflow engine.

3.2.2 Limitations of Activity-Oriented Workflow Languages

Any diagram that is based on an activity-oriented workflow language will naturally emphasize the activities, not on the data artefacts that are produced. The general-purpose is to standardize a workflow and to identify tasks for automated execution. Yet, in healthcare processes, there usually is not a single medical task that can be automatized. Each task in the process model describes intent. This process intent is described on a coarse-grained ingenuous level; exceptional workflows are absent. In healthcare, process standardization has its limits.

The example in figure 3.4 only requires bilateral data exchange, thus, we can model it cleanly. In order to dispatch a message to multiple receivers, the modelling of multiple bi-lateral message flows is required, which makes diagrams merely cluttered or even unmanageable. All receivers must be known to workflow design time. Neither BPMN nor any other mainstream workflow language directly supports multilateral message exchange or a message broadcast between participants by an explicit model element. Multilateral data distribution still can be achieved by a central data store, even with an a priori unknown set of “receivers”. Yet, the data store is a black box and it remains unclear how data synchronization can be achieved across independent sites.

From the perspective of inter-institutional cooperation, the initial breast cancer treatment is just one successively written report that is subdivided into the report contributions of the several participants. Process support for inter-institutional scenarios must emphasize supporting the articulation of data demands and the multilateral distribution of data. Such is not the focus of activity-oriented workflow languages and support is currently missing.

3.2.3 Ad-Hoc Sub-Processes: Coping with the Unpredictable?

The *ad-hoc sub-processes* in BPMN provide a mechanism to model a group of activities that have no required sequence relationships. Each included task can be instantiated several times in any order. The sequence and number of performances is determined by the performer. The availability of this element is unique to BPMN in comparison to its activity-oriented competitors. The graphical symbol is illustrated in figure 3.5. The ad-hoc sub-processes are known as "black holes" of the workflow schema. Re-entrance into the well-organized world of the workflow engine is achieved by a termination condition that is explicitly required for ad-hoc sub-processes.

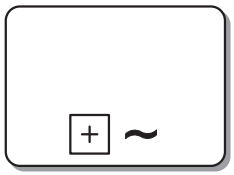

Figure 3.5: The BPMN element type for an ad-hoc sub-process

The ad-hoc sub-process element is intended for human processes and/or knowledge-intensive processes. They seem promising but there are several limitations, yet. It does not allow the dynamic adding of tasks to its set of tasks. Only tasks can be performed that have been included at design-time. It is not allowed to include start or end events, which forbids including composite sub-processes within an ad-hoc sub-process element.

A special requirement for human-oriented workflows is the possibility to define a partial ordering in which each activity must be executed once, most in arbitrary order but some in a strict sequential order. This type of problem is known as Interleaved Parallel Routing (IPR). A product evaluation for IPR-support is available online[8]. The majority of workflow engines does not support IPR. In the context of a BPMN ad-hoc sub-process IPR is neither considered nor is it possible, instead, either strict sequential or arbitrary parallel execution can be modelled.

In conclusion, BPMN recognizes the importance of ad-hoc sub-processes. Still, the specification itself [224, p. 183] in fairness concludes that ad-hoc sub-processes are either not executable by a BPMN workflow engine or that the responsibility should better be delegated to a groupware system. As a workflow element, the ad-hoc sub-process provides design-time-based "ad-hoc-ness" but it does not provide run-time-based "ad-hoc-ness". BPMN does not provide a model to monitor the status of ad-hoc sub-processes. There is also no support for an a priori unknown set of workflow steps or participants.

8 http://www.workflowpatterns.com/patterns/control/state/wcp17.php

3.2.4 Contemporary Research in Activity-Oriented Workflows

Current workflow management research topics that are under discussion by the community are interacting processes and multilateral messaging. A basic model for interacting processes is included in BPMN 2.0 via conversation and choreography diagrams.

Stiehl, in his PhD thesis [159], provides workflow patterns that enable multilateral messaging in BPMN. For this purpose, the workflow patterns explicitly include *routing sub-processes* into the overall workflow schema. The routing sub-processes instrument rule-based tasks to look-up sender/receiver lists. Thus, the workflow pattern emulates the capabilities of an enterprise service bus using the elements of the BPMN language. Still, the concept requires to generate *correlation IDs* and to maintain a *cross-reference-table*. For implementation, he uses concepts that are not yet standardized by the BPMN specification but that are available in the SAP BPMN engine.

The research approach Proclets, by van der Aalst et al. [232, 233], provides an advanced model to interacting processes. Proclets[9] exchange messages, named *performatives*, via *channels*. The Proclets approach proposes a shift in focus from control flow to communication in order to reduce control flow complexity. Still, a dynamically changing and a priori unknown set of participants is not considered.

Besides interacting processes, the formal verification of workflow schemata and the dynamical monitoring of its workflow instances are an enduring issue. For example, in an approach named DocSerFlow [234], van der Aalst has applied methods of linear temporal logic on modelling workflows in so-called *service flows*. Linear temporal logic, e.g., [235], provides means of model checking to automatically verify[10] that a (concurrent) protocol satisfies its specification in terms of absence of deadlocks or similar critical states that could cause the system to crash. The DocSerFlow model provides a rich set of formal relations between events. For example, these event relations are either affirmation or negation of event successions, event responses, or event co-existences. The DocSerFlow event relations are not integrated in van der Aalst's YAWL and have no equivalent in BPMN or other mainstream workflow languages.

9 From an implementation perspective, Proclets had been based on Petri nets and later on YAWL.

10 Sistla and Clarke proved in [236] that checking the validity or the satisfiability for linear temporal logic is a PSPACE-complete problem.

3.2.5 Résumé

Adaptiveness in Activity-Oriented Approaches

Neither Proclets nor BPMN support adaptive change of the sequence flow, data objects, or message structures. In contrast, adaptive workflows are discussed for ADEPT$_{flex}$ [237] by Reichert and Dadam. ADEPT$_{flex}$ is based on a block-structured process description. Change operations in ADEPT$_{flex}$ consider only the control flow. Data flow is an addendum to the control flow and the exchange of data between tasks is based on global variables. Data elements are derived from input/output parameters of tasks. Users can extend the data structure not directly but by inserting new tasks with according parameters or by replacing tasks. This raises a variety of challenging issues with respect to dynamic parameter mapping (cf. [237]). It also leaves significant complexity to the user.

Conclusion on Activity-Oriented Workflow Approaches

Activity-orientation focuses on the sequencing of activities (i.e. control flow). The workflow management research groups as well as vendors, products, and standard setting bodies have made great achievements in providing process enactment engines. Models, notations, and tools allow for process control, workflow automation, and process monitoring. Support for workflow automation is provided by tool support for IT alignment that is to map the workflow schema with existing system interfaces. The whole workflow life-cycle is covered in terms of model, execute, monitor, analyse, and improve.

Amongst others, a workflow designer can easily model task synchronization and simulation tools support deadlock analysis and prevention. A rich set of workflow elements is available. For example, loops and timers allow for repetitive tasks or for scheduled tasks. Various gateway elements allow articulating necessary workflow decisions. Comprehensive event taxonomy includes, for example, event types for signalling and for handling exceptional workflow paths. There is hardly any real-world process whose structure of activities would not be representable by the established workflow approaches like BPMN that have become mainstream. The collective achievements of traditional workflow management recently allows for *process-oriented information systems* in which the system design of the software applications is driven foremost by workflow considerations.

However, it remains very cumbersome to articulate the implication of the control flow on its underlying data using activity-oriented models and notations. These often ignore the informational perspective (i.e. data flow) or consider it only in the context of single tasks. Consequently, an overall view on the process-implicated data units is missing. In addition, schema evolution is quite a challenge. Support for ad hoc processes with

unpredictable activities and participants, which cannot be captured within a workflow schema a priori, remains an open issue. The prime purpose of activity-oriented WfMSs is a system-centric workflow automation. Cooperation of knowledge workers requires a different kind of process support.

3.3 Towards Content-Oriented Workflows

The content-oriented approaches to workflow modelling substitute activities with placeholders for data and data dependencies. Progress in data production implicates progress in the real-world process with its activities. In fact, the focus is shifted from "activities with underlying data" to "data with underlying activities". The main characteristic in content-orientation is to separate the data structure from the process structure, and to support formal bindings between data state and process enactment. Thus, a key point to content-oriented workflow approaches is the maintaining of an overall view on any process-implicated data units.

It is necessary to stress that all related approaches are research and work in progress, as I mentioned in section 1.2.6 of the introduction. There is no overarching conceptualization for the various approaches. The term "content-oriented workflows" is my umbrella term. Thus, the key concepts are first illustrated independently of any specific approach. This section has an informal and illustrative purpose, providing several examples. Thereafter, the individual research projects will be discussed in the subsequent section.

3.3.1 Introductory Example: Life Cycle of Content Units

An introductory example is the write-and-review scenario. Write-and-review is the most elementary example for content-oriented process progression. It might involve only a single content unit, still representing a multi-participant process by means of its different content states. Write-and-review scenarios can be easily understood independent of any domain, thus, they appear in some of the content-oriented research publications for illustrative purposes.

In a publishing scenario, content states are publication-editing states like draft, submitted, reviewed, and published (cf. fig. 3.6). The write activities and review activities themselves are not modelled but are implicit: sitting in an office or at home, doing the writing or reading. The implicit activities are more complex than they might appear prima facie. Various examples for tacitly accomplished writing activities would be editing text, generating graphics, including graphics with captions, finishing chapters, or organizing references. Tacitly accomplished reviewing activities, for example, subsume

verification of facts, proofreading of orthography and grammar, revising the hyphenation and typesetting, checking up on the layout composing, or even validating the colour with so-called prepress proofs. In conclusion, changing data and data states implies well-educated knowledge workers.

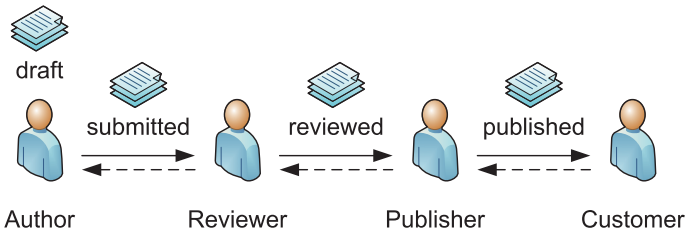

Figure 3.6: A write-and-review scenario

Obviously, there are different types of workers with different roles, similar to activity-oriented workflows. Thus, a content-oriented workflow model for write-and-review processes is required to formally articulate for each state which authority is allowed to change or revert the state into a successor or predecessor state.

Write-and-review processes are commonly single-artefact scenarios with only a few articulated content states. State changes are commonly linear and each state change indicates that the publication is delegated to the next type of process participant. This special type of content-oriented workflow is common enough and simple enough that it does not necessarily require an explicit workflow layer but is implicitly implemented by many authoring systems. Well-known examples are content management systems[11], conference/submission management systems[12], and code review systems[13] for software projects. Still, any of these systems only provide a vendor-specific and hard-coded sequence of activities with predefined editing types or review-state types.

The introductory example has illustrated the life-cycle of a singular content unit. It has shown that the underlying activities that lead to state changes are not necessarily explicitly listed but are just tacitly applied. An additional example, the job application example, will extend the basic write-and-review scenario with more complex content-to-content and content-to-actor interdependencies. Thus, it will be motivating the demand for customizable, general-purpose, content-oriented workflow models and notations.

11 For example, the open-source Content Management System (CMS) Alfresco, the open-source CMS Zope with the DCWorkflow extension, as well as the commercial system Documentum by EMC.

12 For example, the open-source conference management system OpenConf as well as the free of charge system EasyChair or the the commercial system EDAS by EDAS Conference Services LLC.

13 For example, the open-source code review system ReviewBoard or TeamViewer as well as the commercial system Crucible by Attlasian or CodeCollaborator by SmartBear.

3.3.2 Revisited: Data Flow

Not only is the distinction to activity-oriented control flow of interest. There are also distinguishing features to traditional data flow. Data flow modelling is older than workflow modelling. Data-Flow Diagrams (DFDs) were introduced and popularized for structured analysis and design by Gane and Sarson in 1979 [238]. It is a semi-formal boxes-and-arrows notation for illustrating directed data associations between data sources, computing processes, and data sinks. Several derivatives exist, for example, the Information Flow Diagrams (IFDs) that were introduced for the Soft Systems Methodology (SSM) by Checkland and Scholes in 1990 [239]. It is a notation that emphasizes the relationship between external and internal information between organizations, systems, or sub-systems.

As we have seen, workflow management systems appeared in the 90s. The data flow was either not supported (e.g., Petri nets) or it was only implicitly supported by associating data with control flow edges or tasks (e.g., EPC or BPMN data objects). Sometimes the notation does not support data flow but it is still supported, technically, by shared variables (e.g., ADEPT). Only a few workflow languages support data flow explicitly, for example, UML activity diagrams as well as BPMN message flow within its limiting boundary conditions, like "messages only between pools but not between lanes".

A data flow models the passing of a statically defined data type from a producer to a consumer or from task to task. In addition, the notion of the life cycle of content units is prominent in content-oriented models, while it does not exist in data flow approaches.

Content-oriented models, on the other hand, might not model any data flow. All data units within a process instance are often assumed accessible to all workflow participants, i.e. actors/tasks/functions, without the need to explicitly pass data between them. Instead of data flow there is a kind of "data-authority flow", as it has already been indicated by the introductory write-and-review processes. Yet, this flow is often implicit by associating a workflow participant as the authority for a content unit in dependency to one or several of its states. The "flow" of data-authority between data states is currently not charted by content-oriented workflow approaches.

3.3.3 Illustrative Example: Job Application

An illustrative example for content-oriented workflow modelling is based on the job application scenario. In contrast to the basic write-and-review scenario, in job application processes there is a set of content units for a single process.

In order to illustrate the different emphasis of activity-orientation and content-orientation, figure 3.7 provides an informal drawing of the job application scenario. The purpose of the diagram is to demonstrate that a workflow can be modelled perfectly well with both types of approaches, in principle. The dashed arrows in the left-hand activity-oriented part are for input/output data flow. The dashed arrows in the right-hand content-oriented part are for data authority associations. The notational focus becomes shifted by making content units the first order model elements. The following narrative will only describe the content-oriented conception[14].

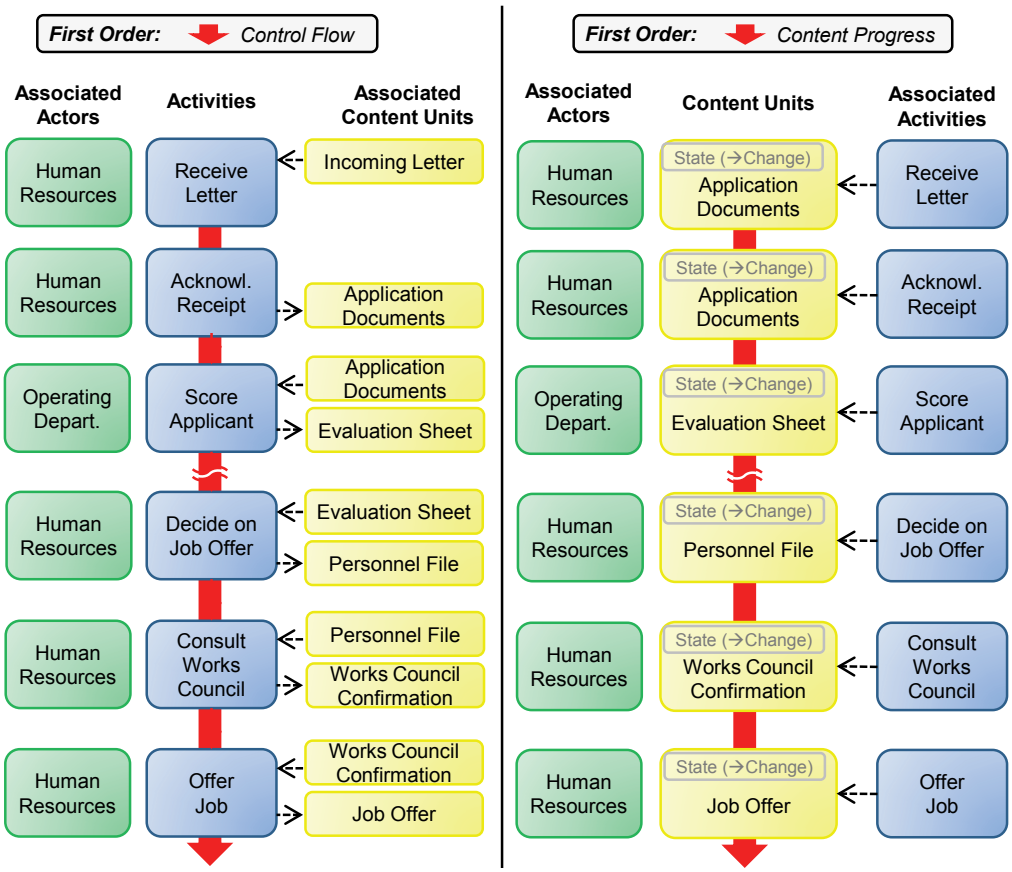

Figure 3.7: Job application: activity-orientated vs. content-oriented perspectives

Initially, the receipt of an application, i.e. a letter with appendices, initiates a process instance. The state of the application content unit is received. All applications with state received are viewed in the work-list of the human resource (HR) department. For example, HR creates a letter for the acknowledgement of receipt. The letter of acknowledgement might run through several states, as in write-and-review, let us assume its final state is

14 Dubray explains in [240] several aspects of inadequacy if using BPMN for this scenario.

submitted. Now, based on the letter of acknowledgement being set into state submitted, a kind of trigger or workflow rule might automatically change the application state into receipt_acknowledged.

Now, any job application whose state changes into receipt_acknowledged will be viewed in the work-list of the operating department. They create an evaluation sheet and fill in scores of their application assessment. The authority for inviting the applicant to an interview will depend on the organization. Let us assume that HR decides about the interview if the evaluation sheet is in state final and the applicant's overall score is above some reference mark. Let us skip the letter of invitation to the interview, possible sheets for the scoring of the interview, and the decision taking about actual employment. Nevertheless, assume that the positive decision results into setting the application state into designated_for_employment.

In our imaginary organization, the process now has progressed far enough such that a personnel file is created because we require a works council decision. Works council decisions are, in our organization, always documented within the personnel file, even if the applicant would later decline the job offer. The personnel file itself is set to the state preliminary. The works council needs to be asked, thus, all applications with state designated_for_employment are viewed in its work-list. Eventually, the work council provides written confirmation within the personnel file, and the state of the application is changed into offer_employment.

Finally, the HR department can write an offer to the applicant (e.g., application in state job_offered). He or she can accept or reject (with according states). A rejection may lead into further negotiation similar to write-and-review process (e.g., with offer revisions). On positive conclusion, the state of the personnel file might be changed into effective. The episode ends if the decision is final.

Finally, even if basic workflow aspects can be articulated with both types of approaches, activity-oriented or content-oriented, there are subtle differences. In activity-orientation, the required data objects and messages are not necessarily explicitly specified but might be tacitly assumed. If a workflow schema is refined for automatized execution, both the system invocations and the data flow are not necessarily reflected in the workflow notation but are programmed within the enactment environment of the engine provider. The first-order workflow elements are the predefined activities and their sequence in form of the control flow. In content-orientation, it is the other way around. As it has been described previously for work-and-review, the required activities are not necessarily explicitly specified but might be tacitly assumed. The first-order workflow elements are the predefined content units and their allowed state transitions. Thus, the type, number, or sequence of background activities can be altered on demand, as long as human tasks are concerned and as long as workflow automation is only of subsidiary concern.

3.3.4 Circulations

In German bureaucratic environments, circulation folders are used for informing departments and for getting approval. Prinz and Kolvenbach analyzed paper-based circulation workflows in ministerial environments during the 90s (cf. [241]). For circulation purposes, the paper-based envelope contains a cover note that lists the recipients of the workflow in hierarchical order. Each recipient approves the document with his signature in the recipient list of the cover note. Additionally, comments or advices are included as annotations to the text. Usually the circulation envelopes are transported by in-house postal service. In urgent cases, the envelope is transported personally by the secretaries between the department levels.

An interesting peculiarity of German ministries is described by Prinz and Kolvenbach: Each role in a ministry is assigned with its own colour, for example, the sub-department manager uses brown, the department manager blue, and the minister green. They must use their colour to make signatures and annotations. The purpose is to simplify recognition at which hierarchical level a comment was made to a text. From the perspective of the earlier discussion about coloured Scrum task cards (cf. sect. 2.2.11), the ink colour of the pencils is equivalent to an adornment to each signature and annotation.

Electronic circulation envelopes enable the forwarding of documents along a specific path through the organization. The primary operation that is applied to a circulation folder is to forward it to the next in the recipients list. It is possible to mingle interactions of write-and-review semantics into circulations, which means that an office worker can send the envelope back to his or her prior in the recipients list. The initial recipients list might only name the departments, thus, the envelope is routed to the according secretariats. Often, decisions about actual office worker recipients that are subordinate to the current department are deferred until the envelope arrives at its secretary. Thus, additional recipients are exploded and added during circulation time. A common problem in circulation workflows are unavailable people, which is basically caused by holidays or by sick leaves. A key question in circulation workflows is “Where is the file?” and this question can be supported by electronic circulation systems.

Electronic circulation systems can be tailor-made based on CMSs like Alfresco, Microsoft Sharepoint, or EMC Documentum[15]. However, tailor-made applications are just a limited compensation for a systematic approach and a general infrastructure to manage circulations. There are only few research approaches that provide a process model for circulations. The available ones originate in the 90s and are discontinued. These are

15 An electronic circulation solution that is built upon Documentum is, for example, marketed by the soft Xpansion GmbH (cf. `http://www.soft-xpansion.com/index.php?p=docman/documentum`).

ProMInanD and POLITeam, which will be discussed in section 3.4.5 in comparison with those scientific workflow approaches that can be considered as content-oriented.

The essential characteristic about paper-based circulation workflows is that they are ad hoc workflows. For the previous write-and-review workflows or in scenarios like the job application, it is perfectly suitable in most cases to have an a priori known list of required content units, associated activities, and required participants. In circulation workflows, the dynamic re-routing of the circulation folder between participants is dominant. The set of content contributors and the resulting set of content units are not necessarily known at circulation initiation.

3.3.5 Conclusion

The purpose of section 3.3 has been to provide an overarching understanding of the concepts that converge into the idea of content-oriented workflows. The key understanding is about representing workflow progression by transitions between content unit states. The next section discusses related workflow approaches.

3.4 Content-Oriented Workflow Approaches

The term "content-oriented workflows" is my umbrella term for several scientific workflow approaches (cf. sect. 1.2.6). The common feature of content-oriented approaches is to articulate workflows based on content unit states and content dependencies and to interpret progress in content production as the equivalent of progress in the real-world process. The next subsections will provide a short characterization of each approach. The main workflow concepts and model elements are highlighted. If available, examples for the notation are provided. The notations are mostly semi-formal or informal, just as they are provided within the respective publications. The following list provides an overview of the approaches each with representative publications:

- the "data-driven" approach [242–244]
- the "resource-driven" approach [245, 246]
- the "artifact-centric[16]" approach [247–249]
- the "object-aware" approach [250–252]

16 It is important to use the American English "artifact-centric" and not the British English "artefact-centric". Otherwise any literature research would be constricted. Thus, I selectively keep "artifact-centric" in American spelling.

As we are going to see in this section, the content-oriented workflow approaches provide a great contribution to the domain of workflow modelling, still, they are not suited for inter-institutional scenarios, at present. The self-portrayal of all approaches is still fragmentary. Nevertheless, the survey intends to provide an all-over impression on the varying capabilities. At the end of the section, a comparative analysis will be conducted that results in a taxonomy of distinguishing characteristics in order to classify content-oriented workflow approaches.

3.4.1 The "Data-Driven" Approach

The *data-driven process structures* have been developed at the department of computer science at the university of Twente, Netherlands, in cooperation with DaimlerChrysler. The protagonists have been Dominic Müller and Manfred Reichert, both from the university, as well as Joachim Herbst, from DaimlerChrysler Research. The primary implementation is COREPRO[17]. The project started in 2005 and has not been maintained after 2007. (Still, Reichert continued his ideas later at the university of Ulm in form of object-aware process management, as we will discuss soon.)

The primary focus of the "data-driven" approach is to implement a Release Management Workflow (RLM) in the automotive industry (cf. [242]). The goal is to allow for IPR, i.e. parallel but synchronized execution, of concurrent engineering processes. For process synchronization, the data objects themselves specify which functions or activities are allowed to work on them. A basic premise is that there is a hierarchically structured product like an automotive. Figure 3.8 provides an example from [243]. On the left hand side, the product structure with its sub-systems is outlined.

The basic assumption is that the dependencies between different sub-processes of a process structure typically base on the assembly of the product to be manufactured (i.e. "bill of material"). Both the total system ("total system: S-Class car") and each sub-system have a sub-process (from encircled and grey-filled start state "S" to end state "E"). In general, the three major phases of data-driven RLM are configuration management, testing/validation, and release. Each sub-process reflects these three phases (the yellow circles, especially "T"/"V" for tested or validated and "R" for released). Each sub-process can be modelled by UML activity diagrams, and state transitions within a sub-process are called *internal state transitions* (the arrows with solid lines). The interaction between sub-processes is based on additional *external state transitions* (the arrows with dotted lines).

17 http://www.utwente.nl/ewi/is/research/completed_projects/completed_projects/corepro.doc/

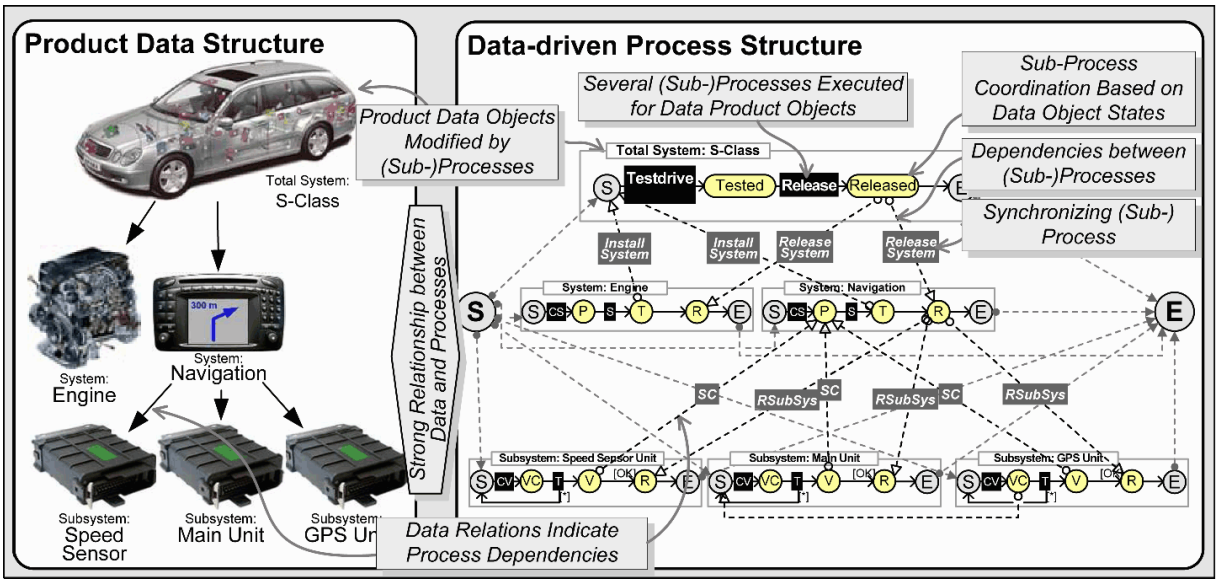

Figure 3.8: Example for the data-driven approach: a product data structure and its according data-driven process structure (adopted from Müller et al. [243])

Another contribution for the graphical notation of the run-time status of an enacted content-oriented process is available from the data-driven project. Müller provides a diagram example in [244] that distinguishes run-time status of states and of transitions by square-shaped and triangle-shaped icons, illustrated in figure 3.9. By using the run-time status icons, the resulting diagram indicates whether a particular state of a process structure has been already passed (state S1), is currently activated (state S3), has been skipped (state S2), or has not been reached yet (state S4). Respectively, the transition icons indicate whether the associated process has been started, skipped, or completed.

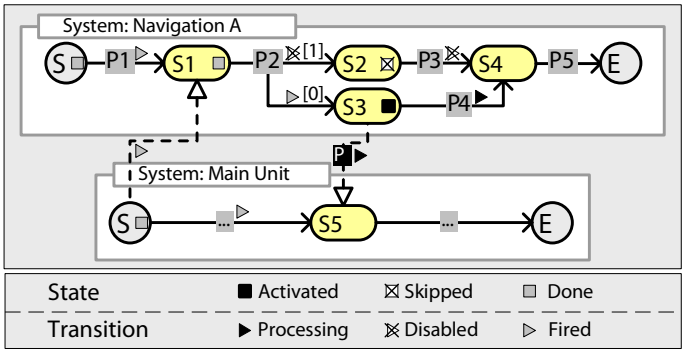

Figure 3.9: Example for the run-time status of an enacted data-driven process structure (adopted from Müller et al. [244])

In conclusion, data-driven process structures support the IPR workflow pattern by providing interleaved synchronization of sub-processes. Thus, the approach extends activity diagrams. Unfortunately, the COREPRO prototype implementation is not publicly available. Finally, the data-driven approach provides a sophisticated workflow model being specialized on hierarchical write-and-review-processes.

3.4.2 The "Resource-Driven" Approach

The *resource-driven workflow system* has been developed at Pennsylvania State University. The protagonists are Jianrui Wang and Akhil Kumar. A prototype implementation is described in [245] but it is not publicly available.

The main focus of the "resource-driven" approach is to build a workflow system entirely inside a relational database management system. Again, the approach replaces control flow dependencies with data flow dependencies in order to represent workflows. Data is generalized into resource by Wang and Kumar, who distinguish four types of resources: data resources, human resources, physical resources, and equipment resources. In their later publication [246] they replace the *data resource* type with the *document* type. The approach concentrates on the document (data resource) type. Any specifics of the other resource types remain future work.

An illustrative example is provided by a process for handling orders from customers (cf. both [245, 246]). The process is similar to the job application scenario, previous in this section, since tasks produce and consume documents. The original diagram is outlined in figure 3.10. The control flow is at the left-hand side and its equivalent resource flow is at the right-hand side. Notably, the arrow and bar symbols have completely different semantics on each side, albeit they have the same appearance.

The arrow-shapes on the control flow side have standard *sequence* semantics. The diagram on the left can be considered as a UML activity diagram. The usage of the bar-shaped symbol for OR decisions by Wang and Kumar is methodically inaccurate because it implies a fork or join node for parallel activities, yet, these are not *parallel* activities but *alternative* sequences. Using the diamond-shaped UML symbol to represent a decision or merge node would have been correct. For the right-hand side, Wang and Kumar reuse the UML symbols but informally redefine them with a resource-driven or data flow semantics. First, the rectangle-shaped symbol is introduced for data resources. Each activity has incoming and outgoing resources, thus, the flow is a strict succession of resource→activity→resource symbols. The arrows have either a *produce resource* or *consume resource* semantics, depending on whether it is an activity→resource arrow type or a resource→activity arrow type. The bar-shaped symbol implies the production

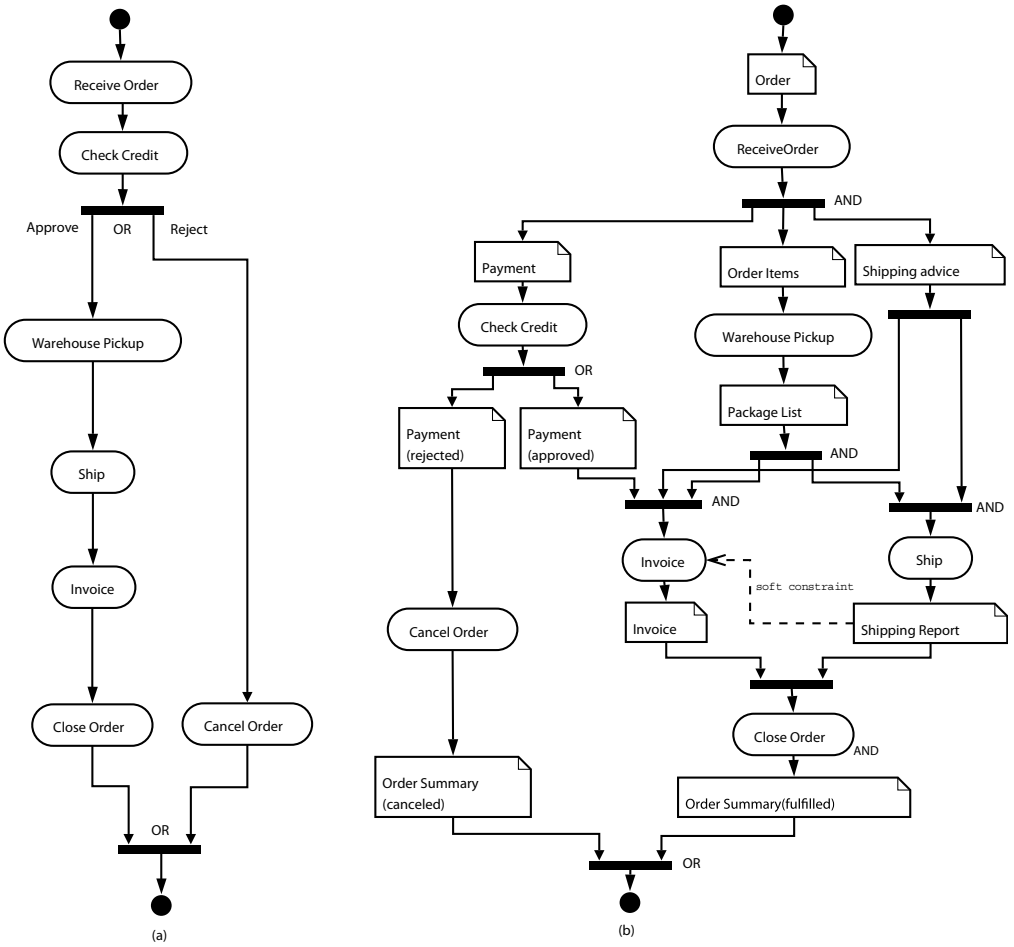

Figure 3.10: The resource-driven approach: order processing workflow with the control flow at the left-hand side and the resource flow at the right-hand side (adopted from Wang and Kumar [245])

or consumption of multiple data resources. An interesting concept is the dotted arrow that represents *soft constraints*, in contrast to the standard solid arrows that are hard (produce/consume) constraints. Soft constraints are used between tasks and resources, if no direct produce or consume relationship is present. They are considered as business rules or business policy constraints because the invoice and the shipping could be performed in any order[18] from the perspective of data dependencies.

The prototype was implemented based on a Microsoft SQL Server 2000. It is not publicly available. The prototype uses database triggers, exclusively, to implement its workflow engine. The system still relies on predefined *process definition files* that include the

18 The exemplary company wants to do the invoice always after the shipping—as it is apparent from the control flow at the left-hand side.

activities and input/output documents, in analogy to the above resource-flow diagram. The functionality of the workflow engine is to manage work-lists. The work-lists are changed on the availability of necessary document input. A key point of Wang and Kumar is to demonstrate the implementation of such a workflow engine entirely inside a database system.

In conclusion, the resource-driven approach demonstrated the application of database triggers for handling workflow events. Still the system implementation is centralized and the workflow schema is statically defined. The project appeared in 2005 but many aspects are considered future work by the authors. Research did not continue on the project. Wang completed his PhD thesis in 2009 [253]; the thesis does not mention the resource-driven approach to workflow modelling but is about discrete event simulation. Finally, the resource-driven workflow system is an early approach that considered workflows from a content-oriented perspective with the purpose to provide support for plain document-driven processes, which is missing in traditional activity-oriented workflow engines.

3.4.3 The "Artifact-Centric" Approach

The *artifact-centric business process model* has been developed at the IBM T.J. Watson Research Center in New York. The protagonist is Richard Hull. The approach has no primary implementation. Yet, for example, Bhattacharya claims in [247] that it had been applied in various internal and external IBM client engagements. Recently, the approach has become part of a larger effort at IBM Research, called Project ArtiFact™. ArtiFact™ seems to provide a primary implementation of the artifact-centric approach in the future.

The primary focus of the "artifact-centric" approach are business processes and the accompanying enterprise application integration. Again, the key idea for managing workflow is "to shift the focus of business modelling from the actions taken to the entities that are acted upon" [247, p. 3]. The focus for artifact-centric modelling is on database records used to store information pertinent to a given business context. Artefacts are required to have a unique identifier that allows identification of an artefact across the enterprise. Some IBM methodology to identify business artefacts is provided by Nigam and Caswell [254]. A graphical notation was not available until recently.

The project develops a formal "artefact-based business model" as well as a declarative semantics for artefact life-cycles, from artefact creation to its completion, based on the use of business rules. The artefact-based business model contains artefacts, services, and rules. *Business Artefacts* have attributes and an enumeration of states. *Business Services* are modelled as functions that read and write on artefact attributes and that trigger

artefact state change. *Business Rules* are used to invoke business services if a certain attribute-state constellation appears. An accordingly three-fold set of language element instances forms an *artefact system*. The artefact system is basically the equivalent to a workflow schema. The specification of the formal model was developed by Gerede and Su (cf. [255, 256]). It allows for process verification in a content-oriented fashion. For example, it allows to automatize the answering of model verification questions "exists a successful completion for an artefact?", "exists a dead-end path for an artefact?", and "exists attribute redundancy within an artefact system?" (cf. [247, 257]). Altogether, the artifact-centric model is abstract to a great extent. The human-perceivable workflow itself, in terms of a coarse-grained process intention or a process progression, is implicit and hidden behind the mutual reaction and logic interdependency of services and rules (cf. [258]).

Since its absorption into IBM project ArtiFact™, the Guard-Stage-Milestone (GSM) model has recently been introduced by Hull in [259]. It outlines a first draft on a graphical notation. An example from [259] is provided in figure 3.11 for illustrative purposes. The boxes are stages, the diamond-shaped icons are guards, the circle-shaped icons are milestones. Milestones play a primary role. They indicate the closing of a stage, indicating whether the milestone has been "achieved". Achieving a milestone will fire an accordingly named event. Events can trigger diamond-shaped guards. Each guard symbol implies the existence of a rule-based constraint on a business artefact, which is not illustrated in the figure. Simplified speaking, a stage interacts with other stages by closing its milestones. However, it is unclear if the simple dashed-orange arrows are expressive enough to visualize the relationships that result from rule-based association. Rules can be complex logically formulated predicates on an arbitrarily large set of events and states. It is an open issue how rule-based object relations can be graphically visualized in general. Currently, the GSM graphical notation is a concept and there are no editing tools. Unfortunately, neither any artifact-centric implementations nor the ArtiFact™ product is curretly publicly available.

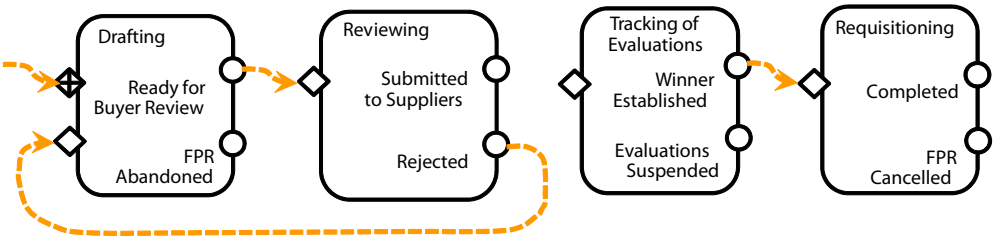

Figure 3.11: Example for the artifact-centric approach: the Guard-Stage-Milestone notation (adopted from Hull et al. [259]) *Note: no legend is provided.*

In conclusion, the artifact-centric approach appears as a mature framework for general-purpose content-oriented workflows. The distribution of the enterprise application landscape with its business services is naturally considered, yet, the workflow engine itself seems to be centralized. The process enactment seems to be tightly coupled with a technically pre-integrated database management system infrastructure. The latter makes it most suitable for manufacturing processes or for organizational processes within a well-defined institutional scope. The approach remains work in progress. Still, it is a relatively old and established project on content-oriented workflows. Funded by IBM, it has comparably high number of developers, thus, it is a promising project.

3.4.4 The "Object-Aware" Approach

The *object-aware process management* has been developed at the University of Ulm, Germany[19]. The protagonists are Manfred Reichert, who continued his "data-driven" research project, and Vera Künzle. The primary implementation is PHILharmonicFlows.

The primary process element of the "object-aware" approach is the *data object*. Each data object consists of a primary key and a set of (non-key) attributes. Attributes are optional, initially; they need not be filled out at instantiation-time. Instead, each data object type has several declared object states. Each object state defines which subset of the object's attributes must be filled out. Thus, more and more attributes become mandatory as an object instance traverses all its object states.

The research group uses a job application scenario for illustrative purposes. The object-aware process conception of an internal application review is illustrated in figure 3.12. The diagram outlines one object instance with its attributes and their values that progress over time. The light-grey boxes, above the value sets, are the corresponding object states. The dark-grey boxes are the implied activities. The activities are outlined a second time at the bottom, forming the implied activity sequence. This activity sequence is supplemented to point out the duality between the process structure perspective and the data structure perspective.

For each object type there is a sequence of states and state changing activities, forming an object-type-specific sub-process. These sub-processes are called *micro process* and figure 3.12 has illustrated such a micro process. The object-aware approach primarily considers object types that are structurally related to each other via *aggregation relationship* (cf. [260]). The example for such an aggregation within the overall scenario

19 The term that is also used by the project's researchers in their German publications is "datenorientiertes Prozess-Management".

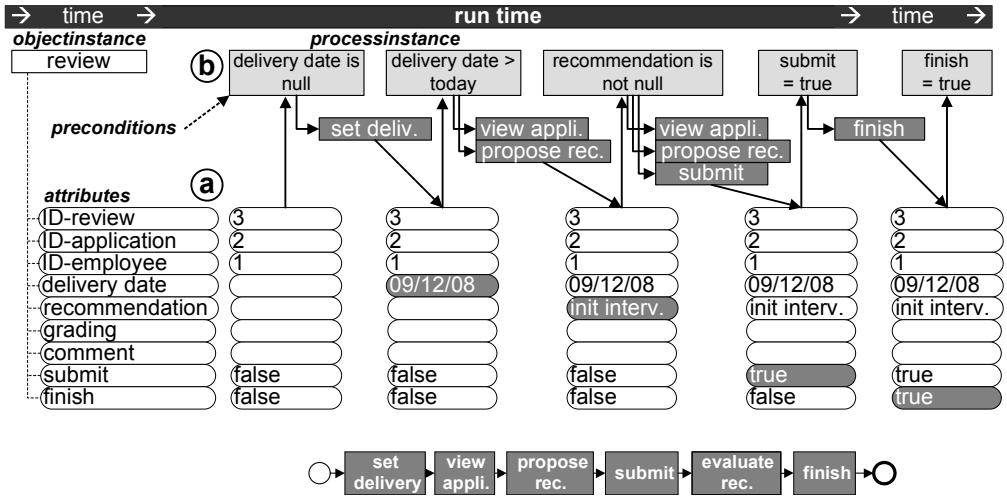

Figure 3.12: Example for the object-aware approach: process structure vs. data structure of a micro process (adopted from Künzle and Reichert [250])

is a job application being reviewed by different employees, thus, an application object aggregates multiple review objects.

In addition to the micro processes, the object-aware approach considers *macro processes*. As we have seen, micro steps strictly relate to attributes of a particular object type. In contrast, a macro step strictly refers to whole object types, each in a particular state. The macro process for the job application scenario of the object-aware approach is outlined in figure 3.13.

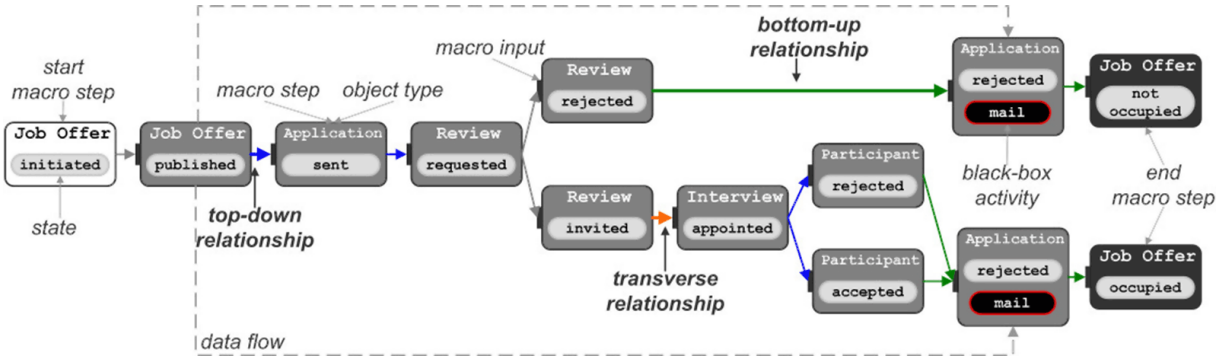

Figure 3.13: Example for the object-aware approach: macro process (adopted from Künzle and Reichert [252])

The blue *top-down relationships* and the green *bottom-up relationships* indicate transitions between object types that are first of all in an aggregation relationship. The aggregation relationship is part of the data structure perspective and the top-down/bottom-up

relationships enhance it with a process structure perspective. In fact, process coordination is mostly modelled along aggregation relations. This is due to the heritage of the object-aware approach from the data-driven approach. Still, the object-aware approach extends the data-driven model by *transverse relationships*, i.e. the orange arrow in figure 3.13. Transverse relationships are intended for arbitrary transitions between structurally unrelated objects; yet, the specifics of transverse relationships are work in progress.

Another factor that is placed emphasis on is access control [261]. The access authorization is considered as the concept for user integration into the workflow model. It is considered as a primary challenge because the tight integration of process and data necessitates that process authorization must be compliant with data authorization and vice versa. For this purpose, Künzle and Reichert analyse in [251] requirements for actor assignment and authorization.

The implementation of the object-aware approach is PHILharmonicFlows [252]. Unfortunately, it is not publicly available. The system implementation is centralized and uses a relational data model. The object type and object state schema is statically defined. PHILharmonicFlows generates both a process-oriented display and a data-oriented display. The process-oriented view is to display work-lists of micro steps for assigned users. The data-oriented view is to automatically generate basic user input forms for each data object. For actor assignment and authorization, PHILharmonicFlows manages an authorization table with a three-dimensional classification schema that holds for each combination of object attribute, object state, and user role the permission for either read or write. This access information is primarily used for generating the basic user input forms because the write flag controls whether the equivalent GUI widget field is set editable or not.

In conclusion, the object-aware approach manages a set of object types and generates forms for creating object instances. The form completion flow is controlled by transitions between object configurations each describing a progressing set of mandatory attributes. Each object configuration is named by an object state. During the data production flow the responsibilities for providing object values shifts between users and the data production flow is discrete by defining a sequence of object states.

The discussion is currently limited to a centralized system, without any workflows across different organizations. However, the approach is of great relevance to many domains. For example, Juliane Blechinger implemented a customized form-based application "DQ-Step" (e.g., [262]) for an industry partner in the energy sector, at our institute within the scope of her PhD thesis [263]. Her DQ-Step is based on the same key understanding of data and processes, as it is prevalent in the object-aware approach. Blechinger's work has not been about workflow models but about data quality in concurrent engineering. Her

work does analyse, motivate, and explain the fundamental need for such a tool platform in detail, for her non-healthcare domain. Finally, the object-aware approach and its PHILharmonicFlows system are going to provide general-purpose workflow systems for generic enactment of data production processes.

3.4.5 Résumé

Adaptiveness in Content-Oriented Workflow Approaches

Content-oriented approaches commonly rely on fixed content schemas and status triggers to drive workflow automation. They do not consider run-time adoption of content schema, life-cycle configuration, or artefact status attributes.

In the resource-driven approach by Wang, the process definition files are predefined and not subject to change. In the artifact-centric approach by Hull and IBM, there is no concept for run-time adjustment of its workflow schema. In the data-driven and object-aware approaches, i.e. COREPRO and PHILharmonicFlows, data is managed based on object types—at run-time, the number of object instances and links may vary but the types and their structure is statically defined at workflow design-time.

For supporting ad hoc processes that are enacted by knowledge workers, the process structure and participant description must be able to evolve at run-time. Furthermore, it must be possible to adjust the workflow by the human actors themselves to their emergent needs, instead of necessitating a system administrator or workflow specialist to perform any run-time changes. In conclusion, an adaptive workflow artefact and artefact attribute model needs to support demand-driven data extensions.

Missing Aspect: Circulation

None of the contemporary research approaches to content-oriented workflows considers circulations (cf. sect. 3.3.4). Approaches that analysed circulations from a workflow perspective appeared only during the 90s.

The ProMInanD system [264, 265] was implemented by Karbe et al. in 1990. ProMInanD implements Electronic Circulation Folder (ECF) for German ministerial environments. The interesting about ProMInanD is its analysis of circulation routing. It results in a rich set of circulation operations that are motivated by ministerial circulation scenarios. The set includes various re-routing operations for cases where deviations from predefined migration routes are required. The complete list of circulation operations is: *forward*, *postpone*, *inform*, *not me*, *refer back*, *append*, *delegate*, *shortcut*, *shift*, and *fetch back*—a

description of these operations is provided in [264]. The purpose of the ECFs is not only to inform the governmental departments but also to allow the various office workers to contribute content units to the circulating ECF container.

From a technological perspective, the ProMInanD system is tightly bound to a Sun workstation desktop environment and a central TransBase™ database system[20]. The database is used as migration server in order to physically move an ECF from one workstation to the next. Each workstation needs a ProMInanD desktop installation that is written in Objective-C. The prototype implementation is not publicly available. The data flow is user-shifting and it is discrete by defining a sequence of migrations for the ECF between desktop stations. The two available publications provide an informal narrative and do not provide a formal model or a design of its system architecture.

Some years later, still in the 90s, Prinz and Kolvenbach, who already provided us with the analysis of paper-based circulations in section 3.3.4, were also involved in a German government project POLITeam [266] that implemented electronic circulations and shared workspaces. The system had not been publicly available[21]. Early POLITeam publications provide reflections on its applied software development model, a *helical model* for iterative system design (cf. [266]). Another one explains legal considerations to circulations (cf. [267]). Some publications concern the pros and cons of different Graphical User Interface (GUI) notification mechanisms like background status notification vs. background animation vs. non-modal windows vs. modal windows (e.g., [268]). In [269] experiences are discussed about the control of membership of a shared workspace. A late publication also provides an analysis on the impact of the introduction of the system on the real-world processes (cf. [270]) as well as selected experiences from the overall project procedures (cf. [271]). However, the various publications do not provide insights on the POLITeam system architecture, the model representation of its workflow conception, or its circulation routings. Indicated by [272], it appears that the POLITeam system was a client-server architecture with desktop client applications that is suitable for intranet environments within the well-defined scope of an organization.

The ProMInanD and POLITeam approaches from the 90s are pioneer approaches on content-oriented cooperative office work. However, current research on content-oriented workflows has not integrated aspects of circulation workflows into its considerations. A key factor seems to be that circulation workflows are ad hoc workflows based on

20 TransBase™ is still maintained. It is a trademark of TransAction Software GmbH.

21 The POLITeam system was developed in cooperation with the VM-Gedas as an industrial partner. The authors explain that it was implemented as an extension to the commercial groupware product LinkWorks™ from Digital Equipment Corporation (DEC). However, DEC was acquired by Compaq in 1998, which subsequently merged with Hewlett-Packard in May 2002. LinkWorks™ was taken off the market.

dynamic re-routing between participants. Thus, adaptiveness is a requisite for supporting circulations. Eventually, the combination of content-oriented workflow methods with methods for adaptive system design and adaptive content models could extend the field of application to knowledge-driven ad hoc processes. The amplification with distributed workflow execution could extend the field of application to inter-institutional scenarios.

Distinguishing Characteristics of Content-Oriented Workflow Approaches

There is a plurality of content-oriented workflow approaches each with a different orientation. An empiric observation over all approaches results in five distinguishing characteristics: the element type system, element instantiation, content progression, content accessibility scope, and process assembly. The *element type system* characteristic describes whether the content units are simply typed elements like n-tuples of attributes or whether they are complex typed objects that allow for inheritance or composition. The *element instantiation* characteristic describes whether a single content unit as a model element in a workflow schema can be instantiated within a workflow instance exactly once or whether multiple element instances are supported. The *content progression* characteristic describes which type of model is used to describe the relation between content units in different states and the successive advancement towards a workflow termination. The *content accessibility scope* characteristic describes whether the access to a content unit is restricted to some kind of process context. It must not be misunderstood as security-related access control. The *process assembly* characteristic describes whether nested sub-processes are supported. Figure 3.14 illustrates these characteristics.

Two kinds of element type system can be observed. The data-driven/COREPRO, resource-driven, and artifact-centric approaches apply *simply typed* content units. They use plain state objects, possibly with an additional set of attributes. Only the object-aware/PHILharmonicFlows approach allows *complex typed* content units because it considers class-based relationships like inheritance and composition. Accordingly, two kinds of element instantiation can be observed. The same three approaches that use simply typed elements use *unique elements* within a workflow schema, i.e. they are instantiated at most once within each workflow instance. Only the object-aware/PHILharmonicFlows approach allows for *multiple instances* of a workflow schema element at run-time within the scope of a workflow instance. Notably, PHILharmonicFlows allows for class-based instantiation. Changing an object structurally requires the change of its class and is not supported at run-time.

Three kinds of content progression can be observed. The data-driven/COREPRO and resource-driven approach instrument specialized forms of UML *activity diagrams* to describe states of content units and overall workflow progression. Thus, both approaches

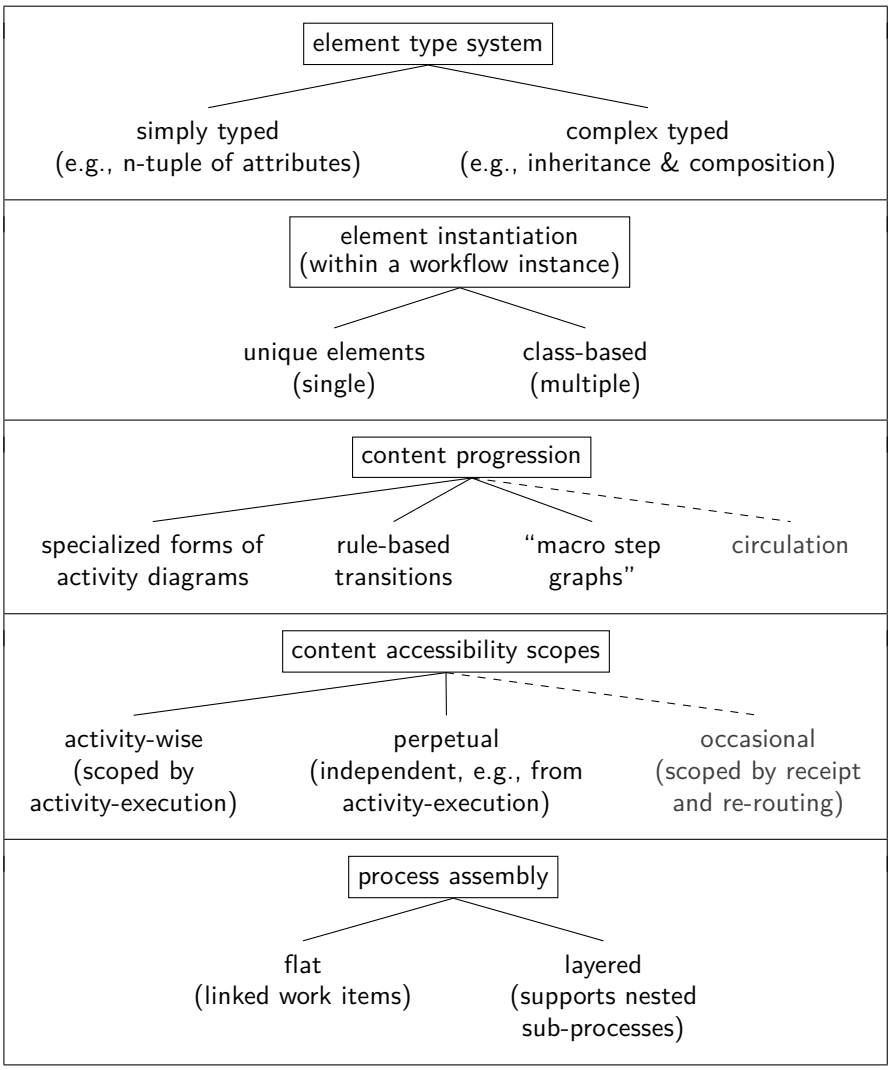

Figure 3.14: Types of characteristics for content-oriented workflow models

are inherently assisted by a graphical notation. The artifact-centric approach uses a formal *rule-based transition* between content unit states. A graphical notation of the rule base or the interrelationships between rules concerning content progression is not available. As a complement, the graphical GSM drawings are under development, however, this is still ongoing work, and the relationship to the underlying rule base is an open issue. In contrast, the object-aware/PHILharmonicFlows approach applies "*macro step graphs*" to describe workflow progression. It is assisted by an according informal boxes-and-arrows notation. In conclusion, the community of content-oriented workflow approaches has no consolidated understanding of content progression. Furthermore, there is no notation available that is both descriptive and formal, as it has been achieved by activity-oriented workflow approaches like BPMN.

In addition, a fourth kind of content progression is outlined in figure 3.14. It relates to the discontinued circulation approaches in the 90s, thus, it is outlined in grey and with a dashed line. The ProMInanD and POLITeam approaches describe content progression by *circulation* in form of dynamically changing routes between participants. Neither a formal nor a graphical notation was provided by these approaches. However, using circulation routes to describe workflow progression is a distinctive characteristic.

Two kinds of content accessibility scopes can be observed. The data-driven/COREPRO and resource-driven approaches still have a notion of activity and access to content units is only provided in the context of an activity. Thus, they allow only *activity-wise* accessibility. The artifact-centric approach supports only activity-wise access, too, because the business services are a substitute for the notion of activities and the whole approach is about business rules that specify under which circumstances business services as activities are allowed to access business artefacts. Formal analysis methods on rule sets are provided to check for mutual exclusion of activities and to prevent deadlocks. Unrestrained access at an arbitrary point in time is not supported. In contrast, the object-aware/PHILharmonicFlows approach allows access on objects independently from activity execution. Thus, it could be called *perpetual* accessibility. There are also provided means for access restriction of actors to content units only within selected activities.

Again, if circulation approaches are taken into consideration, a third kind of content accessibility scope can be observed. In a circulation approach, the content accessibility is bounded similarly to activity-wise access. However, the scope is the receipt and the self-decided re-routing. In contrast to the other forms, this kind could be described as *occasional* accessibility.

Finally, two kinds of process assembly can be observed. The characteristic distinguishes whether an approach supports nested sub-processes or not. The ones without such support link work items in a *flat* graph structure, whereas the former ones provide *layered* process assembly with multiple nested graphs. The resource-driven and artifact-centric approaches do not support nested sub-processes. The data-driven/COREPRO and the object-aware/PHILharmonicFlows approach support both flat structures as well as nested sub-processes.

An integrated view on the classification of each approach is illustrated in table 3.2. The characteristics are listed in abbreviated form. Both circulation approaches are special because none of the other approaches considers circulations or cites any according publications. Thus, both approaches to circulation, and the characteristics that have been derived solely from them, are again coloured in grey.

		approaches					
		"data-driven"	"resource-driven"	"artifact-centric"	"object-aware"	ProMInanD	POLITeam
element type system	simply typed	X	X	X	X	X	X
	complex types				X		
element instantiation	unique elements	X	X	X	X	X	X
	class-based				X		
content progression	activity diagrams	X	X				
	rule-based			X			
	macro step graphs				X		
	circulation					X	X
content accessibility scope	activity-wise	X	X	X	X	X	X
	perpetual				X		
	occasional					X	X
process assembly	flat	X	X	X	X	X	X
	layered	X			X		

Table 3.2: Classification of content-oriented workflow approaches

A final aspect about each research approach is whether it is discontinued. But this characteristic is not about the workflow model, thus, it is not part of the taxonomy or the comparative table. Both circulation approaches are discontinued, as are the data-driven and the resource-driven approaches. The artifact-centric and the object-aware approach are research in progress. Moreover, they are the most promising concepts with the highest degree of maturity.

Conclusion on Content-Oriented Workflow Approaches

The field of content-oriented workflow models is relatively young. The intended application differs and the terminology is quite heterogeneous. The artifact-centric approach targets enterprise application environments. In a nutshell, rules monitor artefact states and trigger system function calls. Workflow automation is important, thus, a formal declarative semantics has been devised. A graphical notation for the workflow was not part of the initial considerations but is currently work in progress.

The object-aware approach supports user-centric data production processes. The data structure becomes the primary workflow element and each activity is subordinate to its

data object. At run-time, several human actors fill-out data forms to progressively fulfil the data requirements. Workflow automation, by delegating tasks to system functions calls, is currently not part of the considerations. Both approaches provide general-purpose concepts for generic enactment of content-oriented processes.

The artifact-centric and object-aware concepts provide considerable achievements on workflow models. The content-oriented perspective is a major complement to the traditional activity-oriented perspective. The activity-oriented approaches have provided a paragon for fine-grained workflow control, for decades. Thus, the content-oriented approaches are engaged to achieve similarly sophisticated model elements. End users and workflow designers are confronted with novel, if not to say eccentric, tools and methods for the content-oriented paradigm. Users would require training on the rich models and their subtle implications. However, this is necessary as well for intricate activity-oriented models like BPMN.

In conclusion, the available content-oriented workflow approaches are suitable for concise and in-depth modelling of business processes, office processes, or manufacturing processes, within a well-defined institutional scope. Yet, the process enactment remains tightly coupled with a technically pre-integrated application system infrastructure or database system. The content-oriented workflow engines themselves remain centralized. Workflow schema evolution, ad hoc processes, and distributed workflow execution are open issues.

3.5 Active Document Approaches

The term “active document” is an umbrella term for several approaches that allow for active properties on digital documents. The approaches are grouped into four categories. The first category is file system. The infrastructure to add active properties is tightly integrated into the operating system environment. The second category is windowing system. The infrastructure is limited to the local desktop environment and concerns only embedding and linking to documents. There is only one according approach from Microsoft; still, it achieves a sophisticated approach to a graphical and document-oriented human-machine-interaction that fulfils the active document metaphor. The third category uses web browsers as execution environment. Again, there is only one according approach, TiddlyWiki. It implements a genuinely self-contained electronic document that embeds its own content editor. The fourth category is about component-based active documents. Traditional component infrastructures are extended by document-oriented considerations. The according infrastructures depend on the installation of component run-time containers, thus, these approaches are suitable for distributed but well-defined institutional scopes. The following list provides an overview of the approaches. The

preceding number icon in the list indicates the category. Each approach is listed with representative publications:

① Placeless documents [178, 179, 273]
① AppleScript Folder Actions [274]

② Microsoft Active Document Containment [275–278]

③ TiddlyWiki [279]

④ Ercatons [280, 281]
④ Active XML [282, 283]

The concept of active documents must be understood independently of workflow considerations. A general definition of active documents should not foreclose any type of application. Thus, the active document definition in this thesis (cf. sect. 2.2.12) has been conditioned by the document's self-containment of its actions but has been non-descript about the operational purpose of its active properties. Some of the approaches allow only for editing. Most approaches allow for general-purpose logic[22] in active properties. Embedded editing capabilities are relevant for inter-institutional system integration. Capabilities for general-purpose logic are relevant for workflow enactment. As it has been done for content-oriented workflow approaches, at the end of the section, further analysis will be conducted that results in a taxonomy of distinguishing characteristics to classify active document approaches.

3.5.1 File System

Two approaches integrate active properties facilities within the file system infrastructure. First, the Placeless documents approach for that the idea of an active document was first discussed. The technological background about the Placeless documents has already been explained in sect 2.2.12. This section, in addition, will explain an approach by LaMarca, who was part of the Placeless documents project, to use active properties for implementing basic content-oriented workflows. The second approach that integrates active properties into the file system infrastructure is Apple's approach of AppleScript Folder Actions for Mac OS X. Both approaches are suited only in a well-defined institutional scope because the infrastructure is tightly coupled to an explicitly administrated and uniform operating system environment.

22 General-purpose logic means that support for a Turing-complete [284] scripting language or programming language is integrated and according code fragments can be embedded as active properties.

Placeless Documents

The Placeless documents project from Xerox PARC provides an abstraction from document- and file-management interfaces [178, 179]. The Placeless documents system primarily implements a Network File System (NFS) server, providing access to stored documents through the standard NFS remote file access protocol. In addition, Placeless also provides remote HTTP-based file access via the Web-based Distributed Authoring and Versioning (WebDAV) protocol. The infrastructure includes facilities to implement active properties for arbitrary documents. The Placeless documents infrastructure was a middleware that required technical installation with comprehensive administrative privileges. Figure 3.15 provides an outline from [178] on the Placeless middleware. The implementation is not available publicly.

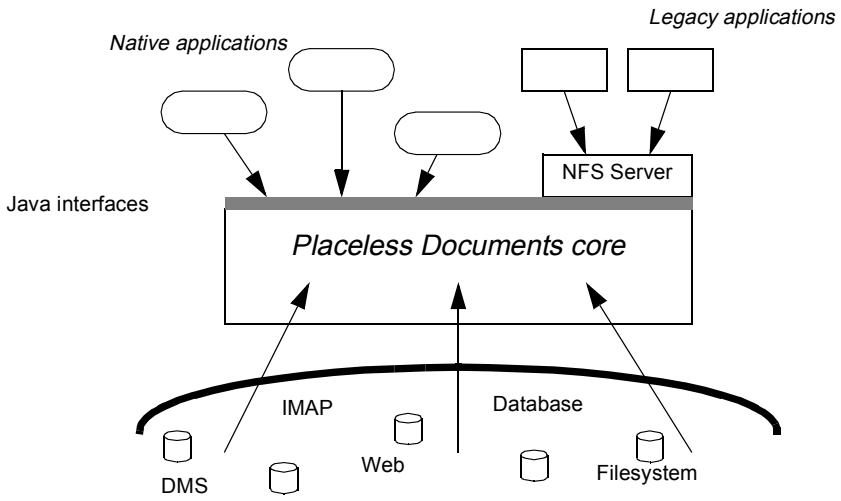

Figure 3.15: Outline on the Placeless middleware (adopted from Dourish et al. [178])

LaMarca et al. provide an exposition of the Placeless system aspects and how they can be applied to the construction of document-based interaction (cf. [273]). They thereby provide a proof-of-concept that the facilities of Placeless can be applied to the domain of workflows. LaMarca outlines two scenarios, travel approval and hiring. The hiring scenario is similar to the job application scenario that was described for content-oriented workflows (cf. sect. 3.3, p. 111). The travel approval scenario explains which influence Placeless documents could have to content-oriented workflows.

In the travel approval scenario, users can construct itineraries any way they wish and are free to choose the application and document format of their choice. This is considered as a key feature by the authors and differs significantly from traditional workflow systems where relevant data must be manipulated with vendor-provided tools or pre-integrated applications. For user interaction, a trip status document (TSD) serves as a drop target

for new trip itineraries. Once an itinerary has been dragged onto the TSD the approval process is initiated. Thus, the TSD manages a list of documents that need approving, each document with an according file reference. The TSD implements this by an active property that is triggered upon file drag-and-drop. The TSD also adds an active property to the original itinerary file: the *ApproveOrDenyProperty*. This is because the actual approval or denial of a trip shall be performed on the itinerary document itself. When a manager opens a travel itinerary that requires his or her vote they view the document as usual but something else happens as well: they are presented with a Yes/No voting box, created by an active property, which allows them to decide to approve or deny the trip. The *ApproveOrDenyProperty* is what managers interact with when casting their votes on a trip. This property can determine if the user who is currently viewing the document (it is attached to) is a manager whose decision is needed for this particular travel request. When appropriate, the property can create and display a GUI component with a Yes/No button for voting.

LaMarca's description is primarily a concept paper. It provides a narrative on the user story. Screenshots of a prototype's GUI exist (cf. fig. A.6 in the appendix) but the screenshots provide no insight on the system design or implementation. It is unclear how the managers are informed that they need to approve a travel. The suggestion is that any notifications to users are done via e-mail. It is unclear how the *ApproveOrDenyProperty* determines whether the currently viewing user is a manager. The suggestion is that group management facilities of the underlying file system abstraction are instrumented but the interface is not available. It is unclear how the GUI widgets are specified and in which way the presentation frameworks or libraries are technically coupled to the Placeless middleware. It is unclear how active property implementations could be used in different organizational environments. The characterized implementation seems to be custom-made for a dedicated institutional environment. Finally, LaMarca does not describe a workflow engine but describes two active properties and a novel way to conceptualize document-based workflows. He demonstrates that the Placeless documents middleware provides a file system infrastructure that allows for according custom-made implementations. LaMarca's ideas are a precursor and his description provides an admirably simplistic aesthetics.

AppleScript Folder Actions

Email client applications have used rule-based sorting for a long time. It first appeared as *Virtual Folders* for an Emacs-based mail reader, named *VM*, in 1991 (cf. [285]). Today, it is supported by GNOME Evolution (since 2000), Opera M2 (since 2003), Microsoft Outlook (since Version 2003), Web-based Google Mail (since 2004), Mozilla Thunderbird (since 2005), and many other mail applications. Virtual folders allow setting up rules

and custom folders for the sake of sorting messages when they come in. Instead of scanning the multitude of messages in the general in-box, one creates a folder for certain newsletters or certain originators or, for example, a virtual folder could also contain all mails with an attachment. Another common usage is to delete mails older than one year (i.e. "spring cleaning"). Apple's Mail application provides rule-based sorting since 2005 and calls it *Smart Mailboxes* (cf. [286]). These Smart Mailboxes become a precursor to an active document approach by Apple for the file system in Mac OS X.

Virtual Folders have also been applied for file systems where they can let you save a search, to reuse in the future. These folders store a passive search criterion as folder property. These properties do not themselves contain logic but the criterion becomes a parameter to an underlying search engine within the operating system. The virtual folders for file systems are dynamically updated by the operating system to contain content that match its embedded criterion. This concept was first implemented by Be Operating System (BeOS) in 1998 and, as I have outlined in section 2.2.12, the BeOS concepts can be considered as a precursor to the Placeless documents ideas. Later, in 2005, the basic virtual folder functionality was integrated by Apple into the file system of Mac OS X. Apple calls it *Smart Folders* and its functionality is implemented by the Apple Spotlight engine. For the sake of completeness, it should be mentioned that the same functionality has been integrated by Microsoft into Windows Vista since 2006 as *Search Folders*. Apple Smart Folders and Microsoft Search Folders should still be considered as passive artefacts because all active parts belong to the OS environment.

In 2008, Apple extended this concept by introducing *AppleScript Folder Actions* (cf. [287]). "Folder Actions" technologically allow for associating file system folders with AppleScript scripts (cf. fig. 3.16). Apple itself never used the terminology of active documents; still, it fulfils the metaphor. There are thirteen pre-installed scripts for folder actions. For illustrative purposes, I describe some of them:

- "Add: new item alert": A pop-up window appears when an item has been added to the folder. This script is, for example, useful for shared folders to let you know about additional content.
- "Close: close sub-folders": Whenever an item is added to the folder, the windows for the folder and all its subfolders close automatically. This script is designed for screen clean-up and can be used, for example, for archiving folders—when dropping items there is the one and only relevant user action.
- "Convert: PostScript to PDF": Whenever a PostScript file is added to the folder, this script converts it into a PDF file.
- "Image: Duplicate as JPEG": Whenever an image file is added to the folder, a conversion into the JPEG format is automatically applied.

Figure 3.16: Attaching AppleScript Folder Actions in Mac OS X by an end-user to an arbitrary folder (screenshot by Goldstein in [274, p. 223])

Clearly, these pre-installed scripts are not sophisticated but basic additions to a user's automation arsenal. However, once a user has a script in place, the simple act of dragging a file from one folder to another causes the actions specified by the script to occur. Cohon describes the usage of folder actions in a tutorial [288]: key benefits can be gained in cases where a user has multiple files to be acted upon. In principle, it is possible to apply any AppleScript scripts to a folder. However, AppleScript is meant to provide end-user scripting in an application-neutral way (cf. [289]). It is not a programming language and relies on the built-in functionality of other applications to handle complex tasks. Still, these scripts represent active properties of a folder artefact. Thus, the folder itself becomes an active document. To my knowledge, there is no approach that applies this infrastructure to implement coordination for content-oriented workflows across multiple desktop stations or between several users. However, the AppleScript Folder Actions as a technological platform is an occurrence of the active document metaphor in a mainstream software environment.

3.5.2 Windowing System

A windowing system provides a GUI framework for a desktop environment. The windowing system provides access on graphics hardware, support for keyboards, and support

for pointing devices such as computer mice. In Unix-like environments, the windowing system (e.g., X Window System) supports the implementation of various window managers (e.g., GNOME Mutter or KDE KWin). In a Microsoft Windows or Apple Mac OS environment, the windowing system and window manager is integrated into the operating system and is largely non-replaceable.

In Microsoft Windows, the window manager is tightly coupled with the kernel's graphical subsystems, which allows for sophisticated concepts of GUI integration. Already in 1990, Microsoft invented the well-known Object Linking and Embedding (OLE) technology. The benefit of OLE is to edit different kinds of data objects within a document via external applications without leaving the application of the compound document. Microsoft distinguishes "embedded" and "linked": embedded objects are physically stored within the compound document; linked objects are physically stored separately and referenced by a file system locator. Embedded or linked objects are, for example, charts, tables, or drawings that are included into a Word document or PowerPoint presentation. For both embedded and linked objects, the windowing frames of the external applications are contained within the windowing frames of the compound application. Thus, OLE achieves GUI integration that follows content aggregation. OLE is not limited to Microsoft products but has been accepted by other vendors for products based on the Microsoft platform: for example by Computer-Aided Design applications, like Autodesk AutoCAD, or by business reporting applications, like SAP Crystal Reports. As an extension to the OLE framework, Microsoft has developed a notion of active documents.

Microsoft: Active Document Containment

A Microsoft Active Document always requires an environment in which to work. This environment that hosts Active Documents is an application called Active Document Container (cf. [275]). The container provides a way to create a compound document made up of different documents like Excel, PowerPoint, or Word files (cf. fig. 3.17). The original example of such a container was the Microsoft Office Binder, which was discontinued after Office 2000. Another prominent Active Document Container is the Microsoft Internet Explorer. Together, the Active Document Container application and the Active Document give users the appearance of a single, homogeneous application.

The three most prominent extensions of Active Document containments in comparison to OLE containments are described since 1998 by the MSDN library (cf. [291]):

1. "Active Documents may be multi-page [...]."
2. "[Active] Documents can be viewed full frame, in the entire client window."
3. "[Active] Documents are always in-place active."

In regard to the first statement: OLE does not natively provide embedding of multi-page objects. If multi-page objects are OLE-embedded, only the first page can be accessed. In regard to the second statement: embedded OLE objects are always displayed as a small rectangle surrounded by a hatched border. In contrast, the Active Document is the only embedded document from the perspective of the Active Document Container application and it occupies the entire container application window. In regard to the third statement: for OLE there is a strict and perceptible separation of the menu bars between the container and the contained application. In contrast, Active Document in-place activation supports menu merging: a menu bar is provided to the user that comprises menu elements from both the container application's and the integrated application's menu bars in form of a composite menu. Menu merging is intended to provide an even more seamless GUI integration.

From a technical perspective, Active Documents extend the compound document technology of OLE by providing an additional set of interfaces (cf. [276]), with IOleDocument as the essential one. In the end, there are several other interfaces involved; Williams and Bennett provide a programming reference in [292, chap. 25 & 26]. The Active Document is not the contained Excel, PowerPoint, or Word itself. Instead, the Active Document Container embeds the registered office applications into programming objects that are thereby hosted by the container. Naturally, this is only possible if an editing application complies with and implements the set of interfaces for OLE Active Document containment. By constructing these objects from the registered applications, the container generates Active Documents instances at run-time each with a compound set of passive documents at its core. In a nutshell, the programming interfaces manage views such that the contained applications are displayed in a common frame that uniformly functions

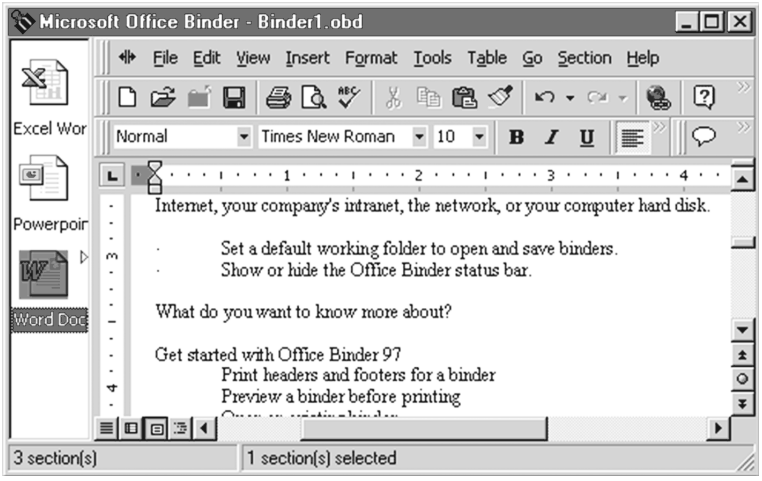

Figure 3.17: The Microsoft Office Binder as container application for a Microsoft Active Document (screenshot adopted from [290])

within a container, yet, the contained applications remain in control over their display functions. Besides, an Active Document uses its container file as its storage mechanism. On that account, it must implement the IPersistStorage interface. Another important issue that is mentioned in [276] is printing. Active Documents can provide support of *programmatic printing* (cf. [278]) by optionally implementing an IPrint interface with multi-page capabilities. Williams and Bennett relate that the concept of programmatic printing was new to the OLE architecture for Active Documents [292, sect. 25.2].

The Microsoft Active Documents approach fulfils the definition of the active document metaphor in this thesis. Indeed, any active properties are provided externally by the container applications and various installed editor applications. Thus, the active properties cannot be copied by copying only the compound document. However, the Microsoft Active Documents still provides an adequate abstraction from the relationship between the document files and its applications. The user does not himself trigger the opening of a content-specific editing application that then opens a file. Instead, the Office Binder or Internet Explorer acts as a unified and content-agnostic windowing infrastructure. Thus, the content-agnostic opening can be understood such that the user opens the compound document directly, to a certain extent, without immediately referencing a content-specific application. The compound document, virtually itself, supplies the user subsequently with available editor applications for its embedded content files.

In conclusion, the Microsoft infrastructure for handling Active Documents is not intended for interactions between several users nor does it imply any form of data synchronization between multiple desktop stations. To my knowledge, there is no approach that applies this infrastructure to implement content-oriented workflows. Over and above, using OLE objects or Microsoft Active Document containments limits the interoperability because these objects are not widely supported in programs for viewing or editing files beyond the scope of the Microsoft platform. However, the Microsoft approach is, besides Apple's Folder Actions, the second and final occurrence of the active document metaphor in a mainstream software environment.

3.5.3 Web Browsers as Execution Environment

One Personal Information Management (PIM) approach is implemented in form of an active document. It concerns Wikis, like the well-known Wikipedia. Wikis were invented by Ward Cunningham. They are implicitly web-based and instrument web browsers as text editors. Wikis were not intended as Personal Information Managers (PIM tools).

PIM is defined by Jones in [293]. PIM is concerned with how people store, remember, retrieve, interrelate, and maintain information for their daily work. Mobile hardware devices known as Personal Digital Assistants (PDAs), like the well-known Palm Pilot™ and

BlackBerry™, had the greatest impact on PIM. They provide a mobile hardware-software-platform that functions as a personal organizer. The according software applications are known as PIM tools and they organize, for example, address books, calendar, and personal notes. Today, mobile phones with the ability to install and execute general-purpose applications, commonly known as SmartPhones, support PIM tools. SmartPhones of the latest generation commonly contain pre-installed PIM tools, thus, they dominate the PDA market, today. Usually the mobile device is supplemented with additional PIM tools for the user's PC desktop environments. For business environments, PIM tool integration with groupware installations is imperative. Amongst others, PIM is also related to personal time management, for example, David Allen's method Getting Things Done (GTD), which is described in his well-known book [294]. Notably, PIM and methods like GTD are related to personal workflows (e.g., [295]). Personal workflows share, for example, the knowledge-driven characteristic and the trait of ad hoc decisions about next activities with cooperative case-based workflows. In conclusion, the field of PIM is very broad.

As said initially, Wikis were not intended as PIM tools. They were invented for collaborative authoring and not for taking personal notes. It is a key concept to a Wiki that it invites all users to edit any page without reservation. They provide a simplified mark-up language for end-users to write entries. Today, there are many Wiki implementations. Most implementations require a database system and web server installation.

In principle, a personally installed Wiki would be well suited as a PIM tool for personal notes because Wiki implementations can naturally store, retrieve, and interrelate entries. The critical condition for Wiki technology to become PIM technology is the ease with that a personal Wiki installation becomes available to individual end-users. This is where the active document metaphor becomes the key concept to achieve PIM via Wiki.

TiddlyWiki

TiddlyWiki is an open-source development[23] by Jeremy Ruston. The TiddlyWiki application is a single Hypertext Markup Language (HTML) web page that combines the interactive user interface and the storage mechanism in a single file. Thus, the TiddlyWiki is installation-free and it is not hosted on the internet but stored locally. The interactive Wiki interface is implemented by embedded JavaScript. The interface allows for creating new Wiki entries, which dynamically adds HTML elements into the file's own

23 http://www.tiddlywiki.com/

Document Object Model (DOM)[24]. A TiddlyWiki is self-modifying and self-contained. Handling a TiddlyWiki has admirably simplistic aesthetics.

The TiddlyWiki storage mechanism is to write its own HTML file back to the file system. Herein lies its single drawback, presently. All mainstream browsers render and execute HTML pages in a sandbox for security reasons and it is not allowed for the HTML page and its JavaScript to access and write local files not even its own. Thus, TiddlyWiki itself is indeed installation-free but still requires a reconfiguration of the installed browser to allow TiddlyWiki to perform its self-storage. That configuration depends on the locally used browser and on the used operating system. The adjustments are well documented and are not overly complex; still, they require technical skills. This self-storage problem is not necessarily a flaw of TiddlyWiki but stems from the fact that browsers are not yet cognisant for such kind of active document approaches.

TiddlyWiki is quite unique. However, a significant online community provides several extensions. Since PIM is related to personal time management, some extensions should be mentioned that integrate the GTD conventions into TiddlyWiki: GTDTiddlyWiki Plus[25], d3[26], or mGSD[27]. In a nutshell, these extend the TiddlyWiki such that multiple entries can be set into the context of a project name and they provide additional calendar and reminder capabilities. The important observation about the GTD extensions is that the TiddlyWiki approach, in general, is capable of implementing workflow support. By means of JavaScript programming, it ultimately allows for general-purpose logic.

In conclusion, from the perspective of the active documents metaphor, TiddlyWiki is probably the most genuine approach. In contrast to Microsoft Active Document Containment it actually embeds its own web-based editor application and uses web browsers as a cross-platform execution environment. TiddlyWikis are portable. Copying or moving a TiddlyWiki file to another user of a team is easily possible. Thus, it can provide ad hoc support to some extent for content creation in cooperative environments, like circulation scenarios. However, synchronization between two replicated files is not provided. Integration with other applications is not supported. Thus, there must be end-user acceptance for authoring documents by means of a Wiki mark-up language instead of using accustomed word processor applications, like Word or OpenOffice.

24 The DOM is a cross-platform and language-independent representation of HTML. DOM technology provides unified facilities to interact with the HTML structure and content.

25 `http://www.checkettsweb.com/tw/gtd_tiddlywiki.htm`

26 `http://www.dcubed.ca/`

27 `http://mgsd.tiddlyspot.com/` (formerly known as MonkeyGTD)

3.5.4 Component-Based Active Documents

A software component is not (necessarily) an active document. Obviously, a traditional software component encapsulates executable logic that qualifies as active properties. However, a software component does not allow for direct interaction but commonly provides only a programmable access. In recent years, the various "on Rails" approaches, like the original Ruby on Rails (e.g., [296]), have implemented the naked object method [297] to allow for Rapid Application Development (RAD). Pawson defined the naked object method originally in this PhD thesis [298]. It is a design method in that the user interface is a direct representation of the domain objects and the interface is completely and automatically generated from the domain objects. The Ruby on Rails framework implemented this type of GUI generation. For Rails it is called *scaffolding* and it is based on web technology, i.e. the scaffolding generates web controllers as well as views that provide HTML and XML output formats. Combining the naked objects concept with a component model provides means for direct interaction such that a component comes one step closer to an active document.

Furthermore, the data model for the encapsulated data of a component is commonly based on the notion of class attributes from the underlying programming language. If the component model is bound to a class-based object-oriented programming language, which is the case for all mainstream component models, a run-time adaptation of the data structure is not possible. Thus, the data model of mainstream component models is not equivalent to an electronic document. Besides, software components not necessarily support serialization of their encapsulated data or their run-time execution state. There are few component models that intrinsically support the persistence for their components, like Enterprise JavaBean (EJB). Commonly, a persistence framework must be explicitly added to the component frameworks—several object/relational mapping frameworks exist for this purpose. Ideally, a persistence mechanism should not only be bound to a relational database system but should allow component migration, i.e. to copy/move a component instance to another run-time environment. Such relocation facilities for component-based frameworks is subject to research (e.g., [299]) and leads into the domain of agent technology ("mobile agents", e.g., [300]). Component migration is not readily available in the mainstream frameworks. Thus, the options to hand over a component between users at different sites will be limited for active document approaches that are built upon component-based middleware.

In conclusion, if a component model is extended by direct human interaction facilities, for example in form of a naked object approach, and if it integrates a built-in serialization for component content states, then it can be considered as an adequate model for active documents by components. In the long term, distributed component models could become a perfect match for the active document metaphor.

Ercatons

Ercatons have been invented by Imbusch, Langhammer, and von Walter [280, 281]. The Ercaton approach provides a component model that combines prototype-based object-oriented programming with an adaptive XML-based and document-oriented data model for its components. The Ercaton specification is independent of a programming language. The reference implementation of the Ercaton component model is the ercatoJ[28] run-time container. It is implemented by the Living Pages Research GmbH, founded by Langhammer. Langhammer holds a PhD in physics, which had some influence on the Ercaton conceptions. Thus, the name "Ercaton" stems from "Mercato" and "Ion" and literally means "elementary market particle" (cf. [281]).

Ercatons provide both a programmatic access as well as a direct human interaction. The naked object method is implemented for Ercatons such that its arbitrary XML structures can be viewed and edited as HTML forms. An Ercaton is a component that is primarily a single XML artefact. Each Ercaton is identified by an explicit String-based ID that is contained within its XML. Furthermore, the Ercaton's XML contains both a part for its content data that it encapsulates as a component as well as a part in which the XML declares the Ercaton's methods, i.e. *actions*, and in which it binds a programming language artefact to an action as the according implementation. Thus, the Ercaton is both an XML document and an active component.

Changes to the Ercaton are applied at run-time, both on the content part or on the actions part. There are no transient attributes and every change to an Ercaton's state reflects on its XML. Changes to an Ercaton can be triggered via its naked object web interface or via its actions. Actions can self-modify its content similarly to changing class attributes. A sophisticated storage mechanism continuously persists the Ercaton and its execution state after each applied action or user change. A key aspect to Ercatons is that "there is no intrinsic difference between a user's and a programmer's view" [281]. The implicit integration of user interface and persistence (aka full-stack integration) is one key aspect to Ercatons' concept of *thing-orientation* [280]. Figure 3.18 from [281] illustrates the full-stack characteristics of Ercatons. Besides the HTML-based user interface, there is the Ercato Shell (ESH) that provides a command-line tool for accessing the XML representation as well as for executing actions. The identification of Ercatons as well as the representation of actions and their parameters is based upon a so called *XReference* specification[29].

28 `http://www.living-pages.de/de/products/ercatoj/`
29 The XReference schema is is outlined in the appendix sect. A.4.2 as listing A.2

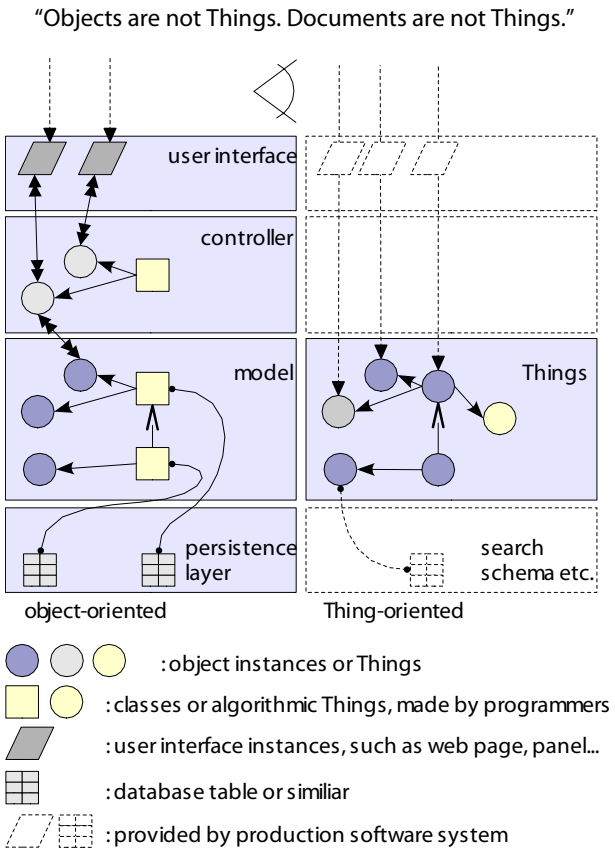

Figure 3.18: The full-stack characteristics of Ercatons (being abstracted as *Things*): implicit integration of user interfaces for direct interaction and persistence (adopted from Imbusch et al. [281])

Ercatons are instance-based. Inheritance is provided by concepts from prototype-based programming (cf. sect. 2.2.7). The prototype-based cloning is done via the XML documents. In order to inherit content and actions from a clonebase, the XML files are merged. For that purpose, an XML algebra has been implemented to add and subtract arbitrary XML structures. Ercaton prototype-based inheritance is defined by its XML algebra (cf. [280]). The implementation of this XML algebra is XOperator, which is also available stand-alone and free of charge[30].

The programming philosophy of Ercatons is described as *organic programming* [281]: "[Organic programming] tries to merge the successful paradigms of the two regimes (objects and documents) into a new programming model which is more appropriate for large-scale projects than traditional OOP methods". Figure 3.19, which is adopted from

30 http://www.living-pages.de/de/projects/xop/

[281], illustrates the statement and the relation between both antipodes. The Ercaton approach intends to allow for emergent evolution of a software system.

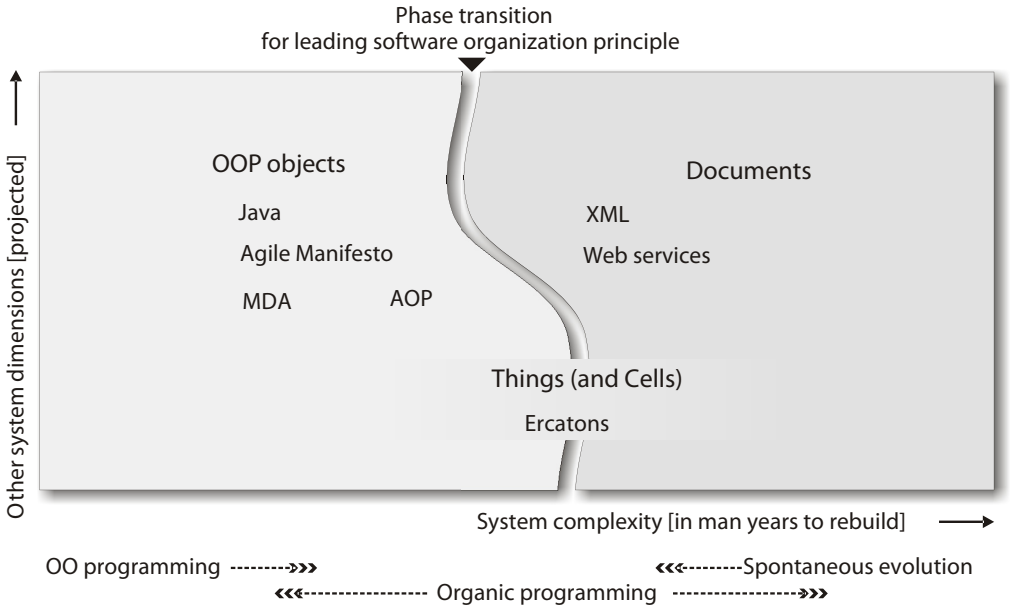

Figure 3.19: Ercatons as organic programming: merging two paradigms, objects and documents (adopted from Imbusch et al. [281])

In a student research project with Manuela Schinn [301], we evaluated Ercatons and organic programming. I was in personal contact with Falk Langhammer who provided us with an ercatoJ installation. For evaluating the various aspects of Ercaton-based programming, a pilot project implemented a web portal for managing medical guidelines. As a reference, the web portal of the German "Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften" (AWMF) was used[31]. AWMF is the German authority for publicly providing the medical guidelines of the various committees. The information being managed by the portal is primarily the meta-information about the guidelines that can be used for indexing, searching, and browsing—the actual guidelines are linked and are made available as PDF documents. The public meta-information at that time had been semi-structured: it was partially divergent seemingly in correlation to different "flavours" of diverse committees. It would have been hard to model a comprehensive data schema to satisfy all occurring peculiarities. Ercatons coped well with the real-world hotchpotch by allowing for instance-based deviations. The semi-structured Ercatons can still be indexed, searched, and browsed. The prototype-based concepts allow for schema-

31 We used AWMF portal at that time, i.e. in 2009. In the meantime it was modernized and the provided information has been restructured. The former portal is documented in Schinn's bachelor thesis [301]. The current AWMF portal for German medical guidelines is available at: `http://www.awmf.org/leitlinien/`

tizing various types of guideline records after accepting their data into the system. Thus, it achieves "pay-as-you-go" or "Data First, Structure Later. Maybe." and allows for a demand-driven method (cf. sect. 1.2.1). From our experience, the Ercaton technology is mature and keeps its promises. Ercatons as a organic programming language and full-stack frameworks can compete with object-oriented programming and mainstream component frameworks.

In conclusion, I earnestly considered implementing my active documents for case-oriented collaboration on the basis of Ercatons. Unfortunately, the ercatoJ engine is not (yet) publicly available. Even if the Ercatons engine is used in various industrial projects, the current installations use a single central ercatoJ engine. Remote access is provided, thus, it integrates in a networked system environment within a well-defined institutional scope. However, component migration and synchronization between remote "Ercaton universes" is currently not supported. Deploying an ercatoJ container at different sites is not more complex than deploying a JEE server or a Ruby on Rails server, however, all of them still require profound technical expertise and administrative rights. Finally, I decided to aim for an installation-free approach in order to achieve genuine ad hoc support. Thus, it is necessary that the active document itself contains anything for execution, independently of a pre-installed component run-time container. However, the Ercatons environment, in contrast to *α-Flow*, is a full-fledged programming environment. Finally, Ercatons is a promising approach and organic programming is a perfect match for evolutionary software systems.

Active XML

The Active XML (AXML) approach has been developed by Serge Abiteboul et al. [282, 283] at the Institut National de Recherche en Informatique et en Automatique (INRIA) near Paris, France. AXML is based on the idea of embedding calls to Web Services inside XML documents. If the Simple Object Access Protocol (SOAP)-based Web Service invocation is executed, the invocation result, in XML format, dynamically enriches the AXML content. Thus, the AXML is like a partially materialized view, which is capable to provide a combined representation of static XML data and dynamically obtained data from its embedded SOAP calls.

The AXML contains parameter that configure when and how often its embedded calls are activated, i.e. in which intervals the materialized view is refreshed. Two reserved XML attributes are used: frequency and mode. The *frequency* can either be a periodic interval, like daily or weekly, or a dedicated point in date and time. The *mode* controls whether the invocation is immediately executed if the frequency-based timer expires or if the invocation is deferred until its AXML content representation is accessed.

The materialization is quite sophisticated. The AXML retains both the Web Service call declarations in parallel to the materialized results. For each call declaration, it can be configured whether the next call result *replaces* the previous result, whether the results are *appended* to each other, or whether the respective XML elements sets are merged. The XML merging is not based on an XML algebra like the Ercaton XOperator. Instead, the XML IDs are used as keys of a map, thus, the element set is merged, thus, elements with the same ID are replaced.

The AXMLs can interact with each other by exchanging AXML data and not only materialized data. Figure 3.20, which is adopted from [283], tries to illustrate the data exchange between two AXMLs as an orthogonal concept to materialization. This orthogonality means that it can be controlled whether one AXML returns its effective content or whether it returns the call declarations. If the call declaration is returned, then the receiving AXML will execute the call itself, for example, with its own frequency and mode. The contract between two AXMLs that decides which of the embedded calls are exchanged by result materialization or by call declaration is named *data exchange schema*.

The AXML reference component run-time environment is based on Apache Tomcat with various XML, SOAP, and Web Services Description Language (WSDL) libraries and facilities to allow for a HTML formatted representation of the AXML. The component container

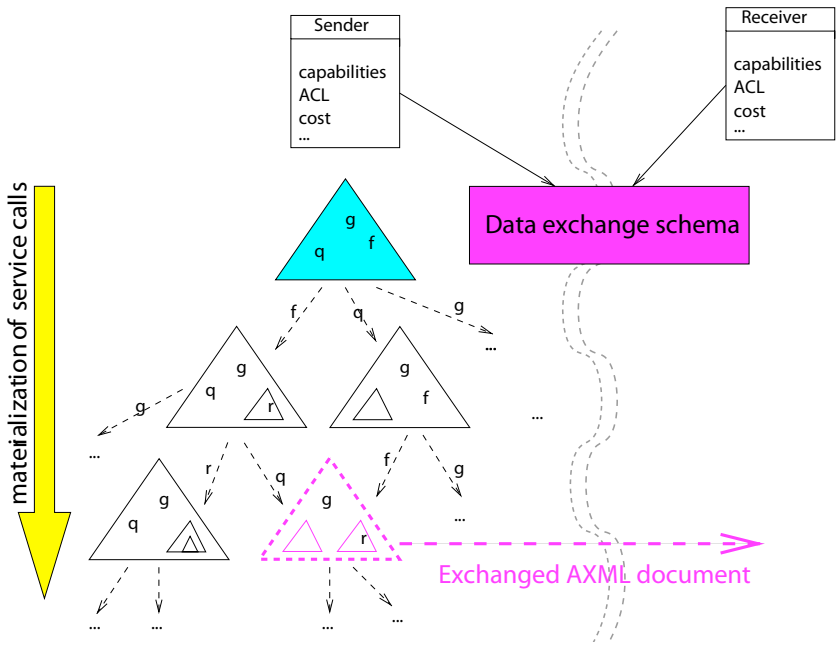

Figure 3.20: The AXML data exchange schema decides which AXML parts are exchanged by materialization or by call declaration (adopted from [283])

implementation is called an *AXML peer*. Figure 3.21 provides a basic overview of the system architecture. The XML materialization of an AXML can be accessed via XML Path Language (XPath) and XML Query Language (XQuery). The reference container implements a form of query optimization concerning materializations. From a given XPath or XQuery the set of relevant embedded service calls is selected. Thus, not all embedded calls are potentially invoked but only the ones within the subtree-scope of the query (and only if the frequency and mode require a renewed invocation). The implementation is publicly available[32].

The Active XML component model provides a document-oriented method for the composition and federation of data applications. Each AXML is primarily an XML document. It is a single file and exchanging AXML or parts of it between component containers is a key characteristic. Each document can embed Web Service calls as active properties. Thus, the approach fully qualifies for the active document metaphor.

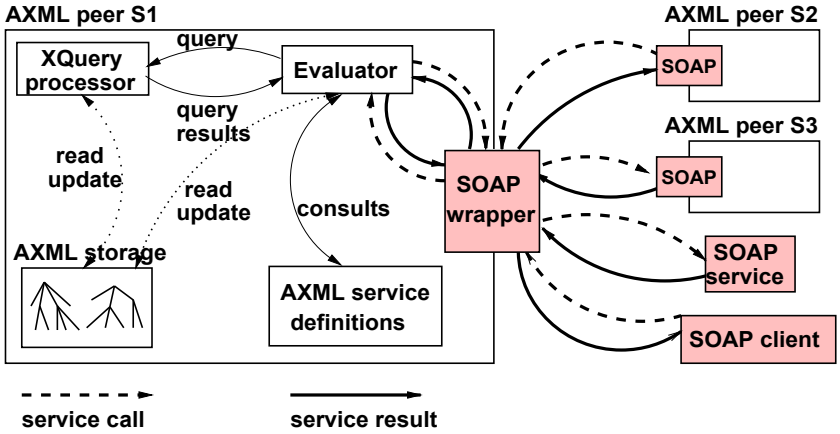

Figure 3.21: The AXML system architecture overview (adopted from [282])

In comparison to Ercatons, organic programming seems to be more versatile because it is based on a full-fledged prototype-based programming model. However, the advantage of AXML is that it provides data synchronization between distributed sites. Using the Active XML approach in a distributed peer-to-peer environment requires the installation of AXML peer run-time containers at different sites, which still requires technical expertise and administrative rights. Thus, ad hoc cooperation would be limited. Still, AXML is a promising approach.

32 http://webdam.inria.fr/axml/

3.5.5 Résumé

Distinguishing Characteristics of Active Document Approaches

This section about active documents has introduced a zoo of technologies. An empiric observation over all active document approaches results in four distinguishing characteristics: direct interaction, operability, portability, and remoteness. The *direct interaction* characteristic describes the kind of interface by that an end-user interacts with the active document, thus, the means that change content units and ultimately trigger active properties. The *operability* characteristic describes the technical boundary in which an active documents can be executed or, in other words, extent of the run-time execution environment. The *portability* characteristic describes whether the active document can be migrated from one run-time execution environment to another. The *remoteness* characteristic describes whether remote interaction with an active document is possible beyond the scope of its run-time execution environment. The *run-time versatility* characteristic describes whether the active properties support ready-made editing or whether they allow for general-purpose logic. A general-purpose versatility is fulfilled if active properties can be changed or re-configured at run-time but it is irrelevant which level of technical expertise would be necessary to change their behaviour. There is a partial correlation between the direct interaction and versatility characteristics. The former describes the *triggers* for active properties and the latter describes the *action* of the active properties. An editor can provide both, the change events as triggers as well as change operations as basic actions. Figure 3.22 illustrates these characteristics.

Four kinds of direct interaction can be observed. Placeless documents and Apple Folder Actions use file operations like drag-and-drop events as trigger for active properties. Microsoft Active Document Containments provide a form of embedded editor. TiddlyWiki and AXML instrument a HTML-based interface. Ercatons provide a HTML-based interface but also a command-line interface via the ESH tool.

Four kinds of operability can be observed. In fact, the operability characteristic have been used for the four categories that have structured the section. Placeless and Apple Folder Actions are limited to their particular file system environment. Microsoft Active Document Containments are bound to the underlying windowing system. TiddlyWiki depends on web browsers, thereby assuming ubiquitous support of CSS and JavaScript irrespective of browser compatibility issues. Ercatons and AXML require each their particular component run-time environment.

Three kinds of portability can be observed. TiddlyWiki, Microsoft Active Document Containments, and Apple Folder Actions are file system artefacts, intrinsically supporting portability because they can be easily copied and moved. Ercatons and AXML can be

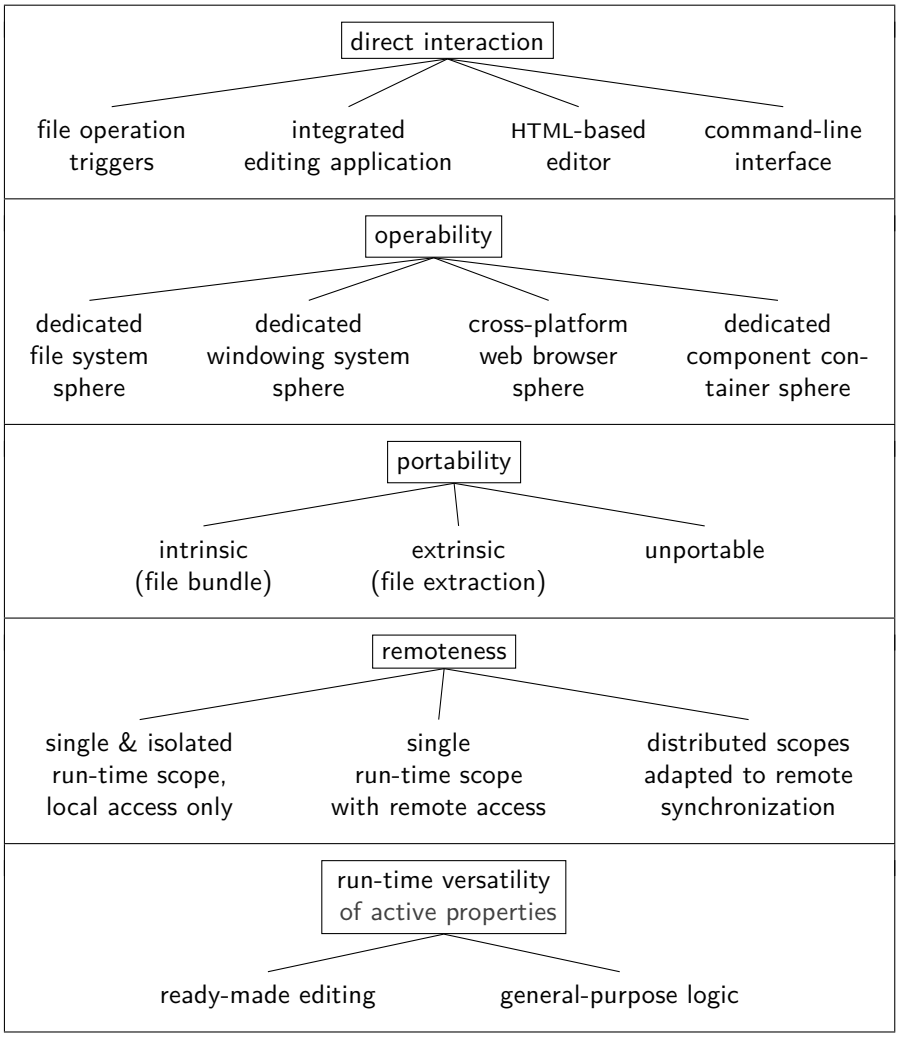

Figure 3.22: Types of characteristics for active document approaches

externalized, thus, they can be migrated from one run-time container to another. For active documents in a Placeless environment, the active properties are seemingly managed inside the Placeless server. There is no indication that a migration into another Placeless site would have been possible.

Three kinds of remoteness can be observed. TiddlyWiki, Microsoft Active Document Containments, and Apple Folder Actions do not provide remote access. Ercatons allow for remote access via web service technology and Placeless allows for remote access via WebDAV. AXML allows for remote access, in addition, it intrinsically allows for synchronization between distributed active documents.

Finally, two kinds of versatility can be observed, as has been mentioned initially during the introduction of the section. Microsoft Active Document Containments allow only

for editing, a user cannot adapt the provided active properties. All other approaches allow versed users to implement general-purpose logic in any active property. Placeless and Apple Folder Actions are capable of executing scripting languages but they do not provide ready-made editing. A TiddlyWiki provides editing but can also be extended through JavaScript logic by modifying its HTML file. The Ercatons approach itself provides editing as well as a full-fledged programming environment. AXML provides editing and is capable of executing any logic by means of web services.

An integrated view on the classification of each approach is illustrated in table 3.3. The characteristics are listed in abbreviated form. The additional aspect whether an approach is discontinued is not as important as for content-oriented workflow approaches. Only the original Placeless documents project is discontinued.

		approaches					
		Placeless Documents	Apple Folder Scripts	Microsoft A.D.C.	TiddlyWiki	Ercatons	Active XML
direct interaction	file operations	X	X				
	editing appl.			X			
	HTML-editor				X	X	X
	command-line					X	
operability	file system	X	X				
	windowing system			X			
	web browser				X		
	comp. container					X	X
portability	intrinsic		X	X	X		
	extrinsic					X	X
	unportable	X					
remoteness	isolated		X	X	X		
	remote access	X				X	
	distrib. scopes						X
run-time versatility	editing			X	X	X	X
	logic	X	X		X	X	X

Table 3.3: Classification of active document approaches

Conclusion on Active Document Approaches

Hopefully, the idea of active documents seems not as eccentric any more as it might have appeared initially in section 2.2.12. Some approaches like AppleScript Folder Actions and TiddlyWiki allow for a comprehensible illustration of the active document metaphor. Approaches like Ercatons and AXML demonstrate that active documents can be applied to distributed systems.

A critical aspect concerning inter-institutional scenarios is operability. Currently, most approaches are bound to a dedicated and pre-installed run-time execution environment. Only the web browser-based TiddlyWiki uses a run-time execution environment that can be assumed as ubiquitously available across various operating systems and hardware platforms. There are two major programming language platforms that can also be considered as ubiquitously available across platforms: Java and the .NET framework. It is an intrinsic design goal to the Java language to allow for run-time environments on any operating system or hardware platform. The .NET platform adopted this design goal and even if the original and dominant Microsoft implementation is tightly-bound to Windows there are several mature .NET alternatives for cross-platform support[33]. Yet, an active document approach that solely depends on a cross-platform programming language run-time environment does not exist to my knowledge. However, the *α-Flow* implementation will be of such kind, using a common Java virtual machine as its runtime environment.

3.6 Summary

This chapter has described state of the art for diverging research domains. First, system integration in healthcare has been discussed, both from a data perspective and from a functional perspective. In the end, standards like HL7 CDA are well suited as a document-oriented foundation for semantic integration in healthcare. However, document-oriented exchange protocols or reference architectures are an open issue.

For process support, activity-oriented workflow approaches have been described. BPMN has been used to provide an impression on the capabilities of standard workflow languages. Subsequently, the limitations to standard workflow approaches have been discussed. For complementary purposes, the alternative paradigm of content-oriented workflow modelling has been described. In order to ease the overarching understanding, key concepts

33 Alternative .NET implementations, for example, are Mono, DotGNU, or CrossNet, each supporting many operating systems.

have been illustrated independently of specific content-oriented workflow approaches. Subsequently, the available content-oriented workflow approaches have been discussed, both their capabilities and limitations. A taxonomy of distinguishing characteristics has been presented in order to classify the diverging approaches.

Finally, the domain of active document technology has been explored. Several approaches have been identified that can be considered to fulfil the active document metaphor. The scope of these approaches ranges from local file systems to distributed software component frameworks. In order to illustrate distinguishing characteristics as well as associational characteristics, another taxonomy has been presented to classify active document approaches.

Three disjoint domains have been studied: inter-institutional system integration, workflows, and active documents. Contributions from all these domains will be necessary to achieve process support for adaptive-evolutionary information systems in healthcare.

4 | The User Story of dDPM

> “Planning is an unnatural process.
> It is much more fun to do something.”
>
> (Sir John Harvey-Jones)

To illustrate the vision of distributed Document-oriented Process Management (dDPM) this section describes a user story of a hypothetical cooperation based on dDPM. This user story is written as an analogy to Berger’s famous study [302], in which the vision of Electronical Health Record (EHR) and integrated care had been described similarly. The dDPM setting is based on breast cancer treatment.

4.1 A Hypothetical Cooperation

“... The patient prepares to leave the gynaecologist’s consulting room. Just yesterday she spotted a knot in her breast. The gynaecologist has just completed a sonography and the knot seems to be dubious. In his mind the future unfolds: he needs to send her first to a radiologist (for a mammography) and afterwards, if necessary, to a colleague at the local hospital (for biopsy and histology) in order to determine if it is breast cancer. Such misfortune will bring her to the university hospital in the major city nearby for primary therapy. It is likely that more than half a dozen of doctors will be involved and the referral to the radiologist is the potential beginning of a complex collaboration to save his patient’s life.

The doctor sits before his desktop computer. He drops the original referral file onto a special icon on his desktop and thereby transforms it into a case file. His colleagues will become participants to the case as soon as they have access to a copy of the case file. The case file is an active document: changing the file in one copy will automatically synchronize information with its distributed copies. The case file supports emergent complexity.

The case file includes the referral as its first information unit and it includes the gynaecologist's own electronic address. The doctor stores the case file to the patient's chip card.

... The radiologist's receptionist can open the case file without any particular installation on his system. Opening the case file will launch an embedded viewer, in which the receptionist can open the embedded referral with his local system. During the admission, the receptionist stores the case file into the local patient record as a file attachment for his boss to access after the examination.

... The radiologist has finished the mammography and his report is ready. Normally he would send the report to the gynaecologist. Instead, he just opens the case file and drag-and-drops the electronic report into it. Because the case file holds the electronic address information for the other participants, the active document will send the new report to the gynaecologist's electronic post box. The radiologist likes it, because the active document holds the collaboration context and makes things easy.

... The patient returns to the gynaecologist's office. The gynaecologist opens his copy of the case file and it automatically synchronizes its content with the gynaecologist's electronic post box. The report of the radiologist appears as the newest information unit. He opens the report. The value of the Breast Imaging – Reporting and Data System (BI-RADS) indicator is higher than four; the patient must go to the local hospital for a biopsy. Thus, he prepares another referral and puts it into the case file.

Again, he copies the current case file on the patient's chip card. The hospital will contribute the report of the biopsy operation. The tissue and the case file will be sent to the pathologist. He will contribute the histology report to the case file. The gynaecologist's case file will always receive any information that is contributed by future participants.

The gynaecologist looks at his patient. He assures her that there is still some chance that the histology proves that it is not a malignant tumour. ..."

4.2 Technical Implications

The user story involves various types of human-machine interaction. First, we have the transformation from the case-initiating referral document into an active document (i.e. α-*Doc*) for distributed case management. Figure 4.1 illustrates the initiation and the changes to the distributed case file. It outlines the scenario from a more technical perspective.

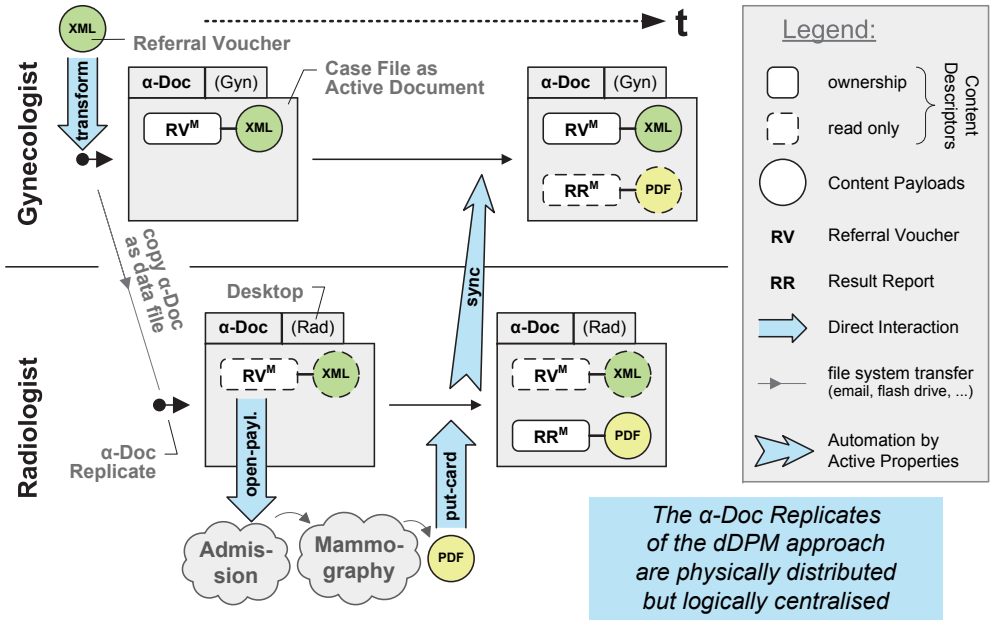

Figure 4.1: An *α-Doc* that changes during the user story

The referral is supposedly in HL7 CDA format, thus, it is illustrated in the figure by a circle being labelled with "XML". The four major rectangular shapes being labelled with "*α-Doc*" each illustrate the active document in different content states at various points in time and at the two different sites.

The initializing referral is an atomic digital document file ("file" like data file as in computer file system). The referral and the mammography report become content units within the case file ("file" like dossier, envelope, or ring binder). The active document is molecularly structured, internally. For now, assume that it is basically something like a (self-executable) ZIP archive. More specifically, the Java programming language allows for executable ZIP files in form of JAR files.

The assimilation of the "XML"-labelled circle into the active document is meant literally. The circles within the *α-Doc* are accompanied by small rectangles; these are small descriptor data files. Descriptors will ultimately allow for arbitrary process-relevant meta data about a content unit. For now, assume that it just describes the name for its content unit and stores which participant has contributed the content unit. The content unit and its descriptor are symbolized with a dashed frame if the content unit is not owned and contributed by the actor at the site.

The user story describes that the active document can be stored to a chip card, it can also be a flash drive or anything that stores data files. Thus, from a file system handling perspective the molecular active document is still a coherently handled and portable

data file unit and it must be possible to apply file copy & move or to use it as an E-Mail attachment or file attachment to a local patient record. This characteristic is symbolized by the "copy *α-Doc*" arrow in figure 4.1. It simply replicates the *α-Doc*.

At the radiologist's site, the *α-Doc* can be executed and an embedded editor lists the included content units. For now, assume that the embedded editor allows for opening the payloads as easily as a file attachment within an E-Mail application. The payloads are opened with a locally available application, likewise to the E-Mail attachment metaphor. The content unit for the referral has now a dashed frame at the radiologist's site, because a participant is not allowed to edit foreign content units. Each participant can contribute further content units, for example the "PDF"-labelled circle for the mammography report, by simple drag-and-drop.

In section 3.5 we have seen different active document technologies with three types of direct interaction. The file operation triggers and the embedded editor are relevant for the drag-and-drop-based contribution scenario. If the user is currently executing the active document and has opened the embedded editor, he or she can drag-and-drop data files into the GUI area (cf. the TiddlyWiki style or Microsoft Active Document Containment style). If the user is not currently executing the active document, it is still possible to contribute new content units just by using an operating system-based or file system-based drag-and-drop (cf. the Placeless style or AppleScript Folder Actions style). The operating system executes the active document with a parameter for the new data file, thus, the active document can directly display the specific window for just accomplishing a single contribution.

After a contribution has been performed locally, all other known participants must be provided with a copy of the new content unit. This is illustrated with the specially shaped "sync" arrow. For now, assume that each contribution must be synchronized with all other *α-Doc* replicates of the same case. Notably, the *α-Doc*s are only active if they are explicitly executed by its user. The remote replicates are not (necessarily) online at the same time; instead, an offline characteristic is dominant. Thus, the synchronization facilities must involve an underlying infrastructure for store-and-forward messaging.

A user story focuses on the operational embedding, i.e. the conduct of an *α-Doc* case file within healthcare routine. The user story naturally emphasizes active document requirements. Thus, the taxonomy of characteristics for active document approaches (cf. sect. 3.5.5, p. 149) can be applied to the *α-Doc*s of the dDPM approach. Figure 4.2 highlights these characteristics using a green background filling.

In conclusion, the dDPM implementation needs to support two types of direct-interaction, an embedded editor and file system drag-and-drop. A third kind of direct interaction via a command-line interface could be used to automatize interactions like *open-payload*

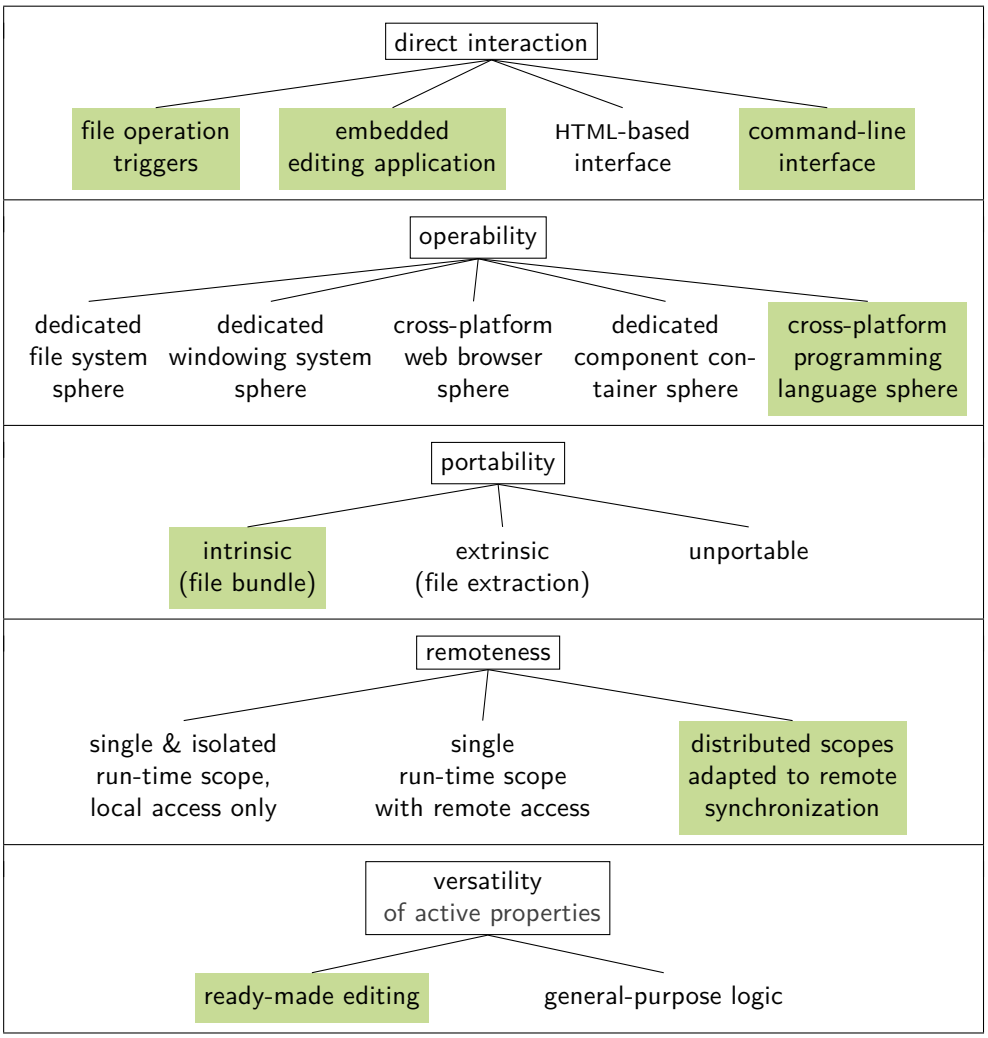

Figure 4.2: Active document characteristics of the dDPM approach

and *put-card*, in figure 4.1, by offering the local information system means to interact with the case file via command-line. Operability can be based on an unmodified Java cross-platform run-time environment. Portability must be intrinsic via file bundling. Remoteness must be adapted to synchronization. From the perspective of versatility, only editing facilities for the case file contents is intended. Editing includes not only editing of the contributed medical content but additionally includes embedded editing applications for the shared work-list or the list of participants. The editing will ultimately imply background actions for local versioning and remote synchronization. However, a distributed case file is not necessarily obliged to allow users the definition or execution of general-purpose logic as the actions of the α-*Doc* active properties.

The next chapter complements the requirements from a content-oriented workflow perspective. The cross-organizational and cross-domain activities for breast-cancer treatment are further refined. The scenario is continued as an exemplary use case for distributed document-oriented medical processes.

5 The Process Conception of dDPM

> “No plan survives contact with the enemy.”
>
> (Helmuth von Moltke, “The Great Silent One”)

The chapter is split into seven sections. The first section details general process characteristics of inter-institutional process and of case-driven processes. The following five sections provide illustrative healthcare use cases. Each use case results in implications and requirements on the process conception.

The five sections are about a) basic document-orientated work-list conception, b) ad hoc decisions and team synchronization, c) case fragmentation and process roles, d) user-defined indicators and process templates, as well as e) content versioning and process termination. The final section discusses the process conception from the perspective of characteristics that have been derived from content-oriented workflow approaches.

5.1 Inter-Institutional and Case-Driven Processes

An environment for distributed Document-oriented Process Management (dDPM) is described by an inter-institutional cooperation that involves knowledge workers who cooperate in the context of a case by means of documents. The characteristics of such process environments can be distinguished into two categories (cf. fig. 5.1): either those that are common to any inter-institutional process or those that are specifically relevant to case-driven processes. The case-driven characteristics apply to healthcare but are also valid in law (legal case management), sales (lead acquisition), insurance (claim handling), or science (research funding processes). Inter-institutional characteristics are domain-independent and emerge with increasing process scopes.

From the inter-institutional perspective, a process involves specialists at different sites that cooperate in a distributed process. It is significant that the participants have equal rights and are not subordinate to each other because they come from autonomous

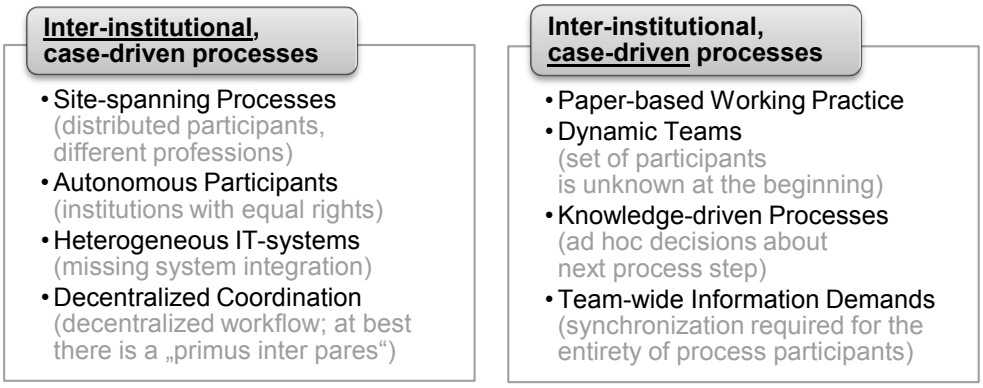

Figure 5.1: Universal process characteristics for dDPM environments

organizations. The cooperation of different organizations implies the existence of heterogeneous information systems. Thus, inter-institutional process support must consider the infeasibility of a priori system integration. The equal right of determination in the process steering implies that a process coordinator does not necessarily exist. Hence, the process is not only distributed but also management is decentralized. However, sometimes there might be a "primus inter pares" who directs his peer participants and who guides the overall process.

From the case-driven perspective, a process distinctively involves a paper-based working practice and a case file is successively filled with enquiries, forms, reports, surveys, notes, or typescripts of any kind. The case participants are knowledge workers and the next process steps are frequently decided ad hoc. Thus, the team is dynamically changing and the eventual set of participants is not necessarily known at case initiation. As a consequence, information transfer is not only necessary between consecutive activities but access to case information is required independently of activities via case file synchronization for the entirety of process participants.

In contrast to traditional case handling, dDPM emphasizes the transition from local to distributed perspective: case handling, as introduced by van der Aalst et al. [69], focuses on workflows like clinical pathways and requires semantic integration of medical data based on data objects and forms. The authors acknowledge that their approach to case handling creates an integration problem because the state of a "case" is derived from intra-institutional "data objects" with well-known schemas that cannot be separated from the workflow schema of the original approach (cf. [69]).

dDPM may complement the idea of case handling. The dDPM approach aims to provide case handling in distributed environments and emphasizes document-oriented systems integration. To resolve system integration issues, in dDPM it is necessary to separate record-centric intra-institutional models from canonical document-oriented inter-institutional

models. The features of documents in contrast to interfaces and records have been discussed in method section 2.2.3. As in any paper-based working practice, shared documents are explicit exports from the local systems. In healthcare, they implicitly reflect selective parts of the Electronic Medical Record (EMR).

5.2 Process Analysis: Document-Oriented Work-List Conception

This section explains how to consider processes in a document-oriented fashion. It describes the diagnosis of breast cancer and includes information from my previous publication [55]. In Germany, the cooperation for a breast cancer diagnosis is managed by a gynaecologist of primary care and involves partners like a radiologist, a clinical gynaecologist, and a pathologist.

The figures that are used to illustrate the use case are independent of any technical framework. The drawings are free form and visualize the paper-based working practice in healthcare; they have no intention to be a formal content-oriented workflow notation. However, the analysis of the paper-based process scenario yields requirements for a content-oriented workflow approach.

5.2.1 Breast Cancer Episode: Pre-Therapeutic Diagnostics

The goal of this treatment episode is to find out whether a knot in a breast is actually malignant cancer or not. The treatment begins with a patient visiting her gynaecologist. The user story in chapter 4 has included minor parts of this initial episode. In addition, an activity-oriented representation of the initial episode of breast cancer treatment in the form of a Business Process Model and Notation (BPMN) diagram has been used in the state-of-the-art chapter (sect. 3.2.1, fig. 3.4, p. 103) to illustrate BPMN by example.

In the following discussion, the superscripts for the actors are A for ambulant (primary care) and H for hospital (secondary care). The superscript of the referrals is I for instruction; there are other types of referrals that will appear in later on episodes. The subscripts for the referrals are M for mammography, B for biopsy, and H for histology.

After the anamnesis, the gynaecologist Gyn^A conducts a sonography. If the result is either malignant or dubious, he/she will send the patient to a radiologist for mammography. After the radiologist's treatment, the mammography report on diagnostic findings is sent back to Gyn^A. The gynaecologist evaluates the radiologist's findings, primarily the medical indicator Breast Imaging – Reporting and Data System (BI-RADS). A BI-RADS

value equal or greater than four strengthens the indication for a malignant tissue; the highest BI-RADS value is five. Any BI-RADS≥4 decides that the patient has to be send to a hospital for a biopsy. The biopsy involves another gynaecologist at a clinic (Gyn^H). The tissue is taken by Gyn^H and sent to a pathologist for histological diagnosis. The histology provides definite evidence, it is the "ultimate" diagnostic authority. The Gyn^A is the one who takes the histology result and is responsible for informing the patient. In the malignant case, another episode begins now by sending the patient to a breast cancer treatment centre for primary therapy.

Even such linear process provides variations, e.g., for elder patients: if the mammography provides high evidence the biopsy and histology is sometimes skipped. Instead, the patient is immediately referred to a clinic for primary therapy. Such skipping is only considered for elder patients if by judgement of Gyn^A the biopsy is considered as a disproportionate burden.

If we are looking back on the BPMN diagram of figure 3.4 on page 103 there are three significant characteristics about an activity-oriented representation: a) Each treatment has been outlined, b) there have been two conditionals elements that have explained medical decisions (after sonography the control flow has split due to "without findings" or "dubious", after mammography the control flow has split due to "BI-RADS≤3" or "≥4") , and c) the arrows that cross the swimlanes (i.e. the data flow between the participants) have neither named nor specified any exchanged artefact. This will be significant when a content-oriented representation will have been derived.

5.2.2 Implications on Process Conception

The dDPM approach eliminates any modelling of expert activities in its coordination model. This is possible because no medical activity in inter-institutional healthcare scenarios can be automatized. Furthermore, it is necessary because any decision for process routing requires either a domain- and section-specific decision support system or a human decision. Activities are fused into the dDPM conception being completely represented by their result artefact. On this account, we will reinterpret the initial treatment episode for breast cancer (see figure 5.2).

At the beginning, the anamnesis is documented and the sonography results in an according report. If the sonography report suggests evidence to send the patient to a mammography, an instructive referral voucher (RV^I_M) is written by the ambulant gynecologist (Gyn^A). After the radiologist's treatment, a mammography report on the diagnostic findings is written and sent back to Gyn^A. If the BI-RADS suggests evidence

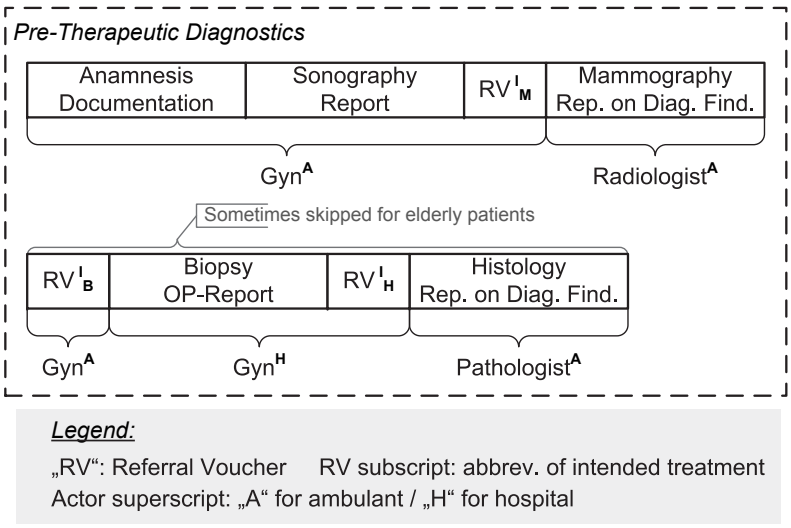

Figure 5.2: The initial treatment episode remodelled in documents

for a biopsy, another referral voucher for biopsy (RV^I_B) is created, destined for a gynaecologist at a hospital (Gyn^H). After the biopsy itself, taking the tissue sample, a third referral voucher for histology (RV^I_H) is used to send the tissue to the pathologist. The pathologist writes a histology report. The histology report is sent back from the pathologist to the Gyn^H at the hospital, who bundles the report with a short report about the biopsy operation and finally delivers the reports back to his ambulant colleague Gyn^A.

There are several characteristics about the dDPM diagram of figure 5.2: a) The communication edges are not modelled, b) the conditional elements of the medical decision processes are absent. c) The whole treatment episode, with its goal to find out whether or not a knot is actually malignant, can be considered as one report that is successively filled by the participating institutions, and d) Each contributed artefact has a dedicated ownership.

The characteristic (a) could be criticized as a deficit. Yet, bilateral communication can be considered as a special case of publish/subscribe and complex cooperative scenarios require a multi-cast distribution configuration. Within a case all published reports are shared with all team members. In medical processes, as long as a physician is part of the team he must get delivered any published content units. There is no demand for content-based publish/subscribe[1], quite the contrary, a technical restriction of the distribution by content filtering would be legally precarious. However, using the case as context for distribution can itself be considered as topic-based publish/subscribe. Content filtering

1 Well-known surveys on the different kinds of publish/subscribe systems are, for example, available by Eugster et al. [303] and Hinze et al. [304].

may be applied locally. Furthermore, it must be possible for participants to leave the team, especially for one-time contributors like pathologists. The attribution of participants into active ones or former ones determines overall case-related communication efforts. All active participants form a fully connected network.

Regarding (b), conditional elements and even rule-based activities in activity-oriented workflow schemas are insufficient for knowledge-intensive expert decisions because any formalization tends to be an oversimplification. Instead, human decision tasks would be necessary as schema elements, possibly supported by complex and specialized local healthcare applications for decision support. Thus, the characteristic (b) basically provides the distinction between intra-institutional decision processes and the intra-institutional coordination work. In many intra-institutional scenarios, workflow formalizations have the purpose to ensure process compliance. For inter-institutional process support in healthcare, the workflow facility has to strictly follow any human decisions. The dDPM approach achieves flexible coordination support by considering the artefacts themselves sufficient as triggers for workflow progression.

Characteristics (c) and (d) are important aspects and motivate *two levels of granularity* for the artefacts. The document artefacts from the initial episode ideally act as a single document, for example, all the successively contributed documentation from primary therapy constitutes a single input for the next episode, the primary therapy. The two artefact-granularity-levels that can be observed are: i) the units of validation and organizational accountability, analogous to paper-based artefacts. ii) The coherent collection of such documents, considered as a successively written collaborative case file. These two artefact levels of dDPM are the equivalent to the two levels of activity-orientation, process and activities.

Content Units and Basic Progress States

The user story in chapter 4 already indicated that the coherent collection of content units within a treatment episode forms a distributed case file that will be represented by an active document, the *α-Doc*. However, the paper-based artefacts within the case file are constitutive to the content-oriented workflow and require a concise term. The term *card* is well suited, for two reasons.

The first reason relates to experiences in project Distributed Electronic Patient File Update System (DEUS) that have been discussed in section 2.1.2. In DEUS we used Digital Cards (DCs) in analogy to the patented *information cards* and the Higgins project's *I-Cards*. The original motivation for the term was that "card" provides a paper-based association. A card is still a coarse-grained content-unit, it is self-contained, and it can basically exist independently. The intended granularity of a card is subtly

more fine-grained than the granularity that is experienced from paper-based working practice in healthcare. A single card should ideally contain only one diagnostic finding, piece of clinical evidence, therapeutic measure, or prescription. In the end, "card" is in association with "stack of cards". The stack of cards is equivalent to the patient file in DEUS and the case file in dDPM.

The second reason is based on Scrum (cf. sect. 2.2.11) and the usage of "card" as part of the terminology from the agile software development methods. Scrum describes a development episode by a set of tasks and articulates them in form of task cards. The same is done by dDPM. Each dDPM paper-based artefact represents a unit of work, basically implying an underlying activity. The process status of each content unit is unstarted as long as only a placeholder exists. If a physician provides a signed report, the work item accordingly becomes completed. From the perspective of Scrum, the unstarted cards are a *backlog* of work items.

In conclusion, each dDPM work item is articulated as a card. Each dDPM card results in a shared content unit within the context of a distributed case file. The status of the cards must be tracked. Scrum is tracking the process state of cards with adornments, i.e. textual or graphical markers on the cards. The same is done by dDPM and adornments are used in form of data attributes to store process-relevant state of cards.

5.2.3 Process Model Requirements

A summary of the process considerations is provided in form of list of numbered and named process model requirements. Mandatory content-oriented process characteristics are listed as *Core Requirements* (CR#). All requirements derived from the initial episode represent mandatory process characteristics. Later sections will provide optional characteristics in form of extended process model requirements.

[**CR1**] **Result-Oriented Work-List** – The work items are content units that successively complete the patient-related information demands. The succession of work items is derived tacitly from the diagnostic-therapeutic cycle. In basic linear processes like the pre-therapeutic diagnostics, the case file may initially be understood as circulation folder.

[**CR2**] **Partial Results from Various Actors** – The overall case file is like one big form with sections that are filled by different actors. The actors can be uniquely identified.

[CR3] **Card Metaphor** – The separation of content units is based on organizational accountability. Thus, there are disjoint sections of the overall case documentation being called cards. Cards are also the units of validation, analogous to paper-based artefacts.

[CR4] **Adornments as Card Progression States** – The life-cycle of content units is articulated in states. Concerning the paper-based card metaphor, states are considered as adornments, i.e. textual or graphical markers on the cards. Content-oriented states like empty, draft, signed, or released may be preferred to activity-oriented states like unstarted, ongoing, and completed.

[CR5] **Domain User Roles** – Unique individual actors like have abstract domain roles like "gynaecologist". An initial responsibility for a card and its later content contribution can be defined by a domain role.

5.3 Process Analysis: Ad Hoc Decisions and Team Synchronization

This section details how alterations of the process appear and how the tumour conference represents a true collaborative process step that requires prior information synchronization. It describes the primary therapy of breast cancer and includes information from my previous publication [305]. The treatment of breast cancer, in Germany, is organized by accredited in-station breast cancer treatment centres cooperating with partners like oncologists, pathologists, and radiologists [46].

5.3.1 Breast Cancer Episode: Primary Therapy

Primary therapy begins when a patient is referred to a hospital that is part of a cancer treatment centre by an ambulant gynaecologist. The goal is to remove the cancer by surgical operation. It operation is preceded with additional diagnostic measures and after the operation the removed tissue is further analysed. The final purpose of primary therapy is to accord the adjuvant therapy during a case-specific tumour conference. The overall document-oriented conception is detailed by figure 5.3.

In the following discussion, the superscripts for the actors are H for hospital, A for ambulant, and TC for tumour centre. The superscripts of the referrals are H for hospital, C for conciliar, and I for instruction. The subscripts for the referrals are BC for breast cancer, HI for histology, and AS/PX/BS for special diagnostic treatments.

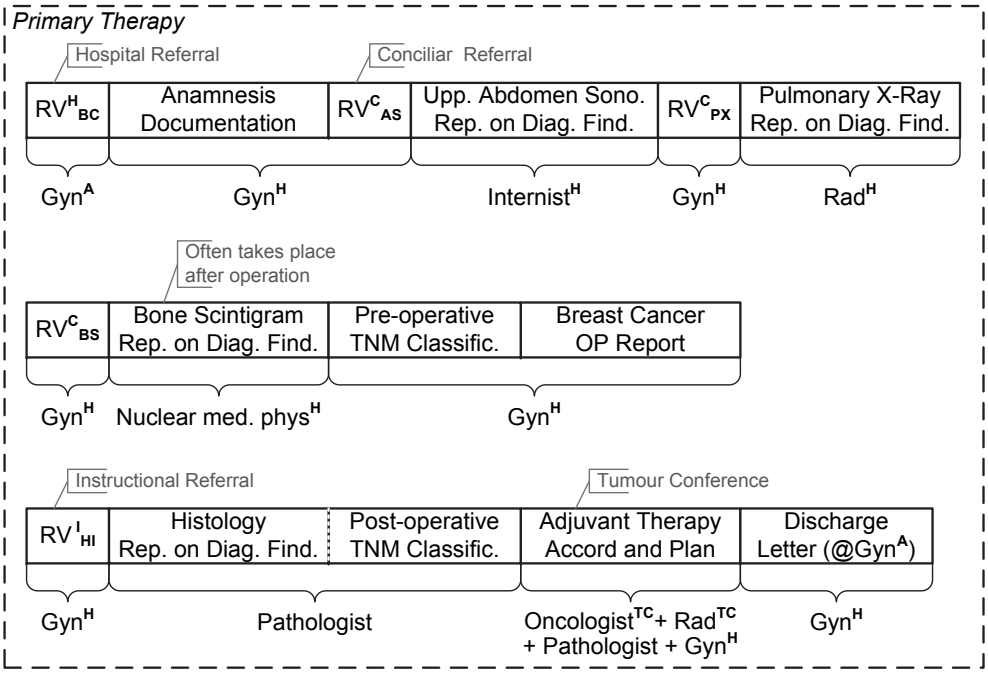

Figure 5.3: The primary therapy being represented in document artefacts

After the initial anamnesis by Gyn^{H}, the patient undergoes three diagnostic steps: i) upper abdomen sonography, ii) pulmonary X-ray, and iii) bone scintigram. All three diagnostic treatments are subsumed under the name "staging". Each of the three additional diagnostics is provided by different actors—namely an internist, a radiologist, and a nuclear medical physician. The referrals to these participants are conciliar ($\text{RV}^{\text{C}}{}_{\text{AS}}$, $\text{RV}^{\text{C}}{}_{\text{PX}}$, and $\text{RV}^{\text{C}}{}_{\text{BS}}$).

The order and the completeness of the staging is decided situation-dependent (i.e. ad hoc process decisions). The upper abdomen sonography and the pulmonary X-ray are always done, and commonly they are accomplished on admission day. The bone scintigram may or may not be done; it does not need to take place before the operation. If it is done it actually is applied after the operation in most cases.

A so-called TNM classification is used to specify the malignant tumour; TNM is an abbreviation for tumor, (lymph) nodes, and metastasis. The preoperative TNM classification is done by the Gyn^{H}, who is in charge of the surgery, and it defines the exact surgical method (e.g., lumpectomy vs. mastectomy). Gyn^{H} then performs the surgery, which commonly takes place on the day after admission. The extracted tumour, tissue, and lymph nodes are sent to the pathologist by a referral voucher $\text{RV}^{\text{I}}{}_{\text{HI}}$. The pathologist contributes a histology report along with the postoperative TNM classification.

The tumour centre itself is considered as a separate institution; with the oncologist as the head of the tumour centre. Gyn^{H} contributes all reports on diagnostic findings to the tumour conference. Other participants are a radiologist and the pathologist. Whereas the pathologist is the same one who has provided the histology, the radiologist need not be the one from staging, but can be a radiologist assigned to the tumour centre, Rad^{TC}. The duty of the tumour conference is to accord and plan the individual adjuvant therapy for the patient: it consists of chemotherapy, radiotherapy, and hormonal therapy. For each of them there is a spectrum of medical variations. The adjuvant therapy plan is developed at the tumour conference and manifested in a document.

In a final step, Gyn^{H} of secondary care writes a discharge letter for Gyn^{A} of primary care. The ambulant gynaecologist as primary physician will manage the adjuvant therapy and is in charge of the post-operative care.

5.3.2 Implications on Process Conception

The characteristics of the initial classification episode are also true for the primary therapy episode: Primarily that the whole treatment episode can be considered as one report that is successively filled by the participating institutions and that each contributed artefact has a dedicated ownership.

Several characteristics of the primary therapy episode can be highlighted: a) The involved conditionals in each activity require medical experts that cannot be formalized or automated, b) the results of each activity is again represented by an artefact but is shared to multiple participants during the episode, c) the exact number of steps and participants is not necessarily decided at the initiation of the process, and d) the sequence of several of the activities is commonly decided ad hoc.

For cancer, as for multimorbid or chronic diseases, the exact treatments and participants are in general unknown in advance (cf. [14]). Even the previous outline of the primary therapy episode itself is only a basic illustrative reference, thus, a simplification. The process model must not constrain the order of the process steps. The order of items in a shared work-list of knowledge workers is only a guideline and not an actual rule.

For multimorbid diseases or diseases with unclear symptoms, multiple treatment steps are usually indicated in parallel. For any kind of disease, there may be triggered diagnostic tasks like histological interpretation in parallel because these do not require the patient and can generally be assumed to run in parallel. Furthermore, the work items for inter-institutional cooperation spans both the tacitly assumed treatment activity as well as the authoring of result reports. The authoring activities do not require the patient and must be assumed to run in parallel.

Work items like card placeholders are managed in a prioritized work-list. The order of the treatment activities is suggested by the prioritisation in the work-list. However, the real-world order is ultimately decided by routing the patient, which is not part of the content-oriented process model. In paper-based working practice, the actual order of treatment steps is documented inside the reports but it is related neither to the dispatch time or receipt time of result reports nor to the creation time of referral vouchers. However, the point in time at which the execution of the underlying treatment activity has happened could be indicated within the shared work-list by means of user-defined adornments. These will be further motivated and explained in section 5.5. In conclusion, the prioritisation of the work items is considered as a *tacit order*: it can be changed freely and the order of the underlying activities must not necessarily adhere to the work-list order.

Although the first episode already involved pairs of referral vouchers and result reports, these pairs are now considered in the refined context of a work-list that represents parallel work with tacit order. Naturally, the referral voucher is always contributed and "owned" by another physician as the according result report. Thus, the actor information does not bind both content units but separates them. Still, it seems best practice to have a referral followed by its result report, as it has been illustrated in figure 5.3 in the context of the three diagnostic treatments of the pre-surgical staging. If the pair-wise occurrence should not be broken by re-ordering the work-list, then the pairing needs to be explicit with a content unit relationship. Notably, a missing referral voucher document does not necessarily imply that the according diagnostic or therapeutic measure would be prohibited. A referral voucher can be supplied later if it is missing for some reason. Thus, the cohesion between referral voucher and result report does not imply strict work order. In the final analysis, a cohesive-content relationship may be created between two content units. The semantics is that coherent content units are habitual neighbours in the work-list. Thus, the semantics only affects re-ordering of the work-list because cohesive content units should be handled atomically. In addition to the neighbour position, the cohesion might be indicated with an extra visualization like a connector; figure 5.4 provides an example. Finally, the cohesive-content relationship is just a convenient way to ease work-list re-ordering.

The apparent consequence from the considerations about soft cohesive relationships is the request for a dependency relationship that implies strong order. As a disclaimer, it should be pointed out that none of the exemplified use cases actually requires strong work order because work order is always circumstantial. However, a strong dependency may be articulated by a relationship from one content unit (placeholder) to other content units whose completed contents are required as necessary information input. Each required-content dependency ("content production B requires content A") implies strict work order ("first activity A then activity B"). However, the content-oriented dependency

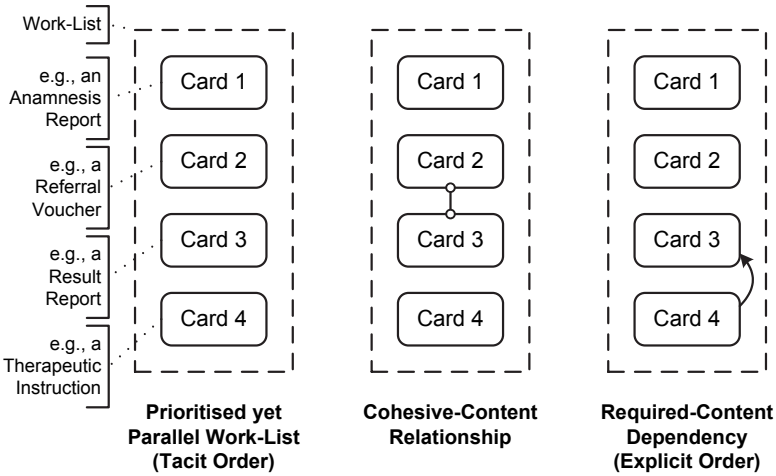

Figure 5.4: Example of prioritised work-list of content units, an exemplary visualization of a cohesive-content relationship between referral voucher and result report, and an exemplary visualization of required-content dependency.

arrows are always reciprocal to the activity-oriented control flow conception, thus, the tip of the arrow is drawn on the reverse side. The reasoning of this seemingly "backward-chained" visualization stems from a principle of responsibility: "I am only responsible and authoritative for my own work and can only demand but not command work of others" (keeping in mind that in inter-institutional cooperation the participants are basically assumed to have equal rights). In conclusion, the required-content dependency "B requires A" is a *backward-chained mandatory demand*, whereas its control flow equivalent "A then B" is a *forward-chained mandatory command*. Still, in content-orientation the referencing work item (B) will be blocked until its precursor (A) is fulfilled, i.e. until its result has been released in valid form.

Figure 5.4 illustrates the work-list of content units. It also provides an exemplary visualization of a cohesive-content relationship. between referral voucher and result report. An exemplary required-content dependency is illustrated on the right side.

Finally, in a cooperation there are occasions that require the full access to all previously compiled information. The tumour conference requires all previously achieved reports on findings. Notably, the tumour conference is a team activity. The conference can be executed both in person or virtual via a remote telephone conference. In preparation for a virtual tumour conference, the reports need not only to be routed to a single participant but it is required to synchronize information between multiple participants.

5.3.3 Process Model Requirements

In addition to the mandatory *Core Requirements* (CR#), optional requirements are listed as *Extended Requirements* (XR#). Mandatory concepts are the ones that are directly deduced from the healthcare use case description. In contrast, the articulation of content cohesion as formal relationships is not necessarily an accustomed domain concept. Also, the explicit data dependency for strong work order did not actually appear in the exemplified use cases uses because work order is always circumstantial in healthcare. Thus, the according process model requirements of dDPM are considered optional.

[**CR6**] **Parallel Work, by Default.** – Multiple possible next treatment steps are usually indicated in parallel in cases with unclear symptoms. Furthermore, the work items for inter-institutional cooperation are the contributions of result reports. The distributed progression in terms of authoring is generally assumed to be accomplished in parallel.

[**CR7**] **Tacit Order by Non-Binding Prioritization** – Work items like card placeholders are managed in a prioritised work-list. The least prioritisation criterion is the creation order of the work items. A backlog of unfulfilled work items is assumed to provide sufficient orientation for work order in most cases. Comprehensive subtleties in order dependency will be tacitly adhered to, as in paper-based working practice.

[**XR8**] **Cohesive-Content Relationships** – A soft relationship may be articulated between cohesive content units. Content cohesion does not imply work order; however, it should be visualized.

[**XR9**] **Required-Content Dependencies** – A strong dependency may be articulated by a relationship from one content unit (placeholder) to other content units. The referencing work item will be blocked until its precursors are fulfilled, i.e. until their results have been released in valid form.

[**CR10**] **Location-Independent Access** – Team activities achieve collective results that the participants would be incapable of accomplishing when working alone. Basic circulations are not sufficient in collaborative scenarios. Instead, location-independent access must be provided that allows for remote team activities. For example, location-independent access can be achieved by remote synchronization of case files.

5.4 Process Analysis: Case Fragmentation and Process Roles

After the primary therapy, i.e. removal of the tumour, the post-operative care and the adjuvant therapy run in parallel for the first six months. The adjuvant therapy will be described to exemplify case fragmentation and process roles.

5.4.1 Breast Cancer Treatment: Adjuvant Therapy

The adjuvant therapy for breast cancer treatment subsumes three disjoint therapeutic processes: chemotherapy, radiotherapy, and hormonal therapy. In contrast to primary therapy, the treatments during adjuvant therapy are ambulant. Chemotherapy requires an oncologist and radiotherapy requires an radiologist. The gynaecologist as primary physician of a breast cancer patient manages the overall adjuvant therapy and refers the patient to the oncologist and radiologist. The hormonal therapy is performed by the gynaecologist himself without cooperation.

The radiotherapy is (usually) performed for six weeks on a daily basis (cf. fig. 5.5). From the perspective of inter-institutional cooperation, it implies a quite minimalistic data exchange if no complications occurs. Radiotherapy often-times involves only an initial referral and a final report from the radiologist after six weeks. In some cases an interim report is written. From a medical perspective, it must be understood that the patient still visits the gynaecologist in parallel for post-operative care. Thus, patient condition aggravation is mainly discovered during post-operative care, which will be discussed in the next section.

Chemotherapy (usually) takes six months and (usually) requires only six visits in total. Accordingly, the visits are arranged every three to four weeks. Each of these visits forms a so-called *cycle*: each chemo treatment results in a therapy report by the oncologist and entails a follow-up check by the gynaecologist. The follow-up check is also documented and the according report is shared back to the oncologist. Thus, chemotherapy is a bilateral cooperation.

The hormonal therapy (usually) takes five years. Its applications are (usually) performed monthly. From an organizational perspective, the hormonal therapy goes together with visits for post-operative care.

Not necessarily all three types of therapy are conducted. Only the radiotherapy is implemented as good as always. Chemotherapy and hormonal therapy can be done in combination. However, for some tumour types, only chemotherapy is applied and for

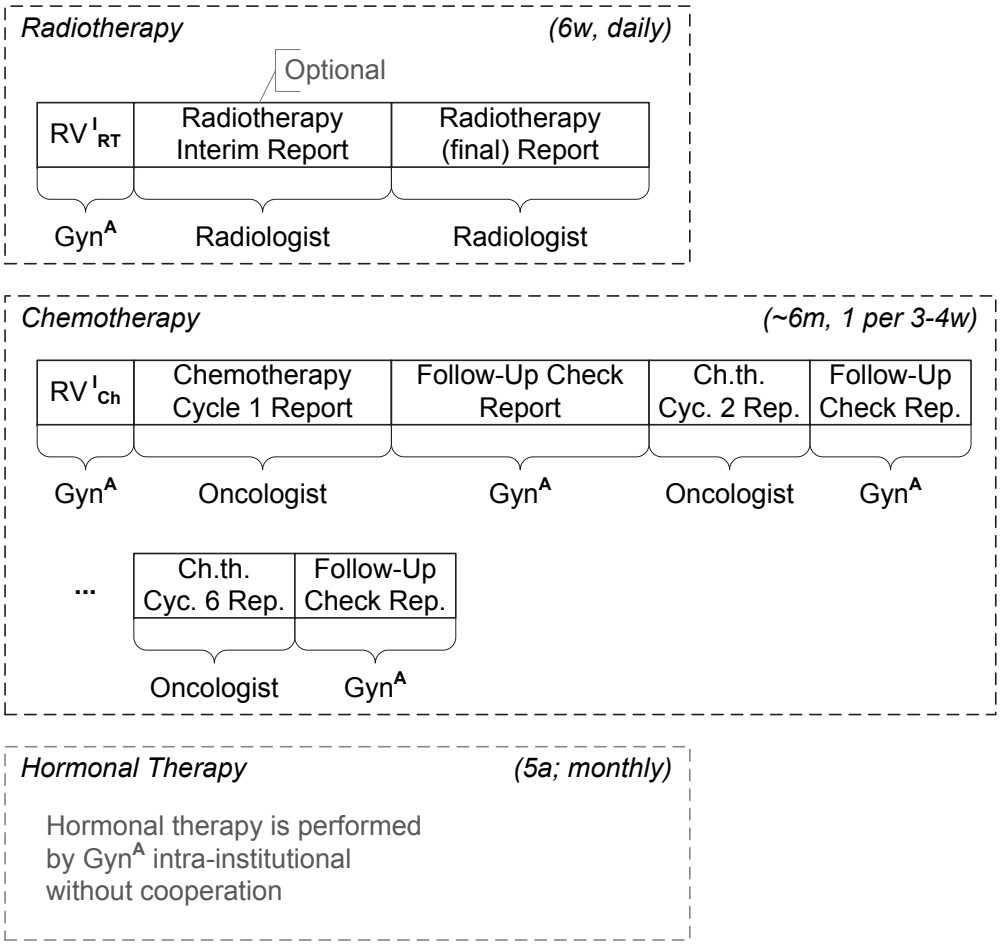

Figure 5.5: The threefold process of adjuvant therapy for breast cancer being represented in document artefacts for inter-institutional cooperation

other types, a hormonal therapy is considered sufficiently effective. In addition, a patient can influence his own therapy plan based on her personal environment, age, or family considerations. For example, if the statistical life expectancy is very low, a patient might decide against chemotherapy because it implies a heavy strain on her health and on her family life.

5.4.2 Implications on Process Conception

The gynaecologist acts as a coordinator. From the perspective of each participant, the involvement in the adjuvant therapy of each patient appears as one case. Thus, from the perspective of the coordinator, the adjuvant therapy is a single case. However, from a perspective of inter-institutional cooperation, the physicians of chemotherapy and radiotherapy perform their therapeutic measures without any need for cooperation. An

additional criterion is that the treatment period and intervals are quite different. Hence, in terms of inter-institutional cooperation, the overall adjuvant therapy can be considered as the composition of two bilateral episodes and one single-handedly episode.

It is a matter of participant preference whether both disjoint therapies are handled by a single conjoint distributed case file or whether the content units are exchanged by the means of two separate distributed case files. In general, it should be possible to choose freely between both variants. In conclusion, any composite medical case can potentially be fragmented into several case episodes. For dDPM purposes, each case episode has its own distributed case file. Thus, the dDPM term "episode" is intended to be one degree more specific than the general term "case". The scope of one dDPM episode defines exactly one scope of remote data synchronization. For sub-structuring, it should be possible to group content units into sub-lists, within an episode and its distributed case file. The dDPM sub-lists provide a content-oriented substitute for (activity-oriented) nested sub-processes.

In retrospective, we already fragmented the overall breast cancer case from the very beginning: into "pre-therapeutic diagnostics", "primary therapy", and "adjuvant therapy". This might have seemed artificial. Indeed, it would be perfectly suitable to use the same distributed case file for the entire breast cancer treatment. However, the terms are fixed domain expressions and the differentiation is conventional because there is a significant phase boundary between them. The characteristic trait is the changeover of responsibility from primary care to secondary care and back to primary care.

The changeover between ambulant and stationary care is reflected in an actor-related characteristic as well. The role of process coordinator is handed over from one actor to another. The process coordinator during the pre-therapeutic diagnostics has been the ambulant gynaecologist. A stationary gynaecologist has been involved in this episode but only in an assistant role. The process coordinator during primary therapy has been a (presumably different) stationary gynaecologist. The assistant participants for pre-therapeutic diagnostics do not necessarily re-appear during primary therapy. Finally, for adjuvant therapy, the coordinator role returns to the original ambulant gynaecologist.

As it has been said before, it should not be necessary to isolate a case into different episodes just because such phase boundaries can be conceived. The knowledge workers should be the only authorities to decide whether an overall case should actually be handled fragmentary by means of multiple episodes. In conclusion, a distributed case file must possibly handle cases with changing process coordinators. Thus, process roles should be supported in complement to medical roles. Process roles like the coordinator role require support for handing over a role between participants like a token.

5.4.3 Process Model Requirements

As before, *Core Requirements* (CR#) and *Extended Requirements* (XR#) are listed. Sub-lists are optional because they can be substituted by multiple isolated episodes and process role labels are not necessarily accustomed domain concepts.

[**CR11**] **Case Episodes as Process Scopes** – The scope of a case episode defines the scope of remote data synchronization. Each scope implies a single distributed case file. The episode is the primary unit for process scopes.

[**XR12**] **Episode Sub-Lists** – Within an episode content units may be grouped into sub-lists. Sub-lists may be nested. They are the content-oriented equivalent to nested sub-processes.

[**XR13**] **Process Responsibility Roles** – Several participants may have special process responsibilities. These could be articulated in form of process role labels, like coordinator. Some of the labels may be handed over like tokens.

5.5 Process Analysis: User-Defined Indicators and Process Templates

This section provides an example for user-defined status attributes and their utilization during treatment episodes. For this purpose, the description of breast cancer treatment is extended its last episode, the post-operative care. Post-operative care will continue for about five years.

The following use case illustrates how aggravation of a patient's condition can be indicated by status attributes. The indication will spontaneously change the course of the treatment and requires participation of additional healthcare professionals. For illustration purposes, the exemplary classifier *condition indicator* might be of use in situations where patients are under periodic medical examination. Consensus finding about the value range must happen outside a software system (the process platform can only foster consensus by supporting ad hoc definitions of classifiers as well as supporting ad hoc changes to the value range by the actors at any time). For the sake of the use case example, the process participants already have a consensus and we assume that they agreed upon a value range of normal, guarded, and serious for the condition levels. Such status can be attributed to any report and indicates the patient condition at the corresponding time and concerning the diagnostic context.

5.5.1 Breast Cancer Episode: Post-Operative Care

If no health problems arise, the post-operative care will follow a common schema, which is illustrated in figure 5.6. Every three months the patient must undergo a clinical examination at her gynaecologist (Gyn^A). Semi-annually she is referred to a radiologist (Rad^A) for a mammography (RV_M). Gyn^A supplies a detailed anamnesis documentation to briefly summarize the preceding treatment. After each examination the radiologist creates a report about the diagnostic findings and makes it available to Gyn^A.

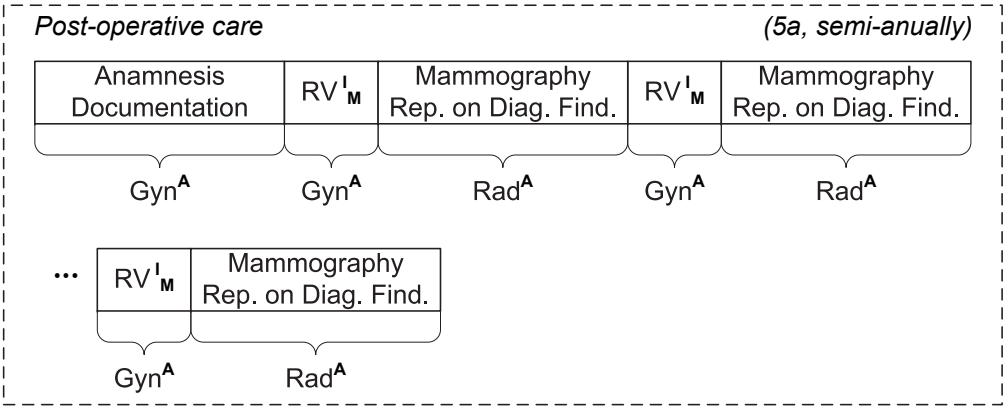

Figure 5.6: Breast cancer: post-operative care episode; no unclear symptoms

Because this is a periodic monitoring, the doctors want to indicate normal and exceptional conditions. Any time during the five years of post-operative care there is the possibility that the patient reports unclear symptoms or her gynaecologist makes a suspicious finding that indicates metastases. Thus, the *condition indicator* is designated as a diagnostic report status attribute.

An Incidence Occurs

If, for example, the patient at some point complains about pain in her upper abdomen and/or a yellowish pigmentation of her skin, the gynaecologist must find the reason for these symptoms as they may be caused by liver metastases. Figure 5.7 illustrates the modified episode. The gynaecologist creates an exceptional anamnesis report. He sets the *condition indicator* of the anamnesis report to guarded. Then he refers the patient to an internist (Int^A) for an upper abdomen sonography (RV_{AS}).

The internist might conclude in his report on diagnostic findings that the occurred symptoms are caused by a gallstone. In this case, the *condition indicator* of the sonography would also be set to guarded because the participants' consensus is that higher escalations

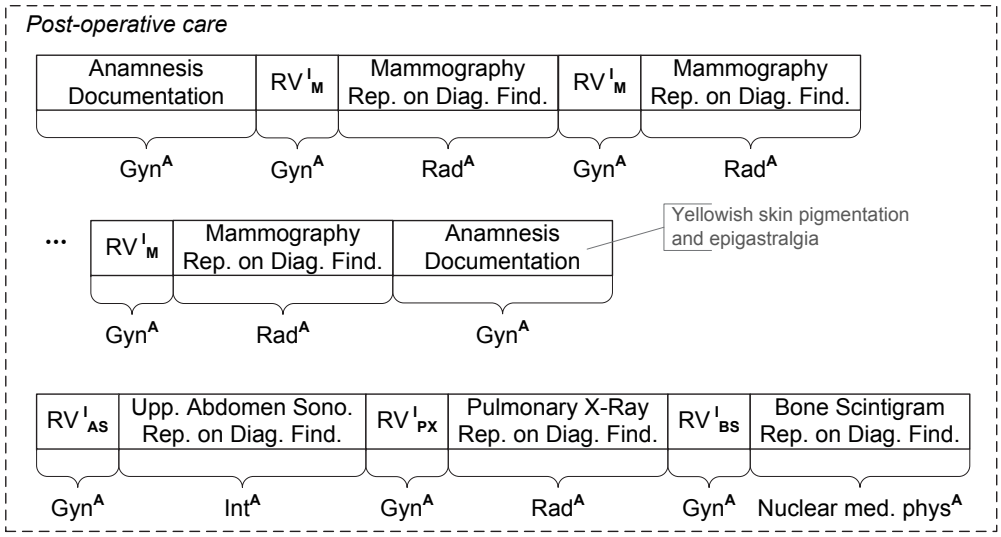

Figure 5.7: Breast cancer: post-operative care episode; classification of unclear symptoms

are reserved for metastases. Of course, the patient is treated by the internist against gallstone but this forms another treatment episode.

For another patient, the initial suspicion could be strengthened by the upper abdomen sonography and liver metastases are now indicated. Consequently, the internist sets the *condition indicator* of his report to serious. The gynaecologist will then instruct further examinations for potential lung or bone metastases: he refers the patient to a radiologist (Rad^A) for a pulmonary X-ray to check for lung metastases (RV_{PX}). A report on the X-ray results is written. The *condition indicator* would indicate the condition based on the X-ray, indicating exceptional lung condition with normal ("without pathological findings") to serious ("lung metastases"). In parallel, a referral to a nuclear medical physician takes place, in order to conduct a bone scintigram in search for any signs of bone-related metastases (RV_{BS}).

For breast cancer, any suspicion of metastases in one of the domains will always trigger the referral to both other domains (in the ternary set of liver, lung, and bones). Any affirmed suspicion (i.e. indicator value serious) will trigger a vital treatment. Treating the metastases will form an episode on its own. It will require a breast cancer centre, an oncologist, and further surgical or chemotherapeutic measures.

Benefits of Status Indicators

As far as described above, the user-defined attributes record process-relevant states of the underlying reports. The status attributes are like Scrum adornments and can be used for visualization purposes, like content unit colourization and marker labelling.

Another benefit of status indicators is the possibility to use them as triggers for automated coordination actions. Possible actions relate to third-party notifications or the semi-automation of process planning. Within the scenario, a modification of the *condition indicator* adornment into a serious state could trigger special notifications, e.g., notify epidemiological cancer registries, which form a hierarchical national organization in Germany and complement the German cancer treatment centres. It should even be possible to offer users some means to define process templates for an escalation process plan. In case of an abnormal condition indication, a process support engine that is embedded within the distributed case file could automatically extend the episode's process structure with the process steps from the escalation template.

Further Adornment Example & Consensus Scopes

Another adornment could be *diagnosis certainty* with exemplary levels from absolute and high over moderate to low. In some situations, it may not be feasible for physicians to make an authoritative diagnosis. Cooperative treatments of unclear symptoms or multimorbid patients require an intensified exchange of expert opinions. To indicate a limited certainty provides new participants with orientation while they gain an overview of the shared files.

Following the initial example of pre-therapeutic diagnostics from section 5.2.1, the gynaecologist creates a *diagnosis certainty* attribute for his initial report and sets the certainty of his own report to low. The radiologist later on provides a report on mammography and sets the certainty to moderate or high, according to the BI-RADS indicator of the mammography. Finally, the pathologist contributes his diagnosis based on the biopsy with an authoritative certainty, so he sets the indicator to absolute.

Even if it seems possible to specify such adornments at design-time, there will always exist various conceptions of indicators both in name and value range. Thus, consensus finding may either be done ad hoc during an episode or it may be provided by an institutional standard or a domain standard. An example for an indicator that is standardized for a domain is the BI-RADS score factor for mammographies as mentioned above. It would be perfectly conceivable, if users decide that they want the BI-RADS value directly available

as a status attribute for mammography reports in breast cancer episodes. The document-oriented process platform should allow for different consensus scopes and distinguish episode-, institution-, or domain-specific indicators.

5.5.2 Implications on Process Conception

The appeal of an adaptive attribute metadata model is its capability to track process status attributes of content units by means of the end-users at run-time. The general system architecture shall enable the users themselves to adapt adornments according to their demands at run-time. We need adornments, in addition to the payload documents, because we allow arbitrary payload file formats. The motive behind augmenting payloads with descriptors/adornments is to avoid upfront system integration efforts.

User-defined content unit attributes can also be considered as post-it notes that are being stuck to a document. There are formal ones like a BI-RADS value for mammography reports, or there could be required free-text adornments for arbitrary annotations as have occurred in circulation scenarios (cf. sect. 3.3.4). Even if the information that is stuck to a document with a post-it could also be provided by writing it inside the document, there will be a subtle difference. Changing the document creates versions of the document. Changing status attributes, like post-its, virtually creates versions of a discrete "cover sheet", more formally called a descriptor that collects all status attributes. The overview of all adornments of a content unit should be possible with a glimpse. In contrast, changing the document version requires the team members to re-assess the content. A general necessity for versioning remains and will be discussed further in the next section. In conclusion, the users ultimately decide if the efforts to maintain a specific status attribute gains any benefit for cooperation. The use case scenario motivates domain-specific status attributes whose exact specification cannot or should not be fixed at the design-time of a distributed process infrastructure because they ultimately are subject to semantic consensus finding between actors, institutions, and domains.

The creation of user-defined content unit attributes must be eased. It would not be suitable if adornments need to be constructed for each card individually. A form of template for adornments is necessary within each case file. The template for an adornment defines, for example, the name, the possible value range, and the default value. Thus, for newly created cards any adornments that are wanted can be selected from the template. Over a period of time, the desirable value range may change if a refined consensus has emerged. Any adaptation of the template should make the changes available for any card that already uses the adopted kind of adornment. Thus the template is not simply copied. The template is in fact a prototype from which the cards clone their adornments

as in prototype-based programming. In conclusion, it is called the *adornment prototype* and created cards selectively inherit adornments from it.

Another type of template may be auxiliary for process planning. During the discussion about benefits, templates for escalation plans have been mentioned. Needlessly to say, *process templates* are convenient not only for dedicated escalations but in general, for example, at case file initiation. As we have seen by the example of all the previous sections, for certain types of diseases some fundamental process steps can be assumed to occur. A process template is like a checklist, it is primarily a list of work items. In the context of result-oriented work item conception, it is a list of card descriptors. An instance-based creation of process templates should be possible by supporting an export of a process template from any case file. Instance-based creation means that the case is stripped from instance-related data like patient information, actor information, or content states. Thus, a schematizing of medical processes can be achieved without efforts by filtering case instances.

For the sake of completeness, it should be mentioned that process templates and cases are not in a prototype-oriented inheritance relationship. Process template are not required as active documents but are exported as simple data files. Thus, process templates are just copied into (or appended to) the work-list of a case. However, there is no special editor necessary for process templates because the case editor can be used. Importing a process template into an empty case is again a fully-qualified case that can be edited. Thus, work-list editing facilities for case files can be used to alter the template elements. Re-exporting the template case into a template file is easy because the filter configuration can be economized by filter_nothing, which is a full export.

The automatic reaction on state changes of user-defined adornments is eligible. However, the user itself would have to extend the case configuration with rules provide state conditions and triggered actions. A rule-based editor would be necessary. For easing the specification of user-defined actions an action library could be supported. The actions would have to be capable to have parameters assigned.

5.5.3 Process Model Requirements

As before, *Core Requirements* (CR#) and *Extended Requirements* (XR#) are listed. Only the basic support for user-defined indicators is a mandatory requirement. All other derived process model requirements are optional because these concepts not necessarily reflect accustomed working practice.

[CR14] **User-Defined Indicators and Annotations** – Support is needed for ad hoc definitions of classifiers as well as supporting ad hoc changes to the value range by the actors at any time.

[XR15] **Adornment Prototype** – User-defined adornments should be re-usable for different cards within a case episode. A case-embedded template for adornments may be provided. Prototype-based programming semantics apply because cards are instantiated from the template and evolutionary adaptations should be propagated.

[XR16] **Consensus Scopes** – The definition of each adornment should indicate on which organizational scope a consensus about its value range exists. The distinguishable scopes are episode-, institution-, or domain-specific consensus scope.

[XR17] **Process Templates** – The abstraction of carried out process steps from a case should be re-usable for other cases. The export and import of process templates from and into a case should be provided.

[XR18] **User-Defined Rules and Actions (on User-Defined Adornments)** – Automation facilities for reactions on any state change of user-defined adornments may be provided. Allowing end-users to manage such reactions would require a graphical rule-editor and a library of configurable actions.

5.6 Process Analysis: Termination Criteria and Content Versioning

This section discusses criteria for completion of work items. Preliminary versions of cards are motivated. Predefined card indications are derived, being separated into visibility, validity, and versioning. Card progression states that have been informally introduced in section 5.2 are formally related to visibility and validity. Finally, termination criteria for case episodes are considered and similarly applied to sub-lists.

5.6.1 Versions of Reports and Progression of Work

Result reports sometimes take months to be finally released in an authorized and signed form. This is a well-known fact from clinical working practice (e.g., [306]). The problem is a notoriously overburdened workload of clinical physicians. In many cases, the basic findings would already be available in the local EMR and could be shared. Yet, writing

discharge letters requires to aggregate the findings and to articulate an interpretation and summarisation, which consumes time. Notably, the physician approves the validity of the discharge letter with his or her signature and upon releasing a signed discharge letter he or she is liable to its content. However, in the absence of an officially released discharge letter and on request from a co-treating physician, physicians are willing to provide preliminary information to their peers as a matter of course. Preliminary versions are invalid in terms of "not signed off" and their content should only be consumed if treated with discreet caution.

In order to support preliminary versions it is necessary to consider the visibility and the validity of a content unit separately. In traditional database-centric approaches, visibility is strictly coupled to validity. Information is only visible if it is committed, and the commit has to ensure the integrity constraints. Most content-oriented approaches inherit a database-centric perspective and none of these approaches distinguishes between visibility and validity. Decoupling validity and visibility in databases had been subject to *database conversions* by Kirsche (e.g., [307, 308]), still, database conversions are not natively supported by database systems.

The validity serves the same function as a paper-based signature. Thus, validity usually implies acceptability instead of formal correctness. Providing electronic signatures for valid documents is subject to the local healthcare information systems. From the perspective of process support for inter-institutional cooperation, validity is a boolean indicator. Thus, the *validity model* essentially consists of the classifiers invalid and valid. As a complement, the *visibility model* consists of the classifiers private and public.

A private-invalid card is a *draft* for non-collaborative purposes or to prepare future collaborative efforts. A private-valid card is readily *prepared* for public releasing. In contrast, public-invalid cards are *preliminary* information. A public-valid card is properly *released* information. If a public-valid content has been released, it is not allowed to change it without versioning because the individual systems require a global version for the tracking of changes. Any other content unit, in terms of visibility and validity, are equally allowed to use versioning as it seems appropriate to its human owner. In fact, for data provenance purposes it wise to apply versioning on all publicly shared data, i.e. also for preliminary versions.

The termination of the work item that is articulated by the card is generally undecidable. The life-cycle of a card is open-ended because new versions are allowed to refine released versions. However, the termination can be defined as a threshold on the progression state. The progression state that is necessary to consider the work item as *completed* can vary. The dDPM default is that work item fulfilment is synonymic to a released card progression state. Table 5.1 provides an overview on the content states in dependence of visibility and validity.

Content exists?	Visibility	Validity		Card Progression State	Work Item Fulfilment
no	-	-	⇒	**empty**	unstarted
yes	private	invalid	⇒	**draft**	ongoing
yes	private	valid	⇒	**prepared**	ongoing
yes	public	invalid	⇒	**preliminary**	ongoing
yes	public	valid	⇒	**released**	completed

Table 5.1: Visibility and validity in relationship to card progression as well as work item fulfilment

A basic question for the dDPM configuration is whether visibility and validity can decreased. If arbitrary changes are allowed, then card progression can be cyclic. By default, dDPM applies two additional rules: 1) "once public always public" and 2) "once valid always valid". These rules guarantee acyclic card progression.

Again, it should be repeated that content validity is not any kind of formal correctness criterion in this context that could be automated, neither syntactically nor semantically. Validity merely indicates that the contributor takes definite responsibility for the content. In [55], I denominated this as *intend validity*. This is significant because versioning provides another notion of validity: the existence of a newer version ends the validity of any previous version. In [55], I denominated this as *technical validity*. The technical validity is *implicit* and completely represented by the content unit version history. As adornment, only the humanly determined and manually indicated intend validity is *explicit* part of the content status model.

5.6.2 Completion of Case Episodes

The case episodes are by default open-ended. The work-list can potentially be extended any time. However, we can identify one implicit interim termination criterion and two kinds of explicit termination indication. The following list describes possible termination states of a work-list that constitute an extended model for case termination.

open-ended The case file has uncompleted work items. The work-list could be extended any time.

ceased All work items of the case file have been completed but the work-list could still be extended any time.

sealed The case file has been explicitly marked as sealed. Until the seal prevails no changes are allowed. However, it is possible to explicitly break the seal and to continue work on the case.

closed The case file has been explicitly marked as closed. It is a permanent state that cannot be revoked. A continuation requires the creation of a new case episode.

Completion of Sub-Lists

If sub-lists are used to sub-structure case episodes, the same rationale applies. However, a non-revocable termination indication seems unnecessary because sub-lists are not archived by its own. As long as the case is not irrevocably closed any sub-lists might still be altered. The following list describes possible termination states of a sub-list.

open-ended The case file has uncompleted work items. The work-list could be extended any time.

ceased All work items of the sub-list have been completed but the sub-list could still be extended any time.

completed The sub-list has been explicitly marked as completed. Until the the marker prevails no changes are allowed. (It is possible to undo sub-list completion and to continue work.)

5.6.3 Implications

Visibility and validity can be considered as predefined platform adornments. Changing both adornments allows controlling publication independent of validity and, thus, allows for preliminary versions. The card progression and work item fulfilment can be derived from the visibility and validity via basic rules.

The version history of each content unit must be available independently for data provenance purposes. Hence, each content unit requires its independent Version Control System (VCS) history. However, the overall team progress, i.e. data production over the complete case file, must also remain trackable. This is approximately the same versioning situation as in parallel software development of a modular system.

The implicit case and sub-list termination criteria are derived card progression states. A rule-based monitoring can be used to automatize the state transition from open-ended into ceased state for cases and for sub-lists.

5.6.4 Process Model Requirements

As before, *Core Requirements* (CR#) and *Extended Requirements* (XR#) are listed. The concepts for case and sub-list termination are optional because they do not necessarily reflect accustomed working practice.

[CR19] **Separation of Visibility and Validity** – In order to support the publication of preliminary content versions it must be possible to control visibility and validity.

[CR20] **Versioning and Process History** – Versioning of content units is mandatory for data provenance purposes. The process history of the overall case episode is reflected by the versioning history of a content-oriented workflow model and its content units.

[CR21] **Open-Ended Case Termination** – The core assumption about case termination is that an end might never occur. Ceased work by interim completion of all planned work items is deceptive. In healthcare, changes of patient conditions occur emergent and additional measures can be required any time.

[XR22] **Case Sealing and Closing** – An explicit indication of case termination might be provided by sealing and closing markers. Explicit termination blocks alteration of the case file.

[XR23] **Sub-List Ceasing and Completion** – Termination of sub-lists follows the rationale of case termination. Work on sub-lists might cease by interim completion of all its work items. Explicit completion markers might be used to indicate termination and to block sub-list alterations.

5.7 Process Characteristics

The general process conception of dDPM is intended to be independent of an implementation approach. The process characteristics of dDPM summarize requirements for inter-institutional and case-driven processes. The healthcare use cases that have been described in sections 5.2 to 5.6 have refined and exemplified several process model requirements. The overall process conception will also be discussed in relation to the characteristics that have been derived from content-oriented workflow approaches.

5.7.1 Consolidated Overview of Process Model Requirements

This section provides a consolidated overview of the analysed process characteristics. These characteristics become requirements for any content-oriented process model that is designed for supporting inter-institutional and case-driven processes. The universal process characteristics that have been discussed in section 5.1 are the baseline. The healthcare use cases from sections 5.2 to 5.6 analysed the process from a content-oriented perspective. Thus, the process model requirements of dDPM that have been derived from

the process analysis can be considered as a content-oriented refinement of the universal characteristics. Table 5.2 provides a consolidated overview of the process conception of dDPM in form of the universal process characteristics as well as core and extended process model requirements for a content-oriented workflow approach.

	Universal Inter-Institutional Process Characteristics
	Site-Spanning Processes
	Autonomous Participants
	Heterogeneous IT-Systems
	Decentralized Coordination
	Universal Case-Driven Process Characteristics
	Paper-Based Working Practice
	Dynamic Teams
	Knowledge-Driven Processes
	Team-Wide Information Demands
	Core Process Model Requirements of dDPM
CR1	Result-Oriented Work-List
CR2	Partial Results from Various Actors
CR3	Card Metaphor
CR4	Adornments as Card Progression States
CR5	Domain User Roles
CR6	Parallel Work, by Default
CR7	Tacit Order by Non-Binding Prioritization
CR10	Location-Independent Access
CR11	Case Episodes as Process Scopes
CR14	User-Defined Indicators and Annotations
CR19	Separation of Visibility and Validity
CR20	Versioning and Process History
CR21	Open-Ended Case Termination
	Extended Process Model Requirements of dDPM
XR8	Cohesive-Content Relationships
XR9	Required-Content Dependencies
XR12	Episode Sub-Lists
XR13	Process Responsibility Roles
XR15	Adornment Prototype
XR16	Consensus Scopes
XR17	Process Templates
XR18	User-Defined Rules and Actions
XR22	Case Sealing and Closing
XR23	Sub-List Ceasing and Completion

Table 5.2: Survey of the process conception of dDPM in form of universal process characteristics as well as core and extended process model requirements

The purpose of the dDPM process conception is to be reference model. More specifically, it is a content-oriented reference model for inter-institutional and case-driven processes. The process conception is independent of an implementation. On that account, the process model requirements from sections 5.2 to 5.6 have been distinguished into core requirements and extended requirements. Only the core requirements must be implemented to count as a workflow approach to dDPM. The core requirements represent characteristics that directly reflect accustomed working practice. The extended process model requirements do not necessarily relate to accustomed domain concepts and do not necessarily reflect accustomed working practice. However, each extended requirement also denominates a reference concept for an advanced workflow aspect. The application domain might have an influence whether support for any advanced requirements is necessary or not. The current conception is influenced by healthcare environments.

As a form of disclaimer, I want to denote that the internal data model of the pilot implementation has been prepared with sub-lists in mind (→XR12/XR23) but the necessary graphical user-interface will not support sub-lists. Flat work-lists are supported, thus, case fragmentation is currently only supported by isolated episodes. Furthermore, the pilot implementation has been designed with rule-based actions on user-defined attributes in mind (→XR18) but neither exists an embedded rule-editor nor is a configurable action library provided.

5.7.2 Content-Oriented Characteristics

The process model of dDPM is a content-oriented conception of case-driven processes. Thus, the taxonomy of characteristics for content-oriented workflow approaches (cf. sect. 3.4.5, p. 127) can be applied to the process model of dDPM. The classification depends whether the process conception is used in its dDPM core variant or in its extended variant. The overview is illustrated in figure 5.8.

The core process model of dDPM is listed in figure 5.8 using a green background filling. First of all, it is a content-oriented workflow model with *simply typed elements*. The process elements are work items of a shared and synchronized work-list. The shared work-list can be edited by any case participant. Each card that represents a task is created individually. Hence, the element instantiation is based on unique elements. There are rules for the content progression of individual cards (cf. tab. 5.1 in sect. 5.6.1) as well as indications for case termination (cf. sect. 5.6.2). On that account, content progression is articulated by *rule-based transitions*. According to the considerations of chapter 4, the overall case is cloned or moved between sites. Thus, there is also a form of content progression in terms of circulation. The cards as content units are accessible for every participant. Thus, a *perpetual accessibility* on available content is supported

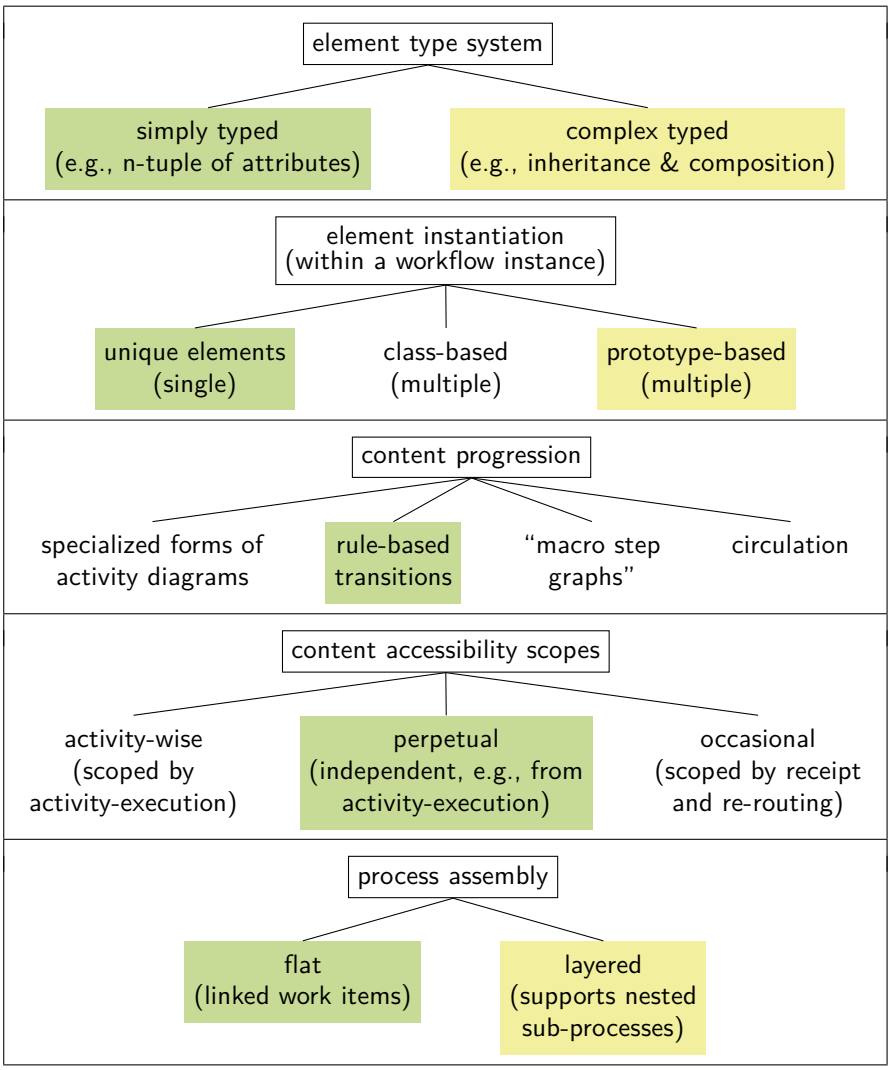

Figure 5.8: Content-oriented workflow characteristics of the dDPM approach

independent of activity execution. Finally, the work-list as a stack of cards is the process assembly. Flat process assemblies seem effectual for most inter-institutional cases.

The extended process model of dDPM is also classified in figure 5.8, which is highlighted with a yellow background filling. The characteristics element type system and element instantiation change because of the concept of adornment prototype (→XR13). The process assembly changes because of sub-lists (→XR10). The adornment prototype is a template for instantiating card descriptors. Furthermore, the card descriptors remain in a prototype-based inheritance relationship with the adornment prototype. Thus, an extended dDPM element instantiation may be *prototype-based* and an extended dDPM element type system may be *complex typed* with inheritance facilities. The sub-

lists concept provides a nested sub-structuring of the initially flat work-list. Thus, an extended dDPM process assembly may support *layered* work-list hierarchies.

5.8 An Ideal Implementation of dDPM in a Pre-Integrated System Environment

At the core, the dDPM idea is to consider cooperation in healthcare just as one successively written report that is subdivided into the report contributions of the several participants. If the vision of Electronical Health Record (EHR) would be realized, process support could rely on a perfectly pre-integrated system environment. In such a system environment, the content-oriented workflow could be articulated by filling in forms sections by various participants. Of course, new patient-related information demand would lead to a successive extension of the case form.

The ideal implementation of dDPM can be illustrated by first breast cancer episode, the pre-therapeutic diagnostics. The scenario still begins with a patient who visits her gynaecologist. The gynaecologist creates a case form, which initially allows to fill-in patient information and to document the anamnesis. The case form is integrated into the local information system. Thus, the patient information is technologically readily available from the system context of the patient visit and is pre-filled into the case form. The gynaecologist will first make a clinical examination, which is a palpation of the breast. He wants to document the result. Frequently used form sections must easily be available in form of form templates that can be placed into new forms. Thus, the gynaecologist adds a section for the clinical examination from a form template repository. The palpation indicates a suspicion of breast cancer, thus, he will conduct a sonography. The necessary form section to document a sonography is again copied from the repository and filled out. Figure 5.9 illustrates the case form that is structured into sections.

A referral to a radiologist is conducted just by creating a referral voucher section. The case is (somehow) shared in the nation-wide EHR system and will be accessible by other physicians that will participate in the case. Thus, the patient visits a radiologist and the radiologist can extend the case form with a section that documents the results from the mammography. If the patient returns to her gynaecologist, he can access the extended case form. Based on the diagnostic BI-RADS indication the case will unfold.

Notably, a section appears that has not been shared via the distributed case file. The according section is the one for the clinical examination. It is not necessarily shared inter-institutionally because it is superseded by the sonography result. An inter-institutional case file contains only digital equivalents of paper-based working practice. In contrast,

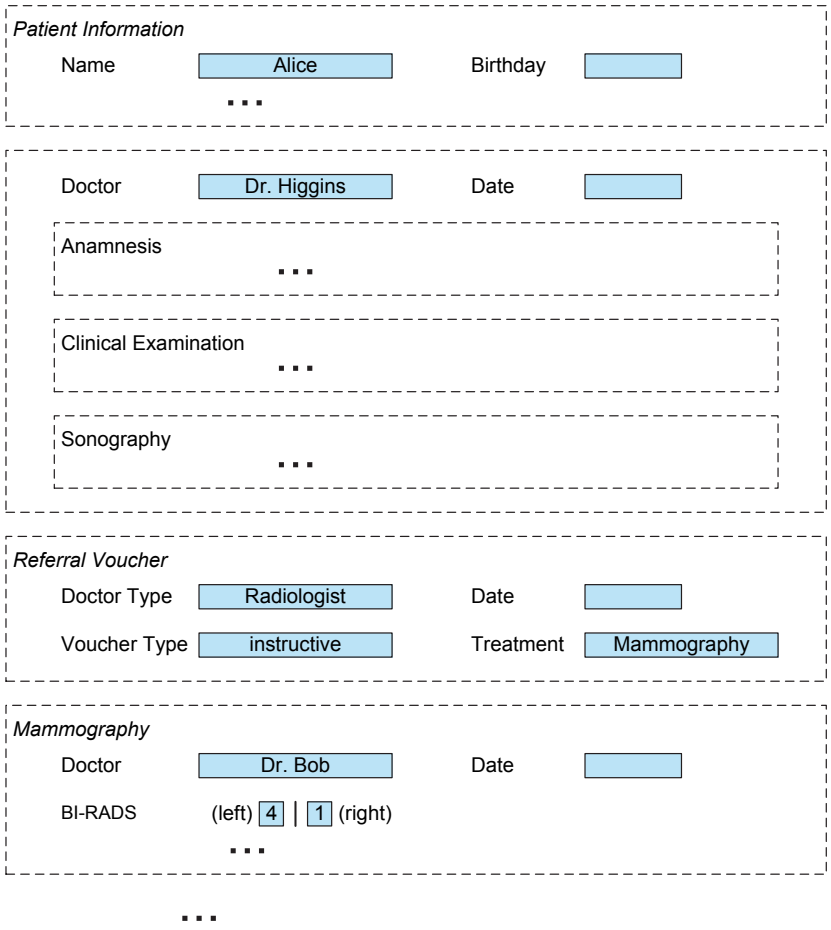

Figure 5.9: A single-form implementation of dDPM in a pre-integrated EHR system environment

the ideal dDPM implementation could contain all available information from the local patient files of the EMRs. The EHR vision establishes a federated continuum of the local information systems.

The dDPM card metaphor does nothing else than to provide a paper-based concept for the various form sections. Currently, there is no large-scale telematics infrastructure that allows synchronizing case forms between institutions. Thus, the dDPM α-*Doc* provides case-individual synchronization scopes. All arguments for document-oriented integration in section 2.2.3, for Deferred System Design (DSD) in section 2.2.6, and for the separation of concerns that has distinguished between content, decision support, and coordination in section 2.2.10 ultimately shift/defer the responsibility for the underlying integration problems. There is no universally accepted content format even if Health Level 7 (HL7) Clinical Document Architecture (CDA) provides a promising candidate. If all participating systems could understand CDA content then the dDPM cards can be

CDA-based forms. However, as long as a pre-integrated system environment cannot be assumed the decision which content formats to be used must be allowed to be taken by the participants at run-time (cf. sect. 2.2.6).

The EHR-based single-form approach to dDPM could measure content progression in availability of form field values. Then again, it would also be possible to integrate explicit progression status attributes within the form sections. In an environment that allows arbitrary payload file formats, we need to manage an external descriptor for each card that stores its content progression status attributes in addition to the payload documents. The term "adornments" for the attributes is just convenient to indicate that the attributes externally augment its card content and that a visual representation for the attributes is favoured.

In conclusion, the "stack of cards" case file structure and the "cards represent tasks" conception to articulate information demands are the inter-institutional equivalent to a successively written report. All considerations in form of the process model requirements of dDPM would be necessary and valid for the ideal EHR-based approach as well: parallel work, tacit order, user-defined indicators, visibility vs. validity, versioning, explicit data dependencies, sub-lists, process roles, adornment prototypes, consensus scopes, process templates, and termination criteria.

5.9 Summary

This chapter has described the process conception of the dDPM approach. The universal process characteristics of inter-institutional and case-driven processes have been analysed. Subsequently, a process analysis for a distributed healthcare cooperation has been conducted based on the example of breast cancer treatment. All breast cancer treatment episodes have been illustrated from pre-therapeutic diagnostics to post-operative care. Each episode has provided a use case scenario that has successively motivated various process model requirements.

The process model requirements from the healthcare use cases represent a refined and content-oriented conception for inter-institutional and case-driven processes. Accordingly, a survey section has consolidated the process characteristics of dDPM in an overview, distinguishing between core conceptions and extended conceptions. The resulting process conception has then been classified using the content-oriented workflow characteristics from the conclusion of section 3.4.

Finally, an ideal implementation of dDPM in a perfectly pre-integrated EHR system environment has been outlined as a reference model. The limitations that stem from

unsolved system integration had been reflected in the user story from chapter 4. Thus, unsolved integration motivates the implementation of dDPM as a content-oriented process model in form of active documents.

III

Pilot Implementation

6 | The α-Flow Approach

> “Show me how you build and
> I will tell you who you are.”
>
> (Christian Morgenstern)

The *α-Flow* approach provides an implementation of distributed Document-oriented Process Management (dDPM). The first part of this chapter describes the workflow model of *α-Flow*, which is derived from the dDPM conception. A meta-model of the *α-Flow* elements is provided and the distributed case file artefact is formalized. The predefined set of adornments is described and it is explained how adornment changes become triggers for rule-based actions. Finally, an architectural overview of the *α-Flow* system is provided.

6.1 The α-Flow Model

The dDPM conception requires ad hoc process support without needing to install a workflow engine prior to process participation. Thus, the implementation of *α-Flow* is based on the concept of active documents, which has been discussed in sections 2.2.12 and 3.5. The *α-Doc* is the primary *α-Flow* artefact. The symbol “*α-Doc*” is synonymous to “active document”. Each *α-Doc* is the distributed case file for a dDPM case episode.

6.1.1 From dDPM Concepts to α-Flow Elements

The dDPM concepts like case episode, card, and adornment are directly represented in *α-Flow*. However, the “*α*”-prefix is applied to the dDPM concepts in order to have an unambiguous correlation between the *α-Flow* approach and its elements. As an initial overview, the dDPM diagram of the primary therapy of breast cancer treatment is taken and the primary concepts are annotated with the respective *α-Flow* elements. The illustration is provided in figure 6.1.

A case episode whose handling is supported by an *α-Doc* is accordingly referred to as *α-Episode*. There is a one-to-one relation between *α-Doc* and *α-Episode*, thus, the

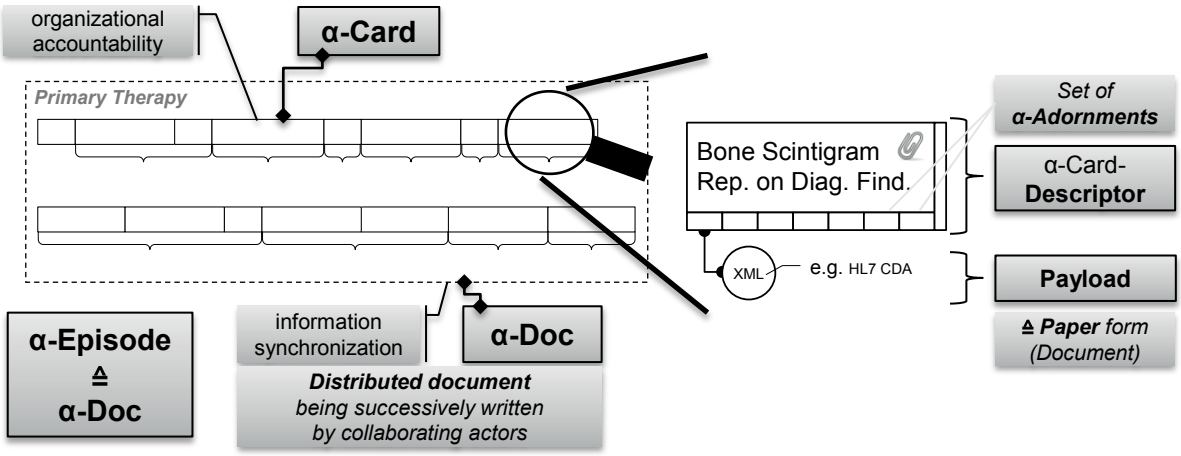

Figure 6.1: The α-*Flow* concepts in the context of the primary therapy of breast cancer treatment

concepts α-*Doc* and α-*Episode* are equivalent, from a certain perspective. The notion of an α-*Doc* accentuates the content-oriented dimension, i.e. the case file concept. The notion of an α-*Episode* accentuates the implied workflow dimension, i.e. the tacitly underlying activities.

The active properties of an α-*Doc* implement an embedded α-*Flow* engine that enacts the content-oriented process model of dDPM. In general, each active document in α-*Flow* carries the workflow context in addition to the medical content. The dDPM cards are shared and managed as content units in the context of an α-*Doc* as α-*Card*s. The dDPM adornments are process-related markers on the cards. Accordingly, each α-*Card* is attributed with α-*Adornment*s.

The α-*Doc* as a molecular file is internally decomposed into α-*Card*s. It provides the content scope for remote synchronization. The α-*Card*s are units of organizational accountability. Each of them is a unit of validation as well as subject to atomic synchronization actions. The α-*Adornment*s for each α-*Card* constitute its *descriptor*. The cards represent tasks. The results of the tasks are contributed to the case file. Accordingly, each α-*Card* has a *payload* as attachment. An unstarted work item (cf. sect. 5.6.1, tab. 5.1, p. 183) is equivalent to an α-*Card* for that only the descriptor exists without an attached payload.

In α-*Flow*, information units are distinguished into two categories: content cards and coordination cards. The content cards are essentially the content units from the dDPM use case scenario. In addition, coordination cards are utilized to store process structure and collaboration resources. Coordination cards are independent of the local application systems and belong to the distributed case handling.

Distributed case files require information synchronization between the participants. From a file replication and messaging perspective, the *α-Doc* is logically centralized but physically distributed. That means that each replicate of an *α-Doc* autonomously uses an embedded store-and-forward messaging for distributed synchronization of the replicates. An illustration of the distributed scenario with remote *α-Doc* replicates is outlined in figure 6.2.

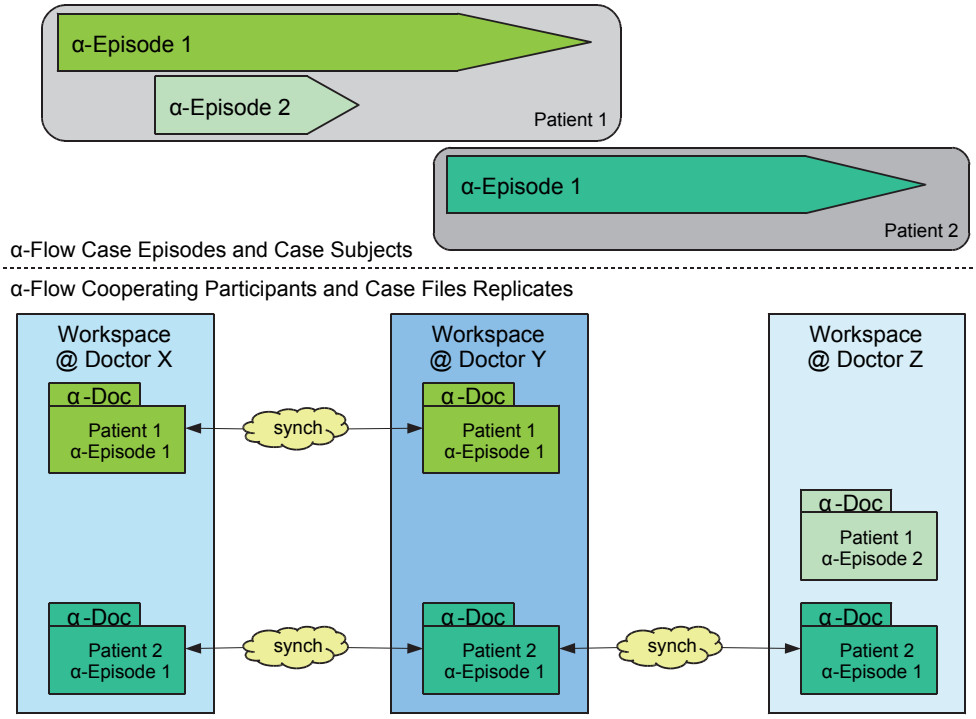

Figure 6.2: Distributed *α-Flow* scenario: *α-Episode*s and *α-Doc* replicates

The messaging infrastructure is ultimately intended to build upon secure large-scale messaging platforms with guaranteed delivery, as it is intended in Germany by the national government project "Elektronische Gesundheitskarte" (eGK). The *α-Flow* engine implements a synchronization protocol. The protocol uses electronic post-box information, which is part of the recorded participant information. One appeal of establishing a peer-to-peer synchronized *α-Doc* for each individual case is that each physician gains only access to exactly the same electronic files as they are already accessible to him or her in paper-based working practice, today.

6.1.2 The Workflow Language

The *α-Flow* approach provides a basic workflow language. Traditional workflow language elements (cf. sect. 2.2.5) are provided in a document-oriented style. The language-logical

terms of the previous section are systematically extracted. The following list describes all *α-Flow* elements. Elements that are used within the description of another element but are defined at a later position are marked with "→", just as a kind of reading hint.

***α-Episode*:** The *α-Episode* is an inter-institutional process that is handled as a case. Each *α-Episode* has its own case file.

***α-Doc*:** The *α-Doc* is a case file in form of an active document. It contains →*α-Card*s (passive) and the →*α-Flow engine* (active).

***α-Flow Engine*:** The *α-Flow* engine is the case handling application that is embedded within an *α-Doc*. The *α-Flow* engine manages the storage of →*α-Card*s, the synchronization between remote →*α-Doc replicates*, and the editing of →*coordination cards*, for example, the shared work-list.

***α-Card*:** The *α-Card* is the composite of a →*descriptor* and a →*payload*. *α-Card*s are distinguished into →*content cards* and →*coordination cards*. From a storage perspective, the *α-Card* is an abstract concept of no physical existence on its own. From a logical perspective, a unique key is generated to identify each *α-Card* and the *α-Card* identifier is shared by its descriptor and payload.

Content Card: Each content card is an *α-Card* that captures a work-item (prospectively) in form of a →*descriptor*. The result of the work-item (retrospectively) becomes also part of the *α-Card* in form of its →*payload*. The payloads of content cards belong to the software applications of the domain, which export them as contributions to the case file.

Coordination Card: Each coordination card captures a coordination aspect of the inter-institutional process. For example, coordination cards conduct information about the work-list as a process structure or information about →*contributors*, →*roles*, and →*institutions* or →*electronic post-box* information for each actor. The coordination cards are independent of the domain applications, they belong to the *α-Flow* engine. Currently, there are three coordination cards: →PSA, →CRA, and →APA

Descriptor: The descriptor captures all process-related status information of an *α-Card* in form of a set of →*α-Adornment*s. The set of *α-Adornment*s for each *α-Card* can be changed at run-time. Each descriptor is an independent electronic document.

***α-Adornment*:** The *α-Adornment* captures a single process-relevant status attribute of an *α-Card*.

Payload: The payload is the result of a work-item in form of an electronic document. It is (virtually) attached to an *α-Card* (descriptor).

Content Dependency: The content dependency is an association between α-*Card*s. It captures dDPM concepts like the cohesive-content relationship or the required-content dependency.

Object under Consideration (OC): The OC is the subject-matter of the inter-institutional process and respective case file. In healthcare, the OC is the patient.

Contributor: The contributor is the unique name of a human actor that is responsible to accomplish a work item. The contributor provides the result of a work-item in form of the payload of an α-*Card*.

Institution: The institution is the unique name of an organizational location of a contributor.

Role: The role is a name that provides an abstract description of the features that are required for accomplishing a work item. In the end, contributors take the responsibility for the realization of whatever the roles promise.

Electronic Post-Box: The electronic post-box is required for messaging purposes. Each contributor must own an electronic post-box.

***α-Doc Replicate*:** The α-*Doc* replicate is a single file system copy of an α-*Doc*. Additional human actors are invited for process participation by handing them over a new copy. The number of α-*Doc* replicates for an α-*Episode* is dynamically changing. The α-*Doc* replicates can be distinguished by their →node identifier.

Node: The physical location of an α-*Doc* replicate. Each node has a unique identifier.

Process Structure Artifact (PSA): The PSA captures the work-list. Thus, its payload contains the complete set of α-*Card* identifiers of an α-*Doc*. The PSA payload covers the order of the α-*Card*s, thus, capturing the prioritization of the work-items. The PSA payload also covers content dependencies.

Collaboration Resource Artifact (CRA): The CRA captures the resources that are necessary to start and complete work-items. Thus, its payload contains information about each contributor like the institution, role, and electronic post-box.

Adornment Prototype Artifact (APA): The APA captures the set of α-*Adornment*s that is used as a clonebase for creating new α-*Card* descriptors. Thus, its payload contains a data structure of α-*Adornment*s.

Corpus Genericus: The corpus genericus captures *α-Adornments* for that the *α-Flow* engine provides predefined functionalities[1]. It is a special set of *α-Adornments* that must exist in each descriptor. This is ensured by predefining them as a mandatory subset of the APA payload.

6.1.3 The Meta-Model

The *α-Flow* language elements can be set into relation to each other. Thus, figure 6.3 illustrates the *α-Flow* meta-model. The structural elements like the *α-Doc*, *α-Card*, descriptor, payload, and *α-Adornments* as well as the *α-Episode* identifier and the *α-Card* identifier have been highlighted in light blue. The *α-Card* is only an abstract concept, as has been outlined above. It captures a work-item both prospectively (planning purpose) and retrospectively (result documentation).

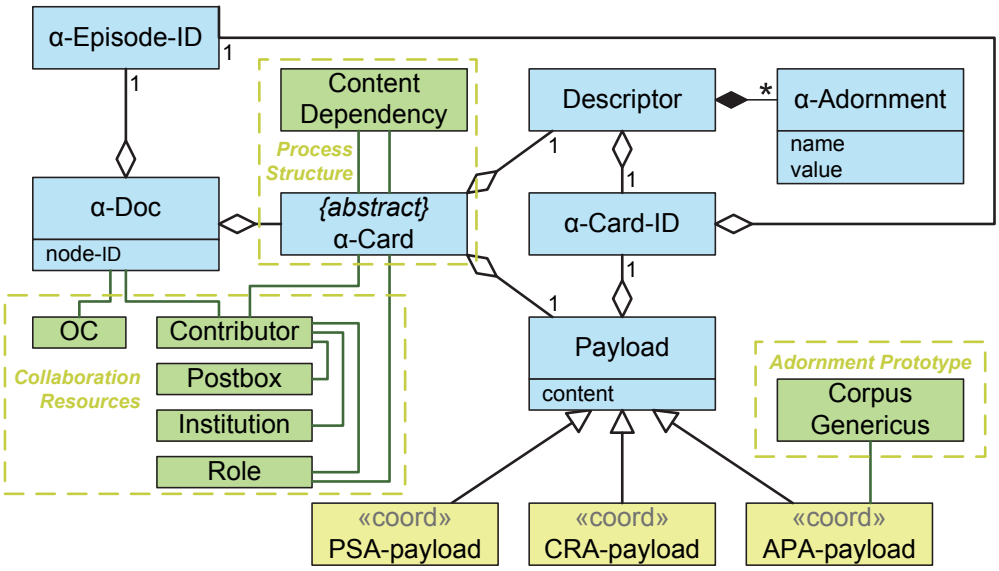

Figure 6.3: The *α-Flow* meta-model

Concepts like the OC, contributor, institution, and role are language-logically associated with the work-item, i.e. with the *α-Card*. Each contributor is also associated with an electronic post-box for messaging purposes. All these elements are considered as collaboration resources. From a data management perspective, they are kept in the CRA

1 The term "corpus genericus" is an artificial term. The "corpus" indicates a "principal part" and "genericus" is a pseudo-Latin construct that indicates the "generic framework layer". Originally it was a wordplay but the term stuck and was never replaced conceptually.

payload. Several α-*Adornment*s are used to associate each α-*Card* with the collaboration resources.

The situation is similar for content dependencies. The work-items are prioritized and can be associated with each other. Thus, the language-logical notation of the meta-model relates content dependencies with a pair of α-*Card*s. From a data management perspective, they are kept in the PSA payload.

The α-*Adornment*s that are predefined by the α-*Flow* framework will be discussed in sect. 6.1.5. These adornments are collectively called the "corpus genericus". As a fixed subset of the adornment prototype they become constituent parts of each α-*Card* descriptor. The adornments of the corpus genericus are not technical constructs but are of immediate interest for the user. For example, they provide the mentioned associations to the collaboration resource elements. From a language-logical perspective, the term "corpus genericus" is used primarily in association with the adornment prototype.

The logical elements for collaboration resources, process structure, and adornment prototype are highlighted in light green. The content artefacts that are used to keep the information are highlighted in light yellow. The purpose of the yellow dashed boxes for grouping the green elements is to provide a correlation to their respective coordination card artefact.

Finally, the existence and relevance of the coordination cards and their payloads is ultimately hidden by the α-*Flow* engine as a graphical case handling application. Thus, users do not necessarily need an explicit understanding of these cards. The reasoning behind this design is that the three coordination cards can technically be handled like the content cards, for versioning and remote synchronization purposes.

6.1.4 Model Formalization

The α-*Doc* (AD) is defined as a tuple that consists of a case episode identifier (e), a set of card identifier (C), a set of α-*Card* descriptors (D), and the set of payloads (P), an embedded case handling application (α), and an optional replicate identifier (r). The replicate identifier is derived from the physical node at which the α-*Doc* replicate is executed. It is allowed to be empty insofar as replicates are transported via a passive physical media like flash cards to new nodes, for example, as an invitation to new actors for participation. In the following, the sets *string* and *binary* represent the volume of all possible arrays of characters or bytes, respectively.

$$AD := (e, C, D, P, \alpha, r) \;:\; e \in string,\, r \in string \vee r = \emptyset$$

Each work item is captured in form of an *α-Card*. Thus, the cardinality (w) of the set of *α-Card* identifiers (C) is equal to the current given number of articulated work items. Each *α-Card* identifier (c_i) is a composite identifier that includes the episode identifier (e) and a surrogate (s_i) as discriminator for the card itself.

$$C := \{c_1, ..., c_i, ..., c_w\}$$
$$w := |C|$$
$$c_i := (e,s_i) \ : \ s_i \in string$$

For each *α-Card* identifier (c_i) exists an *α-Card* descriptor ($\hat{d}_i$). The cardinality of the set of *α-Card* descriptors (D) is equal to the number of articulated work items (w). Each *α-Card* descriptor ($\hat{d}_i$) is also known as the *physical descriptor* because it contains the identification, thus, it can independently exist. The (physical) descriptor is a composite of the *α-Card* identifier and a set of adornments (A_i).

$$D := \{\hat{d}_1, ..., \hat{d}_i, ..., \hat{d}_w\} \ : \ |D| = w$$
$$\hat{d}_i := (c_i,A_i) \ : \ c_i \in C$$

Each *α-Card*-related set of adornments (A_i) is also known as a *detached descriptor* because it is the physical *α-Card* descriptor without the identifying part. Each adornment (a^i_j) is a name-value pair (n,v). Each descriptor can have an individual number (z_i) of adornments.

$$A_i := \{a^i_1, ..., a^i_j, ..., a^i_{z_i}\}$$
$$z_i := |A_i|$$
$$a^i_j := (n,v) \ : \ n,v \in string$$

The set of payloads (P) contains the results of work items. There can exist only one payload for each *α-Card* descriptor. Each payload ($\hat{p}_i$) is the pair of its *α-Card* identifier (c_i) and its content in binary form (b_i).

$$P := \{\hat{p_1}, ..., \hat{p}_i, ..., \hat{p_k}\}$$
$$k := |P| \ : \ k \leq w$$
$$\hat{p}_i := (c_i,b_i) \ : \ c_i \in C, b_i \in binary$$

In order to prepare operational definitions, the superset of detached α-*Card* descriptors ($\mathbb{D}$), the total volume of adornments ($\mathbb{A}$), and the superset of payloads ($\mathbb{P}$) are defined.

$$\mathbb{D} := \bigcup_{i=1}^{w} \{A_i\} \ : \ |\mathbb{D}| = w$$
$$\mathbb{A} := \bigcup_{i=1}^{w} \bigcup_{j=1}^{|A_i|} a_j^i \ : \ |\mathbb{A}| = \sum_{i=1}^{w} z_i$$
$$\mathbb{P} := \bigcup_{i=1}^{k} b_i \ : \ |\mathbb{P}| = k$$

Now, accessors on descriptors can be defined as functions that provide individual adornments (*adorn*) or adornment values (*value*).

$$adorn : \ \mathbb{D} \times string \longmapsto \mathbb{A}$$
$$adorn(A_x, sel) = \ \{a_j^x \mid a_j^x \in A_x \ \wedge \ n \in a_j^x \ \wedge \ n = sel\}$$
$$value : \ \mathbb{A} \times string \longmapsto string$$
$$value(A_x, sel) = \ \{v \mid v \in a_j^x \ \wedge \ a_j^x \in adorn(A_x, sel)\}$$

From a top-level perspective of the α-*Doc* (*AD*) another set of accessors can be defined that provide detached descriptors (*ddesc*), adornment values (*adval*), or payload content (*paylbin*) on the basis of an α-*Card* identifier (c_x).

$$ddesc : \ C \longmapsto \mathbb{D}$$
$$ddesc(c_x) = \ \{A_x \mid c_x \in \hat{d}_x \ \wedge \ A_x \in \hat{d}_x\}$$
$$adval : \ C \times string \longmapsto string$$
$$adval(c_x, sel) = \ \{value(A_x, sel) \mid c_x \in \hat{d}_x \ \wedge \ A_x \in \hat{d}_x\}$$
$$paylbin : \ C \longmapsto \mathbb{P}$$
$$paylbin(c_x) = \ \{b_x \mid c_x \in \hat{p_x} \ \wedge \ b_x \in \hat{p_x}\}$$

The α-*Doc*-embedded software component (α) contains the set of functions. The complete operational semantics of the α-*Doc* is not defined formally but is specified by the entirety of its source code, which is indicated by the ellipses (...).

$$\alpha := \ \{ddesc, adval, paylbin, adorn, value, ...\}$$

The coordination cards are handled as any other α-*Card*s with a descriptor and a payload. The surrogate key that identifies each coordination card is a reserved keyword in form

of the strings "$PSA", "$CRA", and "$APA". The payload of the coordination cards will be of special interest for the system design and can be symbolically defined as *psapl*, *crapl*, and *apapl*.

$$psapl = \{\hat{p_i} \mid c_i \in \hat{p_i} \wedge s_i \in c_i \wedge s_i = \text{“\$PSA”}\}$$
$$crapl = \{\hat{p_i} \mid c_i \in \hat{p_i} \wedge s_i \in c_i \wedge s_i = \text{“\$CRA”}\}$$
$$apapl = \{\hat{p_i} \mid c_i \in \hat{p_i} \wedge s_i \in c_i \wedge s_i = \text{“\$APA”}\}$$

Finally, the descriptors ($\hat{d_i}$) and the payloads ($\hat{p_i}$) are versioned. In fact, the hat ($\hat{}$) indicates that the formula relates to the latest version of each artefact. The bar ($\bar{}$) can be used to indicate the complete volume of versioned artefacts. The mechanisms of versioning will be discussed later. For the sake of completeness, the actual structural definition of an α-*Doc* that internally versions the descriptors and payloads ($\bar{AD}$) can be defined as follows.

$$\bar{AD} := (e, C, \bar{D}, \bar{P}, \alpha, r)$$
$$\bar{D} := \{\bar{d_1}, ..., \bar{d_i}, ..., \bar{d_w}\}$$
$$\bar{P} := \{\bar{p_1}, ..., \bar{p_i}, ..., \bar{p_k}\}$$

The formalization of the α-*Doc* provides a model by which all embedded content units can be considered as mathematical sets of facts. Thus, the α-*Doc* representation is prepared for applying an inference engine (cf. 7.2) for querying such sets of facts.

6.1.5 Adornment Model

In α-*Flow*, adornments are process-relevant status attributes and represent certain aspects of an α-*Card*'s life-cycle and process-related state. Adornments classify α-*Card*s passively or an adornment status change can actively act as an event trigger that implies process progression.

Corpus Genericus

The basic α-*Adornment* model for α-*Card*s has been discussed in [55] and consists basically of adornments for: OC, role, contributor, and institution, validity and visibility, version and variant, fundamental semantic payload type, syntactic payload type, and domain-specific semantic payload type, due date and priority, deferred flag and deleted

flag. The collective of exactly these α-*Adornment*s is considered as the "corpus genericus" of the adornment prototype. The overall concept of the adaptive adornment model has been published in [309] and will later be discussed later. For now, it can be assumed that the APA allows to define a data type, value range, and default value for each α-*Adornment*. The table 6.1 accordingly provides an overview of the predefined α-*Adornment* configuration of the corpus genericus.

The identifiers for role, institution, contributor, and OC have already been mentioned. As an α-*Adornment*, these identifiers reference the corresponding information unit that is managed within the CRA payload. The α-*Flow* replicate that has been executed on a node knows its current user. This information is used as default value if a new α-*Card* is created by the user. Similarly, the information about the OC is initially provided by the first actor when the first α-*Doc* replicate is initially created for case initiation and the OC remains constant for the whole case episode. For conceptually preserving the self-contained existence of each α-*Card* descriptor as an electronic document, independent of its case file, the OC reference is additionally stored in each α-*Card* descriptor.

The visibility and validity of α-*Card*s must be considered separately. In traditional database-centric approaches, visibility is strictly coupled to validity (cf. sect. 5.6.1). For document-centric approaches, it is common to share preliminary documents, by making them visible, even though the content is not guaranteed to be valid. The validity model essentially consists of the classifiers invalid / valid, whereas the visibility model essentially consists of the classifiers private / public. The semantics of the four resulting combinations have been discussed in section 5.6.1.

A versioning model is supported both for content and coordination α-*Card*s. Versioning is mandatory for public and valid α-*Card*s because the individual systems require a global version for the tracking of changes. Any other α-*Card*s, in terms of visibility and validity, are equally allowed to use versioning, as it seems appropriate to the human owner of the α-*Card*.

A variant model was also prepared from the onset of the project. In contrast to versions, there may coexist several variants of an α-*Card*. The particular consideration had been to support variants of the PSA, thus, allowing for variants of the therapy plan in healthcare. Support for PSA variants could provide a platform to integrate formal methods of distributed multi-variant consensus finding. However, the variant α-*Adornment* is unused and multi-variant consensus methods are not supported and are currently out of scope.

The syntactic payload type describes the format of an α-*Card*. A common standard that provides a reference for the syntactic payload types is the MIME [310]. With two semantic payload types, an α-*Card* is classified semantically. The fundamental semantic

α-Adornment	Data Type	Value Range	Default Value
AlphaCard-Title	string	(arbitrary character string)	–
Role-ID	string	(arbitrary character string)	(node-specific)[1]
Institution-ID	string	(arbitrary character string)	(node-specific)
Contributor-ID	string	(arbitrary character string)	(node-specific)
OC-ID	string	(arbitrary character string)	(episode-specific)[2]
Visibility	enumeration	{Private, Public}	Private
Validity	enumeration	{Invalid, Valid}	Invalid
Version	string	(arbitrary character string)	"0"
Variant	string	(arbitrary character string)	"0"
Syntactic-Payload-Type	string	(arbitrary character string)	–
Fundamental-Semantic-Type	enumeration	{Coordination, Content}	Content
Semantic-Content-Type	enumeration	{Documentation, Referral Voucher, Result Report, ...}	–
Due-Date	timestamp	(arbitrary date)	–
Priority	enumeration	{Low, Normal, High}	Normal
Deferred	enumeration	{True, False}	False
Deleted	enumeration	{True, False}	False

[1] The contributor is derived from the current user of the α-*Doc* replicate
[2] The patient is adopted from the α-*Doc* for each α-*Card* descriptor

Table 6.1: The predefined α-*Adornment*s that constitute the corpus genericus of the adornment prototype

type is distinguished from the semantic content type. The fundamental semantic type classifies α-*Card*s into content vs. coordination. In particular, this adornment's purpose is to mark the coordination cards PSA, CRA, and APA. The semantic content type classifies content artefacts, for example, as "documentation", referral voucher", or "result report" and even more specifically as "anamnesis", "diagnosis request", "diagnostic finding", or "therapeutic measure".

The due date adornment allows to annotate the α-*Card* with a date. Currently, there are no automated actions defined on the due date adornment. Yet, appointment scheduling is a very important aspect of process support in healthcare (cf. [22]). Thus, the due date will be a good candidate for triggering user-defined rules and actions (cf. sect. 5.5.2, →XR18), which are currently not supported (cf. sect. 5.7.1).

The priority has been taken into account because e-mail supports this kind of marker. The possible values are normal, high, and low. Changing this adornment is not related with the prioritization of the α-*Card* in the work-list. It provides just a visual marker, like e-mail applications.

The deferred adornment is motivated by Scrum (cf. sect. 2.2.11). Scrum cards that have a so-called impediment get a deferral marker. An impediment is any kind of problem that prevents a task from being started, continued, or finished. Thus, the adornment can be used to publicly indicate to cooperation partners that the fulfilment of a work-item cannot be assumed in due time (by customary standards) but is deferred indefinitely.

Finally, the deleted adornment is important because nothing gets ever physically deleted in the versioned distributed case file of *α-Flow*. The user is allowed to "delete" *α-Card*s but this just marks the cards accordingly and "deleted" cards are merely filtered from the standard graphical work-list display. It is possible to get a display of the "deleted" cards and to undelete them by removing the marker any time.

The overall concept of the *α-Adornment*s as a key-value list ultimately allows users to configure arbitrary adornment templates in the APA as user-defined indicators and annotations, like the *condition indicator* or *diagnosis certainty* (cf. sect. 5.5.1). The adaptive characteristics of the *α-Adornment*s implementation will be discussed in chapter 7. For visualization purposes, a general graphical representation of an *α-Card* descriptor will be used that is outlined in figure 6.4.

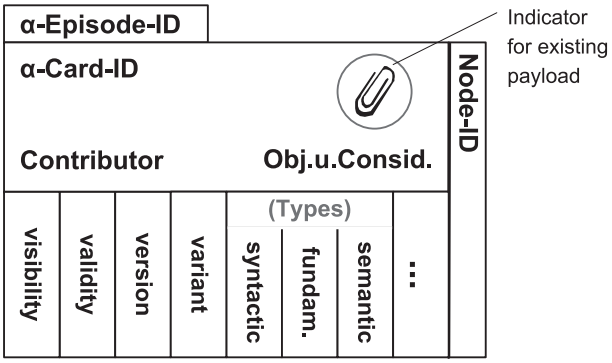

Figure 6.4: General visualization of an arbitrary *α-Card* descriptor with some *α-Adornment*s for illustrative purposes

Adornments as Trigger for Rule-Based Actions

The so-called *α-Kernel*[2] is a subsystem of *α-Flow* that contains a rule engine. It monitors adornment changes and initiates various actions, for example, document versioning and

2 The *α-Kernel* subsystem was originally named *α-Props*, e.g., published in [311]. By some remodularization it became just one subsystem amongst others like the embedded editor, versioning, or synchronization, which are also active properties of the *α-Doc*. Still, the original *α-Props* remains a centrepiece of the *α-Flow* architecture (cf. sect. 6.2). Thus, it was renamed as *α-Kernel*.

message-based synchronization. A basic scenario in which an *α-Adornment* triggers an active property can be illustrated by the visibility and validity adornments.

An *α-Card* represents an open task if there is only the descriptor but no payload. It represents a fulfilled task if there is a payload with visibility set to public and validity set to valid. Additionally, there is a rule: "A public-valid *α-Card* requires versioning". Amongst others, the active properties implement rules like this one. Thus, they may automatically change adornments like the version adornment as a side-effect, which is illustrated in figure 6.5. In fact, changing the versioning adornment will trigger another active property that performs the actual 'versioning action' with its technical implications.

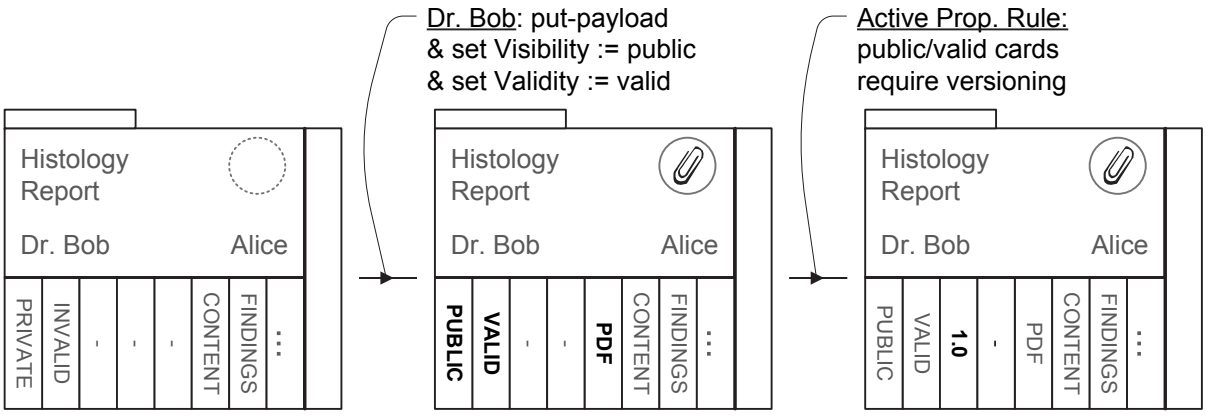

Figure 6.5: Actors or active property rules are exemplarily changing adornment states.

6.2 Architectural Overview of α-Flow

α-Flow is not only a modelling approach but there are connections to system construction. The *α-Flow* engine is composed of several subsystems. The ones that implement direct user interactions are *α-Startup*, *α-Injector*, *α-Editor*, and *α-Forms*. The subsystems that constitute the logic layer are *α-Kernel* and *α-Adaptive*. The subsystems that handle local storage and network messaging are *α-VVS* and *α-OverNet*. An overview on the *α-Flow* architecture is provided by figure 6.6. As a unit of deployment, the *α-Flow* engine is bundled in form of a single Java Archive (JAR) file.

The *α-Startup* subsystem provides the command-line interface and initializes the other subsystems. The *α-Injector* provides drag-and-drop functionality in order to contribute electronic documents into an *α-Doc* case file. The *α-Injector* also handles the initial creation of an *α-Doc*, the so called "alph-o-matic injection" that initially transforms a

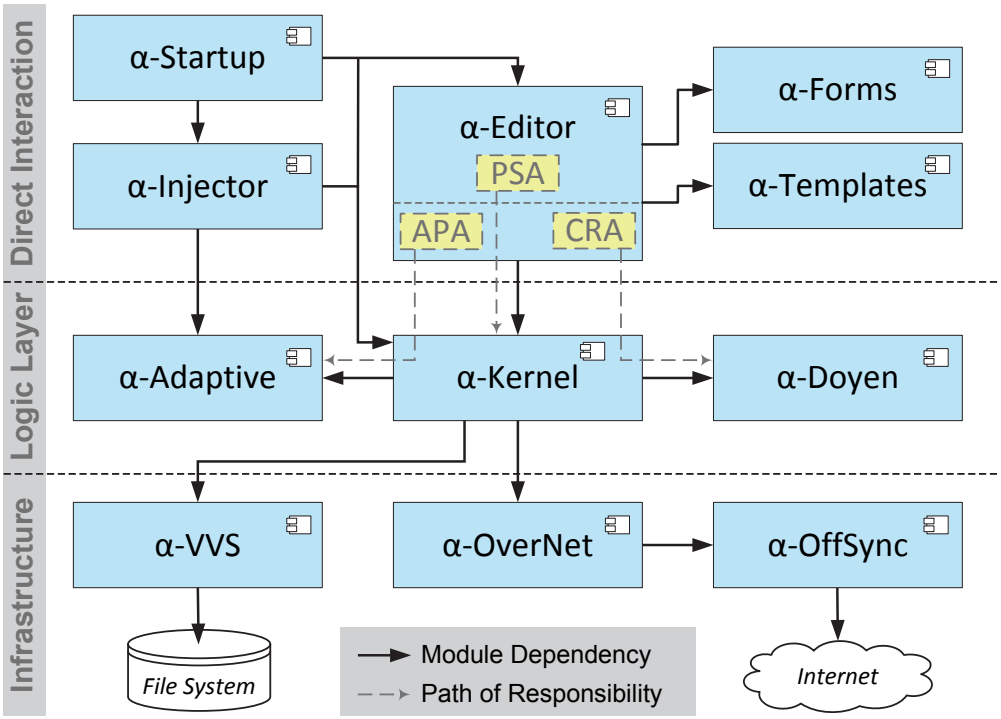

Figure 6.6: Architectural overview of the *α-Flow* engine

passive document file into an active document. The *α-Editor* is an embedded viewer and editor. For example, a central dashboard provides an overview on the card-based work-list. The *α-Editor* also allows for accessing, viewing, and editing of the original content units, the electronic documents stored as *α-Card* payloads, through common editors in the local information system. The *α-Forms* subsystem provides a combined form designer and form editor. For example, it is possible to create a checklist with *α-Forms*. The *α-Templates* subsystem provides import and export of process templates. The export allows for filtering case-instance-specific information from the PSA, CRA, and APA artefacts and combines the resulting elements into a single process template file.

The *α-Kernel* contains a rule engine as mentioned above. The *α-Kernel* monitors *α-Adornment* state changes and payload contributions. It automatizes any reactions and controls local storage and remote messaging. Any changes (either from the local user by the *α-Editor* or from remote peers by the *α-OverNet*) have to be made effective via the *α-Kernel*. The *α-Adaptive* subsystem provides run-time adaptiveness for the *α-Adornments* of each descriptor and for run-time changes of the adornment prototype that is stored within the APA payload. An *α-Adaptive*-related editing component allows human actors to configure user-defined adornments, like *condition indicator* or *diagnosis certainty* (cf. sect. 5.5.1). The *α-Doyen* subsystem handles participant information and

process role labels. The term “doyen” is an English synonym to “spokesperson” or “premier” and is a loose reference to process roles like the “process coordinator”. Amongst others, *α-Doyen* allows for handing over process roles like “process coordinator” from one actor to another like a token.

Concerning the three coordination cards PSA, CRA, and APA there is a distinct subsystem that is the technological sovereign of each. The subsystem that is responsible of the PSA is the *α-Editor* with its prime panel that provides a work-list editor, which is kind of a dashboard of *α-Card*s similar to a Scrum task board of cards. The subsystem that is responsible of the CRA is *α-Doyen*, which extends the *α-Editor* with according panels to view and edit participant information as well as dialogues to hand over process roles. The subsystem that is responsible of the APA is *α-Adaptive*. It provides an editor for the APA and its adornment specifications. Conceptually, it also provides the *α-Adornment*s instance viewer and editor that plugs into the *α-Card* dashboard of the *α-Editor*. These three architectural relationships are indicated in figure 6.6 with the yellow rectangles in the *α-Editor* and the gray arrows.

The subsystems of the infrastructure layer handle changes of *α-Card*s in terms of local storage or remote messaging. From the infrastructure perspective, all *α-Card*s are equal. The single subsystem that is allowed to invoke the infrastructure modules is the *α-Kernel*, which decouples the use-case specific logic from the generic facilities. The *α-VVS* subsystem contains an embedded Version Control System (VCS). It is the authority on the latest state of any content unit. It provides an independent version history for each *α-Card*, whereas each *α-Card* is a composite file unit of descriptor and payload. The *α-OverNet* subsystem implements an overlay network for messaging between the nodes of the distributed *α-Doc* replicates. Currently, the *α-OverNet* utilizes the *α-OffSync* module for data transfer. The *α-OffSync* implements a custom-made synchronization protocol, which applies a so-called Adaptive Vector Clock (AVC) to each *α-Card* as logical time. For data transfer, it uses Simple Mail Transfer Protocol (SMTP) & Internet Message Access Protocol (IMAP) as well as GnuPG for message encryption. The separation between *α-OverNet* and *α-OffSync* architecturally prepares a future replacement of the data transfer module with an implementation that uses store-and-forward messaging facilities of the eGK infrastructure.

6.3 Summary

This chapter has described the workflow language and artefact model of *α-Flow*. The formalization of the *α-Doc* provides a model by which the embedded content units can be formally described, for example, as sets of facts. The platform adornments have

been described, being collectively described as the corpus genericus. An example has illustrated how an adornment change may become the trigger for a rule-based action. Finally, the modular design of the *α-Flow* system architecture has been outlined. The purpose of the architectural overview is to have a reference for a more detailed discussion of the subsystems in the next chapter.

7 The α-Flow Implementation: Challenges and Design Choices

> “Every accomplishment starts with the decision to try.”
>
> (Anonymous)

The handling of distributed case files in form of active documents implies many implementation challenges. Together with several students, I explored the content-oriented workflow paradigm, case-driven cooperation, distributed synchronization, and the bundling of a thing that shall be document and application at once. The following sections provide a brief outline of the challenges and design choices for each α-*Flow* subsystem, describing these aspects that are of general interest. Most subsystems are accompanied by a student thesis, for further reading, each providing in-depth explanations about the according design and implementation.

7.1 Facilities for Direct Interaction

The α-*Startup* and α-*Injector* subsystems concern the handling of an α-*Doc* as a file at the user desktop. Both represent a considerable factor for the kind of user experience that can be expected from Java-based active documents. The α-*Editor* allows the users to access the case file contents. The initial implementation of α-*Startup* and α-*Injector* as well as an early form of the α-*Editor* was constructed by Stefan Hanisch for his master thesis [312].

7.1.1 α-Startup: File Bundling as an Executable JAR

The basic idea to represent a case file as molecular active document is to bundle a case handling engine and the case file contents in a single Java Archive (JAR)[1] file. A JAR file is a ZIP[2] archive. The idea is to implement a Java application bundle that is self-modifying some content part of its JAR.

The bundling of a Java application that contains all its dependencies like external libraries into a single executable artefact is not provided by a standard Java Virtual Machine (JVM) installation. The standard system class loader[3] is implemented by the JVM class sun.misc.Launcher$AppClassLoader, which is not able to load classes from a JAR inside a JAR. This limitation may be circumvented by unpacking all dependent JAR files and "repacking" their contents into a united JAR file. The alternative approach is to use a user-defined class loader (e.g., [313]).

α-Flow uses the One-JAR™ tool[4] for packaging all dependent libraries into a single deployable and executable unit. One-JAR™ provides a custom Java class loader that is included within an accordingly packaged JAR file. The *α-Startup* subsystem initializes all other *α-Flow* subsystems. It embeds the One-JAR™ facilities and ensures the deployment of the overall *α-Flow* engine as a single JAR file unit.

The second challenge in this context is to store the case file content within the JAR file itself. Unfortunately, this has been proven to be impossible at the moment. The symptom is that any time a class loader loads a JAR file, this JAR file's file descriptor remains open for the lifetime of the JVM. This is caused by a design flaw of the JVM and a Windows bug. The JVM design flaw is that up to Java 7 it has not been possible to tell the class loader that the application is finished with loading classes such that the operating system lock on the file could be released. Not until July 2011, Oracle has fixed this, basically by adding the Closeable.close() method to the interface of the class loader. In addition, Windows operating system has known issues with the involved kind of file locking. The problem concerns any systems with the objective to be "hot-deployable" (e.g., [314]). Systems like the Open Services Gateway Initiative (OSGi) container, that provide hot-deployment functionalities, can work around the problem because an OSGi

1 cf. http://docs.oracle.com/javase/6/docs/technotes/guides/jar/jar.html
2 cf. http://www.pkware.com/documents/casestudies/APPNOTE.TXT
3 In fact, the JVM uses a chain of class loaders (cf. http://docs.oracle.com/javase/tutorial/ext/basics/load.html). The first one is the *bootstrap class loader* that loads core Java libraries like the well-known rt.jar that is a pre-installed system library. The second one is the *extension class loader* that is implemented by sun.misc.Launcher$ExtClassLoader and that loads Java libraries or classes from the so-called extensions directories like $JAVA_HOME/lib/ext. The third one is the system class loader, which is colloquially referred as *the* class loader.
4 http://one-jar.sourceforge.net/

container loads other JAR files, the OSGi container does not try to replace or change itself, like it would be necessary for an *α-Doc*. For the purpose of *α-Flow*, the problem currently remains. John Mazz describes the situation in [315]. In conclusion, it is not possible to implement the original ideal form of an active document in Java.

As a result, the molecular *α-Doc* currently consists of a single JAR file and a directory. Both share the same name. The concept has been chosen in analogy to the Microsoft Internet Explorer that saves a complete web page, which is also a composite of files, as a HTML file and a folder that are associated by their shared file name. Finally, an "invite" button is provided, by *α-Flow* within its Graphical User Interface (GUI), that bundles a snapshot of the running *α-Doc* (its content directory and its application JAR) into a ZIP file. The ZIP file can be handed over to a new participant as a single file.

7.1.2 *α*-Injector: Self-Replication and Content Contributions

The user story in chapter 4 has described how a simple drag-and-drop of a passive digital file onto "a special icon on his desktop" creates the case file artefact. The design of *α-Flow* is simple: The "empty *α-Doc*" is used as drag-and-drop target. The empty *α-Doc* is synonymous to the *α-Flow* engine without case contents. Technically, it is simply the JAR file without any content directory. For the purpose of providing a simple and unambiguous term instead of "empty *α-Doc*" or "raw *α-Flow* engine", the JAR or "icon" is also referred to as "alph-o-matic".

The drag-and-drop of a passive file on the alph-o-matic transforms the passive file into an active document, with the original passive file as the first contributed content unit within the embedded case file. This transformation is referred to as *alph-o-matic injection*. The *α-Injector* subsystem handles the alph-o-matic injection, which is self-replicating the *α-Flow* engine. The *α-Injector* logic finds out from which JAR file it is currently running. Then the provided file is examined and the *α-Flow* engine JAR is copied in the same directory of the given file with the same file name. Then some graphical dialogues ask the user to provide his or her information as a contributor as well as essential case information like the case episode name and the Object under Consideration (OC) information. Finally, the alph-o-matic creates the initial Process Structure Artifact (PSA), Collaboration Resource Artifact (CRA), and Adornment Prototype Artifact (APA) file instances for the new *α-Doc*.

The advantage of providing the alph-o-matic application as an identical executable to the *α-Doc* application is that the *α-Flow* engine becomes "virulent", in the best possible notion of the word: If a user has ever participated in any *α-Flow* case, he or she already

has everything that is needed to easily create new *α-Doc* case files. The *α-Flow* engine can be reused by simply copying its JAR file.

Finally, what should happen if a user drag-and-drops a digital file onto a non-empty *α-Doc*? This can mean that the user wants to create a new case file for a new patient. However, it more often simply means that the user wants to contribute the file into the case file. The user could provide it as the result report of an already planned *α-Card* or the user could want to create a new *α-Card* for an emergent content unit. All three possible purposes are supported. A dialogue asks the user of his intention; accordingly, more dialogues handle each use case. In conclusion, the purpose of the *α-Injector* subsystem is to handle[5] all operating system drag-and-drop events.

7.1.3 α-Editor: Dashboard and Content Access Delegation

The *α-Editor* is used to edit the coordination information like the work-list or to gain access on the case file contents. Access to the contributed electronic documents is provided through common editors in the local information system. Thus, an application delegation mechanism has been implemented that provides an abstraction of the operating system-specific facilities for executing locally installed applications. The operating systems Windows Vista and Mac OS X are supported. Linux is also supported but the according delegation facilities depend primarily on the applied window or desktop manager, thus, only KDE and GNOME are supported.

The *α-Editor* is implemented in Swing[6] in order to avoid additional external graphics libraries that would significantly increase the binary footprint of the *α-Doc*. The details on the *α-Editor* implementation are documented in [312].

The graphical layout resembles the style of web pages. It uses a header and footer on top and bottom as well as a navigation menu at the left side and a main information area at the centre. The original conceptual sketch of the user interface prototype, from 2009, is documented in the appendix sect. B by figure B.1. A screenshot of the current *α-Editor* is given in figure 7.1. The screenshot illustrates the card-based work-list.

The work-list shows a list of cards in different content progression states. Referrals and result reports are in a cohesive-content relationship and are displayed as connected pairs of cards. The display focus is on the sonography report and the focus is indicated by

5 Only the initial transformation of a passive document into an active document is considered as an *alph-o-matic injection*, the drag-and-drop contribution of content units into an existing case file is referred to as *content injection*.

6 e.g., `http://docs.oracle.com/javase/tutorial/uiswing/`

Figure 7.1: A screenshot of the *α-Editor* implementation

the small magnifier icon. The right part of screen displays the *α-Adornment*s of the currently focused *α-Card*. At the top right corner of the screenshot, there are buttons for opening and contributing payloads. However, it is also possible to open the latest version of the *α-Card* payload by double-clicking on the cards in the central list. In addition, Java file-drop handlers are implemented that allow the user to drag-and-drop a data file onto the card widgets for contribution purposes.

7.1.4 α-Forms: Checkbox-Based Checklist Forms

The basic *α-Card* payload model is content-agnostic. Yet, one sort of documents is of special significance for human-oriented workflows: the "checkbox-based checklist form". From a certain perspective, the shared card-based work-list is another type of a checklist. If the distributed case file and shared therapy plan are the macro-management of distributed case handling, individual checkbox-based checklist forms can support single process steps as a micro-management. Lenz provides an example of such a checklist form in [31]. The *α-Forms* subsystem has been constructed by Florian Wagner for his master thesis [316] and it provides an editor to design and to fill-out a checklist form.

The challenge of the *α-Forms* project is to provide both the editor for the form schema and the form instance values in a single easy-to-use application. The general idea is influenced by Hypertext Markup Language (HTML) forms [317] and the World Wide Web Consortium (W3C) standard XForms [318] which is based on Extended Markup Language (XML). Nevertheless, neither HTML forms nor XForms allow the storing of instance specific values. For web forms, the instance data is sent as a key-value list to the server but it is not possible to store filled-out web forms as a HTML desktop file because the key-value list cannot be captured locally.

Florian Wagner's thesis [316] provides an evaluation of form-related industry standards and scientific approaches. In conclusion, *α-Forms* has been implemented as a Java-based editor for easy integration into the *α-Flow* architecture. It provides a graphical form composer and a fill-in editor. The composer provides a palette with graphical standard components like checkboxes and text fields. The user can drag-and-drop the form elements from the palette into the form layout. An *α-Forms* example in composer mode is illustrated in figure 7.2. The form loosely resembles the example in [31]

A composed form can be archived as a template and it can be replicated like paper-based forms in order to let them be filled out by other users. Thus, the *α-Forms* editor provides also a fill-in mode. The same example as above is illustrated in fill-in mode by figure 7.3.

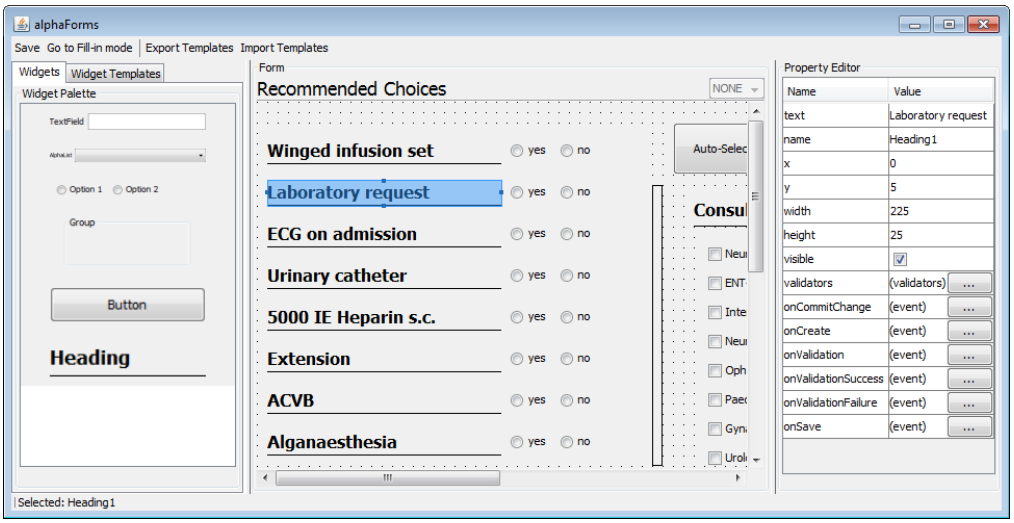

Figure 7.2: A screenshot of the *form composer mode* of the *α-Forms* editor

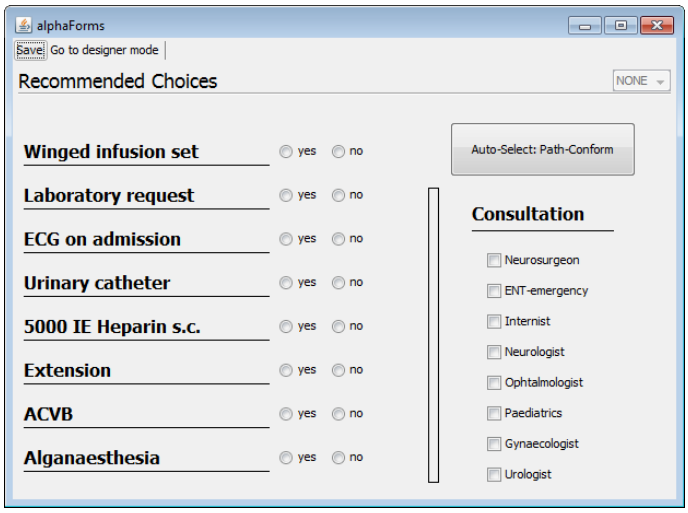

Figure 7.3: A screenshot of the *form fill-in mode* of the *α-Forms* editor

The *α-Forms* system serializes the form schema and layout as well as the filled-out values in a single XML file. In fact, *α-Forms* is designed as an autonomous application. It can be used independent of the *α-Flow* engine; thus, *α-Forms* can be created as autonomous active documents that only carry the *α-Forms* editor as active property.

From the data model perspective, the schema part has been designed in loose analogy to XForms (more details are provided in [316]). The instance data is stored using the

memento[7] programming pattern [319]. A memento object is mainly the serialization of the state of a programming language object. To a certain extend, the set of memento objects is the equivalent to the key-value list that represents HTML form values.

There are some advanced features provided by the *α-Forms* editor. For example, groups of graphical components can be stored as so-called widget templates. Widget templates can be imported and exported as separate files, which is technologically easy because the widget template serialization is the same as the one for the overall form.

Another feature concerns the "Auto-Select"-button in the example illustration. In order to have interactive forms that allow, for example, some form of pre-selection of values, the *α-Forms* editor integrates a JavaScript interpreter[8]. Thus, it is possible to embed automated actions that change some form values, for example, on the event of a click at some button. The scripted actions can be changed at run-time by the user. However, specifying automated actions requires an experienced programmer who understands the internal *α-Forms* data model and is not suited for end-users.

Finally, *α-Forms* is integrated into the *α-Flow* engine. Thus, it is possible to have checklists and forms as *α-Card*s of an *α-Doc*. It is conceivable that decisions in the local context of a form might have an implication to the shared work-list of the cooperation. Thus, each *α-Forms* instance carries a reserved content state attribute whose value can be controlled via JavaScript. The *α-Forms* state attribute could be used like an *α-Adornment* for state-based interaction with an encompassing *α-Doc*. However, an interaction based on content states between *α-Forms* and its encompassing *α-Doc* has not been implemented and is currently out of scope.

7.1.5 α-Templates: Import and Export of Process Templates

The card-based work-list is a basic articulation of the inter-institutional workflow in terms of "What?", "When?", "Where?", and "Who?". Processes that occur in inter-institutional cooperation are generally repeated in similar form. Thus, it should be possible to export and import the workflow-related information of a case in form of a process template. The *α-Templates* subsystem accordingly provides such facilities. It has been constructed by Patrick Reischl for his bachelor thesis [320].

7 The original purpose of the memento pattern was to provide the ability to restore an object to its previous state, for example, for undo operations.

8 The Rhino engine, which is an open-source implementation of JavaScript written entirely in Java (cf. `http://www.mozilla.org/rhino/`)

The workflow-related information of an *α-Flow* case is the information managed in the PSA, CRA, and APA artefacts as well as the *α-Card* descriptors. The *α-Templates* export allows for filtering case-instance-specific information from these artefacts. *α-Templates* combines the resulting elements into a single process template file, in XML format, which can be imported into another *α-Doc*. The challenge for *α-Templates* is that there are several degrees of case-specific information. For example, one doctor often works together with exactly the same partners. The doctor would not like to remove the institution and actor information from the case because they should be available for the next case. The patient information would, of course, always be filtered at process template creation. In another setting, it might be desired to remove the actor and institution information but to retain the role information. In addition, some task cards that, for instance, concern secondary care institutions should retain the institution information (i.e. some regional hospital) but no specific actor references. Moreover, the consulted pathologist remains always the same person. Thus, some task card for a histology report should yet retain the actor-specific information. Arbitrary combinations could be required as *α-Card*-individual degrees of filtering.

In addition, the user-defined adornments have influence on the creation of process templates. If some were created in a given case, the easiest scenario would carry all of them over into the template. Still, it could be required to adopt only some used-defined adornments. A third variant could be, that all user-defined adornments should in general be retained as part of the adornment prototype of the template, but in regard to the work-list, all given *α-Card* descriptors should be cleansed from adornments, before creating the template. In conclusion, the required information of a process template depends on the user who creates it. In general, *α-Templates* provides the user with a set of dialogues that allow him or her to configure filtering of all four parts (the PSA, CRA, APA, and *α-Card* descriptors) on different degrees. Thus, for template creation purposes the user is lead through a sequence of dialogue steps (a “software wizard”), which is illustrated in figure 7.4.

The information that is required from a process template also depends on the user who reuses the template. The template producer and its consumer must not necessarily be the same person. Thus, the same considerations about degrees of case-specific information affect the import as well. Even if there is a process template with many specific details, for example, the institution information is completely retained, the user who adopts the process template for a new case might want to remove all or parts of the template information before case instantiation. Thus, the challenge for the *α-Templates* design has been to use the same sequence of filter steps and dialogues for the importing and for the exporting of process templates. Figure 7.5 illustrates the resulting equivalence of both use cases.

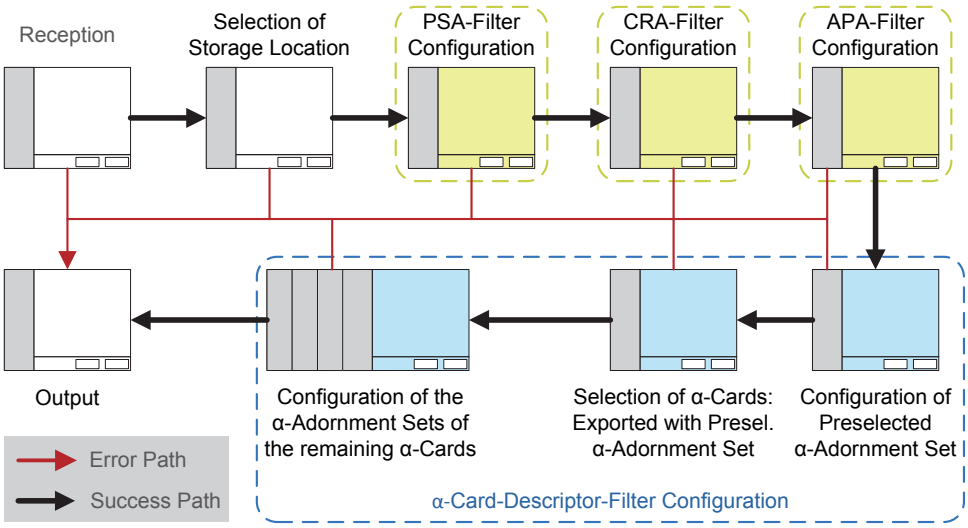

Figure 7.4: The dialogue sequence of the α-*Templates* subsystem (adapted from [320])

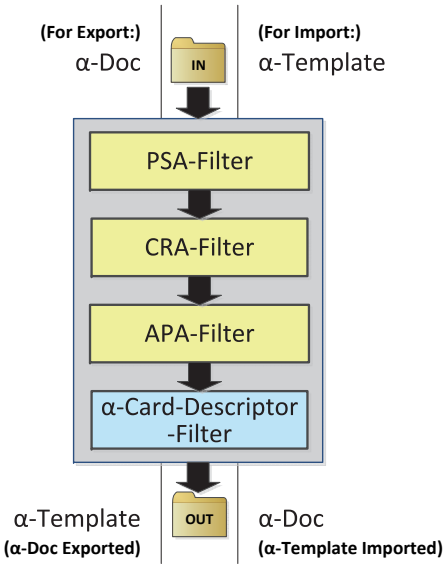

Figure 7.5: The α-*Templates* filter-chain: equivalence of process template importing and exporting (adapted from [320])

Finally, the α-*Templates* subsystem also extends the drag-and-drop facilities of the α-*Injector*. It enables users to drag-and-drop a process template file on the alph-o-matic, which initializes the import wizard. A special filter configuration can be made: the full-export and the full-import. The full-export still removes basic information like the patient information. The full-export also resets the α-*Card* descriptors, for example, it removes any version information and resets visibility and validity to private and invalid (the complete set of pre-definded α-*Adornment* resets for the corpus genericus adornments

is documented in [320]). The full-export and full-import are especially convenient if the process template needs to be edited. A special editor for process templates is not required. The process template can be edited by the standard *α-Flow* engine. It can be dragged on the alph-o-matic, using a full-import, then it can be edited by standard *α-Doc* means, and then it can be re-exported with a full-export. The temporarily created *α-Doc* can just be deleted. Thus, the process template is an empty case file without content and without case handling engine. From a certain perspective, an *α-Templates* XML file is the "pure-workflow" counterpart to the "pure-logic" alph-o-matic, both being one kind of abstraction of an instantiated *α-Doc*.

7.2 Subsystems of the Logic Layer

The logic layer subsumes the subsystems *α-Kernel*, *α-Adaptive*, and *α-Doyen*. The *α-Kernel* subsystem monitors any changes and provides reactivity. The *α-Adaptive* subsystem manages the APA and provides a data model and editing facilities for run-time adaptive content status attributes. Finally, the *α-Doyen* manages the CRA and supports transfer of process roles.

7.2.1 α-Kernel: Rule Engine and Change Control Centre

The example for monitoring *α-Adornment*s states and reacting on state changes has been illustrated in section 6.1.5. There are policy rules like "a public-valid *α-Card* requires versioning", and there are infrastructure rules that control the necessary remote synchronization operations in case of payload update events or descriptor change events. The *α-Kernel* subsystem has been initially constructed by Aneliya Todorova for her master thesis [321] and has been published in [311].

From the onset of the *α-Flow* project, the core logic should be implemented with a rule engine because of system evolution considerations. Rule engines commonly allow changes of rules and actions at run-time. Thus, end-user can be offered to reconfigure policy rules on demand in declarative form. Rule engines combine the reactivity of an Event-Condition-Action (ECA) mechanism with the reasoning and querying capabilities of inference engines.

The ECA principle is known from active Database Management Systems (DBMSs) [322, 323]. The ECA concept provides a general formalism for an event-monitoring scheme that detects manipulation-activities of data, and automatically executes actions in response, when certain events occur and particular conditions are met. ECA rules generalize mechanisms such as assertions, triggers, alerts, database procedures, and production

rules [324]. An ECA event may indicate the beginning or ending of database operations (delete, update, insert), a signal from user processes, or a temporal event from a system clock. An ECA condition is checked, for example, via auxiliary queries on the persisted data. An ECA action is commonly the invoking of a program or a procedure inside a database.

Inference engines implement a predicate logic and derive transitive conclusions from an initial set of given facts, stored in a knowledge base. Inference engines can use either a *forward-chaining* or a *backward-chaining* algorithm for deriving implications [325]. Inference engines have traditionally been used to implement expert systems (e.g., [326]), software agents (e.g., [327]), or rule-based systems for problem solving (e.g., [328]). However, the objective of reasoning over facts by an inference engine is primarily an analytical one and inference engines do not necessarily allow the execution of actions.

Rule engines allow ECA-based action execution based on inference-ascertained conditions. Anelyia Todorova evaluated several Java-based rule engines and finally used the JBoss Drools™ library to implement the *α-Kernel* subsystem. The inner architecture of the *α-Kernel* is illustrated in figure 7.6.

The Drools™ engine provides a Domain Specific Language (DSL) in which rules and actions can be programmed. The primary components of the rule engine architecture is the knowledge base, the session with its working memory, and the agenda. The knowledge base contains the rules and action declarations that are read from the DSL files. The session is used at run-time to insert fact objects into the working memory. A pattern matcher decides per inference what rules qualify for execution but they are not immediately executed but put on the agenda. In case of several matching rules, the agenda applies a so-called conflict resolution to order the rules. The conflict resolution can be influenced, for example, by the rule programmer with the assignment of prioritizations. The rules are executed, one by one, according to the order by the agenda. If the actions change the fact set, the pattern matcher can remove or add rules from the agenda according to the new state of the facts. Further documentation of the Drools™ engine is provided in [329].

A basic set of rules have been implemented, initially. Further projects like *α-Adaptive* and *α-OffSync* have successively extended the rule base with pre-defined rules and actions for the corpus genericus adornments. A run-time editor for possible user-defined rules and actions has not been implemented and is currently out of scope.

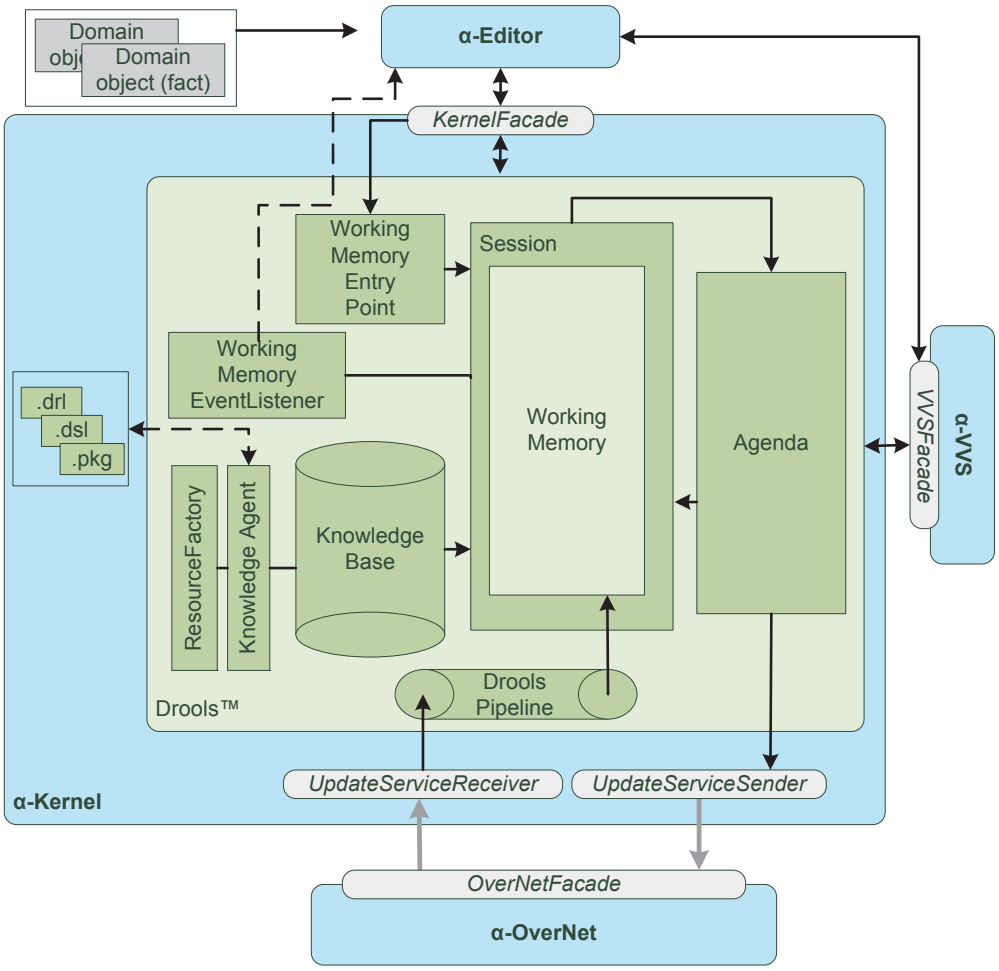

Figure 7.6: The inner architecture of the *α-Kernel* subsystem, embedding a JBoss Drools™ rule engine (adapted from [321])

7.2.2 α-Adaptive: Run-time Adaptive Adornments and the Adornment Prototype

The motivation for user-defined adornments has been discussed in section 5.5, with *condition indicator* and *diagnosis certainty* as two adornment examples. The *α-Adaptive* subsystem provides facilities for end-users to configure custom *α-Adornments* at run-time. The *α-Adaptive* subsystem has been constructed by Peter Schwab for his master thesis [330] and is published in [309].

The general objective of *α-Adaptive* is to allow persisting of data that was not known at design-time or deploy-time. Traditional database schema design freezes semantic decisions at design-time just like classes in programming do. Entity-Attribute-Value (EAV) schema design [331] is a generalization of row modelling. EAV is based on association

lists that originated in artificial intelligence (e.g., [332]). In contrast to the traditional schema design, the EAV design proposes a generic table with three columns: 1) the ID of an entity, 2) the name or identifier of an associated attribute, and 3) the corresponding attribute value for the entity. Thus, semantic decisions for an object are decoupled from altering the database schema because an arbitrary number of attribute-value pairs can be added at run-time.

The *α-Flow* implementation extends the basic EAV schema design by three elements. The first extension is the *consensus scope*, which has been discussed in section 5.5. It is used to capture at which level the consensus about the adornment value range has been established. Currently, four scopes are implemented: users can choose between values *episode-specific*, *institution-specific* and *domain-specific* – the value *generic* is reserved and indicates *α-Adornment*s that belong to the corpus genericus being used to grant the *α-Flow* platform functionality.

The next EAV extension concerns *user-centric data types*. In the original EAV, the physical data type of the attributes is a generic data type like String. There is no data type information included and data type transformations are commissioned to the application. Yet, adornments are user-centric and we require a slender type set from which a user might select a type for his or her adornment. Most data type sets in computer science are system-centric, e.g. primitive types in programming languages[9] or the ones in XML schema as a platform neutral superset. These data types are only comprehensible for programmers and are not adequate to fulfil an end user's plain idea of data types. As a standard for user-centric types, we use the Requirements Interchange Format[10] (ReqIF) as a reference because requirements management is highly user-centric and ReqIF provides a slender type set. Thus, the data types implemented for *α-Adaptive* are *String*, *Integer* (e.g., indicators like BI-RADS), *Timestamp* (e.g., due dates), *Enumeration* (e.g., indicators like *condition indicator* and *diagnosis certainty*) and *TextBlock* (e.g., Post-it notes). *α-Adaptive* extends the EAV schema by adding an additional attribute to store the user-centric data type restriction.

The third extension concerns the prototype-based relationship (cf. sect. 2.2.7) between the APA and all *α-Card* descriptors that are cloned from the APA. The APA contains the superset of all adornments that are used in any *α-Card* of one *α-Doc*. Yet, not necessarily all APA-configured adornments should be used on each *α-Card*. For example, the BI-RADS indicator is only suitable for mammography reports. Thus, a marker is necessary, that indicates the minimum set of adornments that are cloned per default for

9 For example, in C++ a programmer must choose between types {short int, int, long int} crossed with {signed, unsigned} semantics in order to create an arbitrary integer variable.

10 http://www.omg.org/spec/ReqIF/1.0.1/11-04-02.pdf

every *α-Card*. This marker is the *instance flag*. *α-Adornment*s that are not marked in the APA with the instance flag can still be added by the user to each *α-Card* individually, on demand.

In the end, the *α-Adaptive* provides an adornment prototype editor to the user, which allows for changes like renaming adornments, switching consensus scope, or changing data types. A screenshot of the APA editor is outlined in figure 7.7.

The APA is the clonebase for all *α-Card* descriptors. As the considerations about the instance flag have indicated, the user can individually select the *α-Adornment*s that should be used for an *α-Card*. Thus, the APA has a clone-and-select relationship with each *α-Card* descriptor. The basic principle is illustrated in figure 7.8.

The implemented *α-Adornment*s instance editor has already been displayed as part of the screenshot of the work-list dashboard in figure 7.1 on p. 217 as the right part of the screenshot. The button being labeled with "Goto: Schema View" will switch into a mode that compares the focused *α-Card* descriptor with the APA and allows the user to select individual *α-Adornment*s to be added or removed from this descriptor. For the sake of completeness, the same scenario that has been captured by figure 7.1 has been again captured displaying the adornment schema editor; the screenshot is provided in the appendix B by figure B.2.

Java is a class-based programming language and not a prototype-based language. The APA is a composite object structure and cloning of the APA should also clone all subsidiary objects, which is called a *deep clone*. Java provides a simple concept of object cloning because every Java object inherits the method clone() from the Java root class Object[11]. However, the Java cloning facilities provide only a so-called *shallow copy* because any subsidiary objects of an original object are not also cloned but the references of the clone point at the original object's sub-structure. The implication of the Java cloning and the resulting shallow copy is illustrated as the top half of figure 7.9. If a deep copy cloning needs to be implemented, the usual way would be to override the clone() method and to manually apply the cloning to all sub-structures, recursively. *α-Adaptive* uses another approach. The object structure for the *α-Card* descriptors is already prepared for serialization. Thus, the APA is merely serialised and deserialised in an in-memory buffer. This provides a deep copy clone of an arbitrarily complex object structure. The principle is illustrated as the bottom half of figure 7.9.

11 If the programmer wants to use the Object-inherited clone() method, he or she must add the empty marker interface Cloneable to the class, otherwise the JVM complains with a CloneNotSupportedException at run-time.

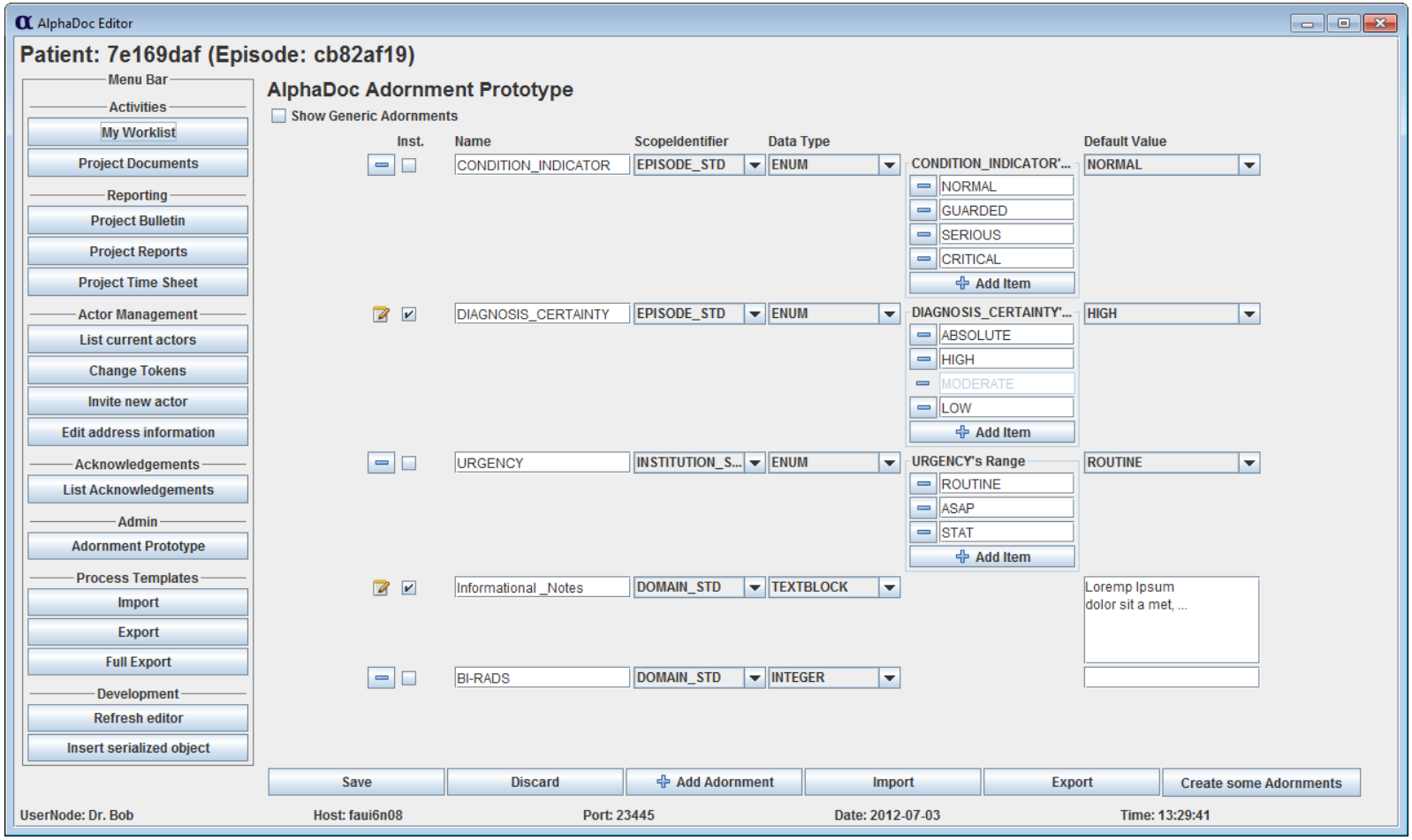

Figure 7.7: The adornment prototype editing panel

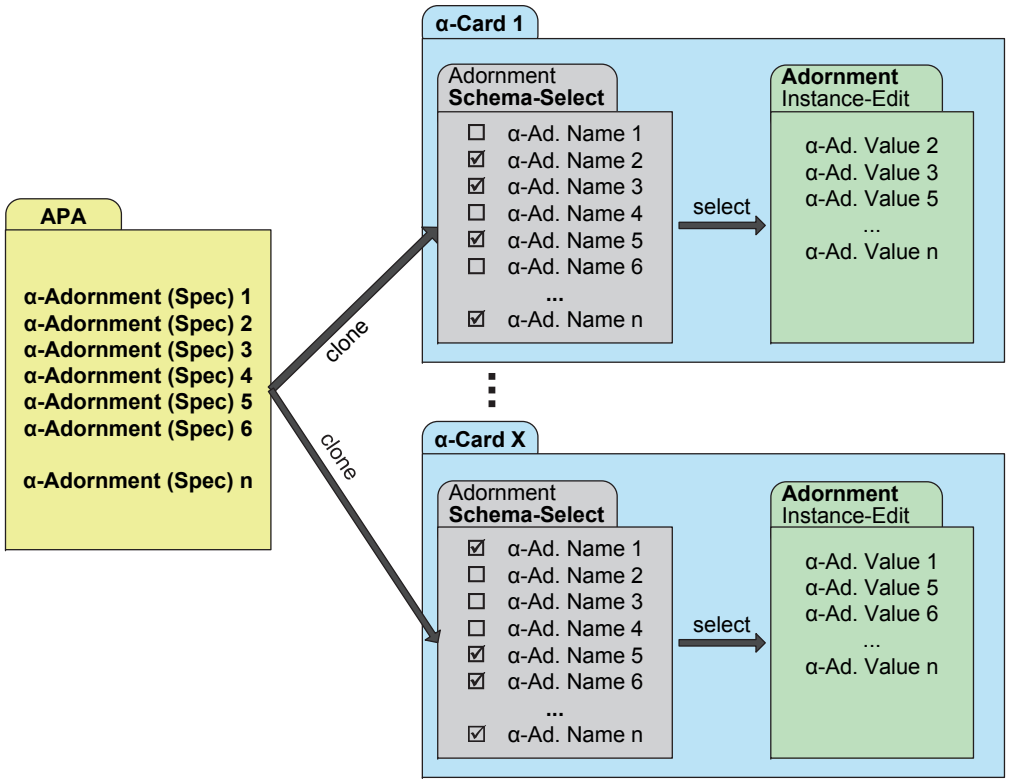

Figure 7.8: The APA in clone-and-select relationships to *α-Card* descriptors (adapted from [330])

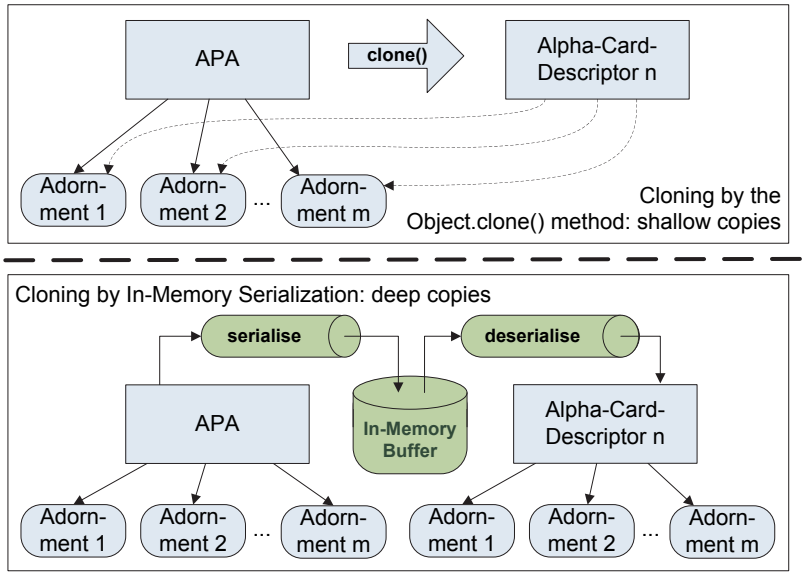

Figure 7.9: The deep-copy cloning of arbitrary Java object structures by the *α-Adaptive* subsystem using in-memory serialization (adapted from [330])

The reference model of *α-Adaptive* for run-time adaptive status attributes is outlined as UML class diagram in the appendix B by figure B.5. Finally, the *α-Adaptive* subsystem supports the concept of inheritance in form of a dangling reference to the clone base. Every time a prototype is modified, all its derived clones are automatically updated. Both the prototype and its clones can be modified at run-time in schema and in value. Prototype changes are propagated to clones but if clones deviate from their parent their specific values remain. Additional information about the *α-Adaptive* implementation is documented in [309].

7.2.3 α-Doyen: Process Role Labels and Token-Based Reassignment

Information about the participants is managed in the CRA. If a new participant is invited by handing him or her over an *α-Doc* replicate, the participant will provide basic information like his post-box information at first execution on an unknown node. Nodes are currently identified by the MAC address of the first network interface card of the system. For later changing the participant information, the *α-Doyen* subsystem provides a basic editor for the CRA. The *α-Doyen* subsystem has been implemented by Christian Hunsen for his master thesis [333]. A screenshot is provided in the appendix B in figure B.3.

At the moment, the participant master data is essentially reduced to unique identifiers for actor and institution as it has been announced in section 1.4.2. It would be possible to allow additional fields like telephone number or address, easily. However, it seems questionable whether participants would like to enter their address information for each case file. Instead, the information should be provided by the local system context. Support could be given to add vCards as attachments to each participant entry, at the same time implementing a vCard viewer as component of the CRA viewer. In order to automate the attachment of the vCard to each *α-Card*, an according system environment variable could be defined that could provide a file system reference to a locally available vCard. Then, if a user participates in an *α-Doc* case for the first time, the user could just confirm the usage of the pre-selected vCard file as his contact information. Yet, the vCard use case has not been implemented and is currently out of scope.

In addition to the master data, the *α-Doyen* subsystem manages process role labels. Currently there are three process roles supported. The *process initiator* label is reserved for the user who has created the first *α-Doc* replicate of the *α-Episode*. The label cannot be transferred. The second process label is *patient contact*, which is a self-administered role, and it is set for each contributor by default. There are some participants like the pathologist, who have no direct contact with the patient. Thus, they can deselect the

patient contact role label. The third process role label is *process coordinator* (or *primus inter pares* or briefly *doyen*). This process role can be transferred between participants like a token. The dialogue to transfer a process token is also illustrated by the screenshot in appendix B (fig. B.3). All process role labels have currently no operational semantics. They provide just visual labels for participants and their purpose is to get a quick overview on the set of participants.

An additional participant-related challenge is to know whether a participant has already received some latest update of an *α-Card*, for example, a newly contributed payload. Thus, the *α-Doyen* subsystem handles receipt acknowledgements for each *α-Card* change by any participant. For this purpose, the *α-Doyen* interacts via the *α-Kernel* with the *α-OverNet* messaging facilities. The management of these receipt acknowledegments is quit sophisticated because it has to consider both the payload versioning and logical version clock for descriptor changes. The data model and the implementation is documented in [333]. The ultimate objective is to provide the users with a visual feedback. Thus, the work-list dashboard is extended by the *α-Doyen* with a receipt acknowledgement panel that displays an icon for each actor and colours it green if a receipt acknowledgement is already available or colours the actor icon grey if no acknowledgement has been received. A screenshot in figure B.4 of appendix B illustrates the visual receipt acknowledgement indications in the bottom right of the work-list dashboard.

7.3 Facilities for Infrastructure Concerns

The infrastructure layer subsumes the subsystems *α-OverNet* and *α-OffSync* as well as the *α-VVS*. The *α-OverNet* and *α-OffSync* provide data transfer between *α-Doc* replicates. They implement a synchronization protocol as well as a join protocol. The *α-VVS* provides local storage, which supports versioning.

7.3.1 α-OverNet & α-OffSync: Synchronization and Join Protocol

*α-Doc*s are files on the participant's desktop and active only when opened. In terms of networking, all nodes have an offline characteristic and usually no two peers are online at the same time. The challenge for *α-OffSync* is to provide generic concepts for synchronization that is offline-capable such that locally conducted synchronization operations provide global consistency across all physically distributed but logically centralized replicas. The *α-OverNet* & *α-OffSync* subsystems have been constructed by Andreas Wahl for this bachelor thesis [334] and have been published in [335].

Creating a synchronization concept for *α-Flow* requires understanding of several foundational requirements and possible issues due to communication anomalies. The communication channel is supposed to use store-and-forward to deliver messages once recipients are reachable. A so-called Non-FIFO[12] behaviour of the channel is tolerated as messages may be delayed in transit or arrive out-of-order.

Due to these preconditions, common synchronization mechanisms, based on mutual exclusion or on other techniques that require reaching a global consensus between participants (e.g., [336]), are not sufficient for the *α-Flow* system. A suitable protocol must detect global conflicts, but for reconciliation, a local decision must suffice because further communication is not possible with all other nodes being offline. A decision about a conflict-free version must be derived instantly otherwise the local human actor is blocked. Local reconciliation must ensure global consistency among all nodes: it provides a globally unified partial order of the versions. The synchronization adopts lists of logical timestamps, inspired by vector clocks [337, 338] and version vectors [339]. Further synchronization approaches such as Independent Updates [340] or Timestamped Anti-Entropy [341] had been evaluated, which is documented in [334].

A so-called Adaptive Vector Clock (AVC) has been implemented as a new data structure for dynamic lists of logic clocks and an offline-capable protocol for synchronization of dynamic nodes in a totally partitioned network. Each AVC holds timestamps in an associative array, which contains one key for each locally known node in the overlay network. Associated counters indicate how often which actors have modified an *α-Card* from which network nodes. Each modification increments the individual counter for the change-originator. Applying logic clocks is necessary to establish an order on change events of an artefact. This order is based on element-wise comparing the AVCs received from other nodes with the locally known AVCs. The pair-wise relation between AVCs is equivalent of the *Happened-Before-Relation* that has been defined by Lamport[13] in [342] and AVCs fulfil the *weak clock consistency condition*, i.e. a partial causal ordering can be inferred from the clock vectors. The resulting relation between two AVCs is outlined in figure 7.10 for illustrative purposes, the AVC data structure and order relation has been formally defined in [334].

If an incoming version orderly succeeds the latest locally known version it is persisted at the latest position in the VCS history. Gaps in an artefact timeline can be computed from AVCs and indicate the existence of further versions being delayed in transit. The

12 First-In, First-Out. Reliance on Strict-FIFO channels would imply that no out-of-order messages could occur.

13 In 1978, Leslie Lamport authored *Time, Clocks, and the Ordering of Events in a Distributed System* [342] and provided first concepts to establish some order on a set of events in a distributed system based on logical timestamps.

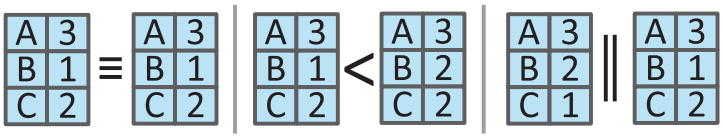

Figure 7.10: The partial ordering relation between AVCs of the *α-OffSync* subsystem (adapted from [334])

out-of-order scenario is illustrated as the left half of figure 7.11 and can be handled without problems. If a concurrent modification is detected then a global conflict has occurred. A local reconciliation has to ensure a successor state that is conflict-free. The reconciliation concerns the logical timestamp and the affected content. The successor state of the AVC can be determined by the element-wise maximum of two AVCs, which is illustrated as the right half of figure 7.11. The reconciliation of the affected content by *α-OffSync* applies a Version Control System (VCS).

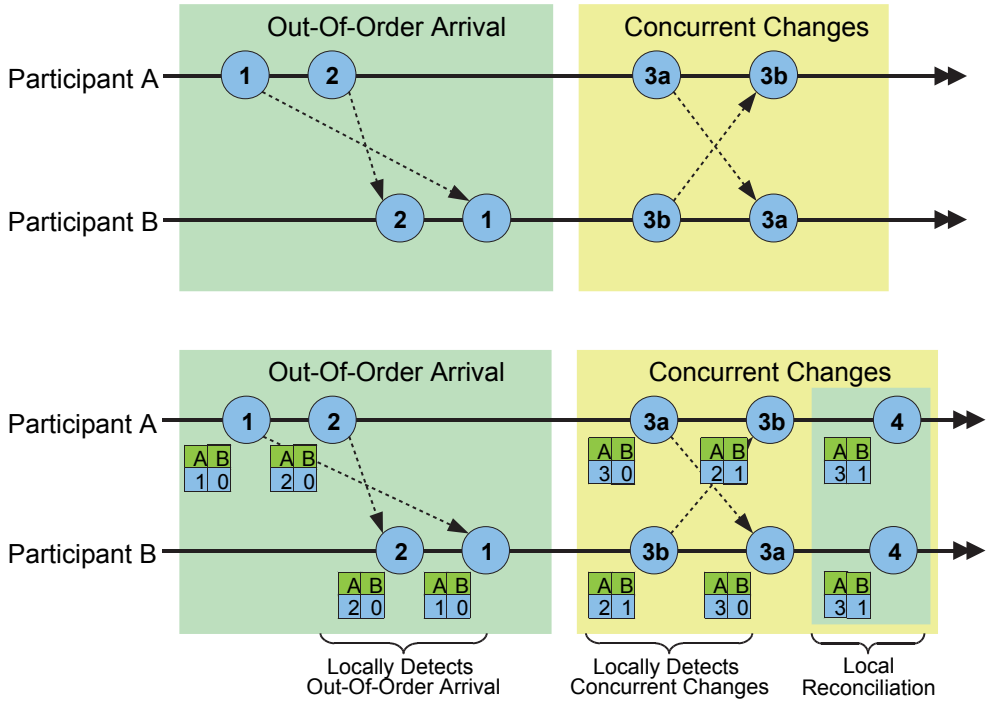

Figure 7.11: The concurrency issues is distributed scenarios and AVCs of the *α-OffSync* subsystem for detection (adapted from [334])

The *α-OffSync* protocol requires a VCS that supports artefact-specific logical timelines together with history manipulation capabilities. The out-of-order situation is handled by creating *empty versions* as placeholders for delayed messages. If a delayed message finally arrives, the version history is navigated back to the right position and the placeholder

version is filled. The α-*VVS* as VCS has been specifically designed and implemented to provide the necessary operations.

In addition, the α-*OffSync* protocol requires the VCS to provide validity-based version markers and navigation paths. Concurrency-related validity, in the context of the infrastructure layer, is not related to the intent validity that is articulated with an adornment of the α-*Card*. The concurrency-related validity is calculated by AVC comparison. Conflict-free versions are considered valid, in this context, and globally conflicting versions are considered invalid. In case of a global conflict, the α-*OffSync* protocol queries the local version repository and computes the last globally non-conflicting version (LGNCV). All versions between the LGNCV and the conflict-causing one are invalidated, using validity-based markers that are provided by the VCS. For data provenance purposes, no version is deleted and globally conflicting versions are locally added to the history in the order in which they have been received. For version history navigation, two paths are available, the *system path* that links through all versions, without distinguishing between valid and invalid ones, and the *valid path* that links only through valid versions. The implication of the AVC-based version ordering on the VCS-managed version history is illustrated in figure 7.12.

For the reconciliation of the content a prime strategy is required that guarantees a globally consistent state without further communication for arbitrary content types independent of the content semantics or its format. Thus, the default strategy for conflict resolution simply identifies the LGNCV and resets the content of the latest version to the historical conflict-free state. After the globally conflicting versions have been marked invalid, earlier, the LGNCV is easily available by the valid path navigation. The illustration in figure 7.12 outlines the relationship between the LGNCV, which is node No. 2, and the latest version, No. 4, that has been created for conflict resolution.

The reset strategy would be inconvenient for the users if concurrent changes are assumed to occur frequently. However, the α-*Card*s are the atomic units of synchronization and each has a dedicated organizational accountability. Thus, a conflict for a content card is only possible if the same participant changes an α-*Card* using multiple α-*Doc* replicates at different nodes. Conflicts for coordination cards like the PSA are more likely because the work-list is shared and is generally edited concurrently. However, changes to the case file, in healthcare, appear primarily in a temporal period that approximates a patient visit. Patient visits occur widespread over time. For this reason, the overall probability of any parallel conflicts is low. The purpose of the α-*OffSync* protocol is to formally guarantee the detection of conflicts and to guarantee the reconciliation in a conflict-free state. The reset strategy is the single strategy that can be generically applied. The conflicting versions are not lost but are available to the user within the VCS and can be used to manually compare and merge the content on demand. The discussion section

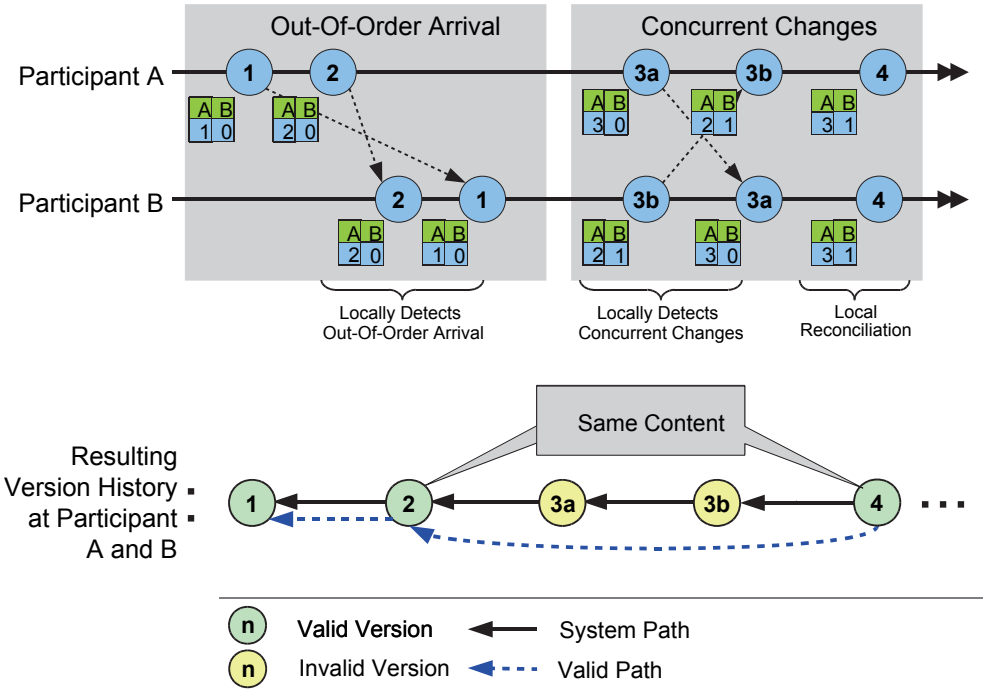

Figure 7.12: The reconciliation of concurrency issues by the *α-OffSync* subsystem with the support of a versioning system (adapted from [334])

8.5.1 will outline some considerations about improved reconciliation strategies, using XML-based difference calculation and automatic merging.

As it has been indicated earlier, Simple Mail Transfer Protocol (SMTP) & Internet Message Access Protocol (IMAP) are preliminarily used for store-and-forward communication. Each actor can use multiple replicates at different nodes with the same IMAP account. *α-OffSync* encodes the *α-Episode*, *α-Card*, actor, and node information into the email header. In fact, the complete messaging of an arbitrary number of case files and actors can be handled with a single IMAP account. More details about the message format and message identification partitioning for multitenancy support is provided in [334]. In the future, the communication platform may be substituted by other protocols being more reliable in terms of guaranteed delivery. The "Elektronische Gesundheitskarte" (eGK) messaging infrastructure would be required for healthcare scenarios. In non-healthcare scenarios, the Extensible Messaging and Presence Protocol (XMPP) would be an option.

To enable dynamic change of actors, nodes, and AVCs (now in terms of cardinality), a join protocol forms a second part of the *α-OffSync* protocol. New copies of an *α-Doc* introduce themselves to their peers and automatically synchronize by mutually exchanging information about locally known *α-Card*s based on their AVCs. New nodes are incrementally updated to the latest state. Each participant gains knowledge about newly

joined actors. The protocol is required to support multiple participants that join in parallel ("N-ary join"). The necessary messages for a scenario with two new participants are outlined in figure 7.13. The illustration also motivates the adaptiveness of the AVC vector clocks because the vector clocks adapt to the dynamically changing number of participating *α-Doc* replicates.

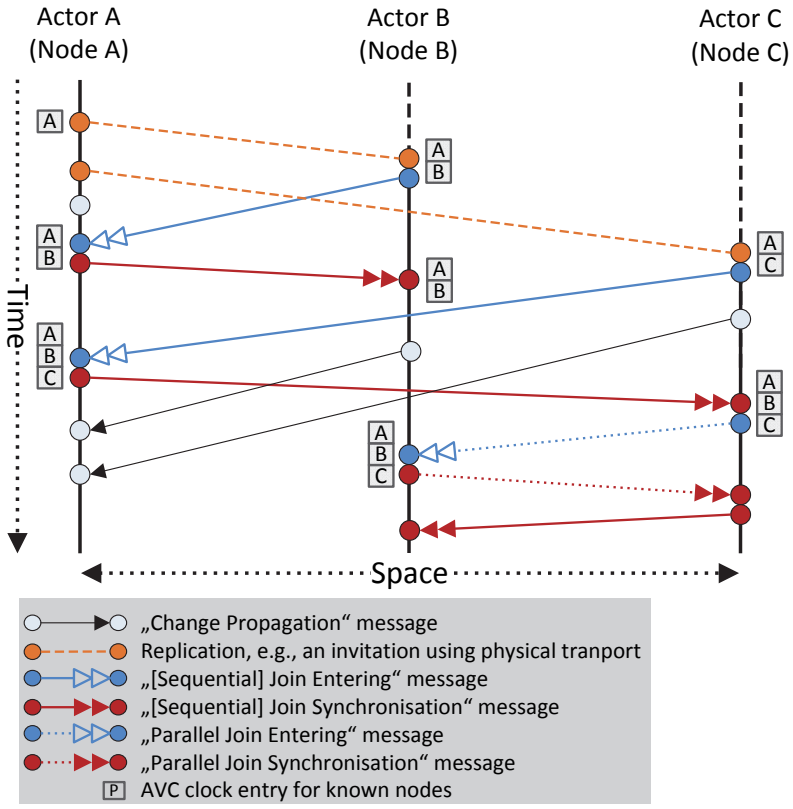

Figure 7.13: The join protocol messages for two participants that join in parallel ("N-ary join") by the *α-OffSync* subsystem (adapted from [334])

Four types of join messages are required. One pair of *entering-* and *synchronization-*message types for *sequential joins*. Sequential joins are scenarios in which only a single new *α-Doc* replicate joins the existing collective, one at a time. However, it is possible that joins are initiated by multiple new *α-Doc* replicates in parallel. A second join phase is required for *parallel joins*, to ensure that every replicate finally knows about all other replicates. The parallel join phase uses a second pair of entering- and synchronization-message types. The messages of type *entering* inform all known nodes about the existence of the new replicate at a new node. In addition, an entering message contains AVCs of all locally known *α-Card*s. This information is used by the receiving nodes to calculate whether the new replicate misses *α-Card*s or updated *α-Card* versions that have occurred in the meantime. The messages of type *synchronization* respond to entering requests and

carry any missing content such that the joined node becomes up-to-date. The purpose of the message types of the second phase is similar. In this phase, the nodes that joined in parallel interact with each other and synchronize. The protocol state machine for N-ary joins, as it is implemented by the *α-OffSync* subsystem, is illustrated in figure 7.14. The colours and shapes of the protocol machine reflect the message types used in figure 7.13. The complete scenario is described in [335], as is the format and semantics of the messages and the protocol implementation. Leaving a case is much simpler than joining, it only requires the deletion of the electronic post-box information from the CRA. The deletion is automatically propagated to the other nodes by standard synchronization but subsequently they stop synchronization messages on this post-box. The removal of the electronic post-box information affects all nodes or replicates of an actor. In the following, the user can delete all of his *α-Doc* replicates.

A unique characteristic of the introduced synchronization approach is the ability to establish a shared view on the process state among all participating actors in totally partitioned networks, where no guaranteed assumptions about the reachability of any network nodes can be made. Global conflicts can be detected and reconciled without additional online message exchange for determining a globally valid version. *α-Flow* is enabled to facilitate the management of dynamic groups of participants by minimizing the effort for inviting new actors and joining an ongoing treatment episode. Thus, it provides the necessary flexibility for inter-institutional processes.

7.3.2 α-VVS and Hydra: Multi-Module Version Control System with Validity-Awareness

From the perspective of VCS, the *α-Doc* contains a repository that is structured into logical units, each logical unit is the equivalent of an *α-Card*, which is an independent set of files. *α-VVS* provides embedded versioning for the *α-Flow* engine within an *α-Doc*. The versioning library of *α-VVS* has been implemented as an autonomous component, the so-called Hydra VCS. The unique functional features of Hydra are 1) multi-module support and 2) validity-awareness. The *α-VVS* subsystem and the Hydra VCS library have been constructed by Scott Hady for his master thesis [343]; Hydra has been published in [344].

In paper-based healthcare processes, logical units of paper artefacts have an independent history. An electronic equivalent with versioning support should preserve independent histories for data provenance purposes. Thus, each logical unit requires its independent VCS history, however, the overall team progress, i.e. data production over all logical units, must also remain track-able. This is approximately the same situation as in parallel software development with conflicting updates and with grouping artefacts like source

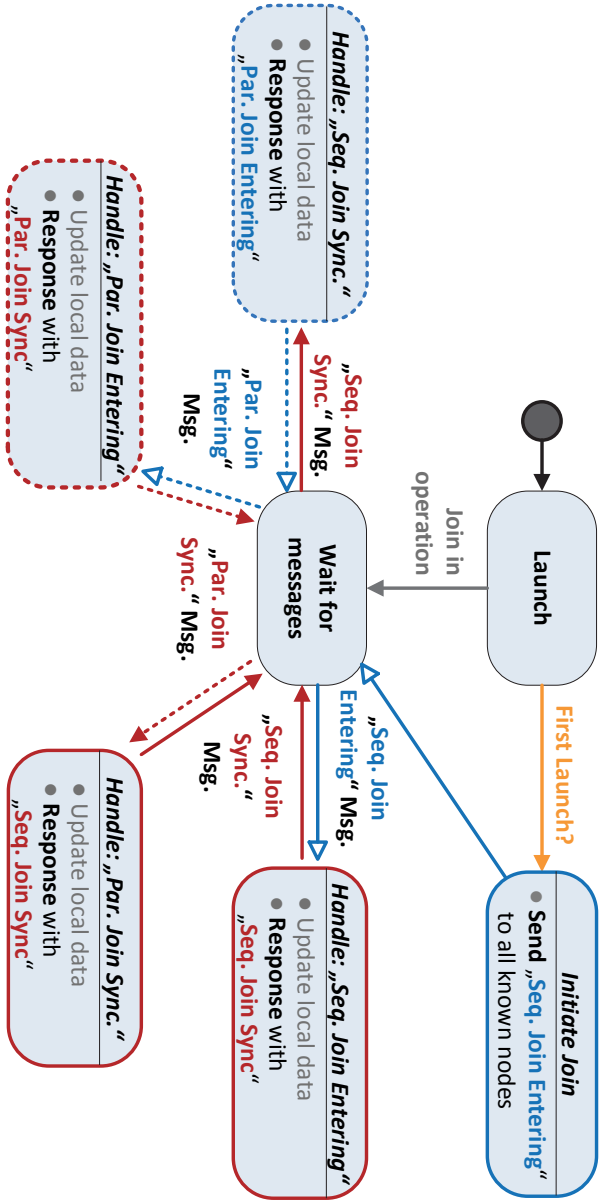

Figure 7.14: The join protocol state machine for N-ary joins as it is implemented by the *α-OffSync* subsystem (adapted from [334])

code files into logical units of software modules. Thus, the "logical unit" (LU) is a unified term for "data module" or "software module".

The Hydra objective is to provide a generic VCS concept for (1) managing multiple LUs within a single repository. The LU is defined as an arbitrary set of hierarchically structured files. Within a single repository, both (i) an independent version history, navigation, and checkout reference for each head must be kept for each LU, and (ii) a common version state over all LUs must be provided for module-interdependency maintenance. The first objective is illustrated in figure 7.15, contrasting the (independent) *multi-modularized-heads* versioning with the *single-head* versioning of mainstream VCSs.

The version history of modules are mingled in a single-headed VCS, even if modules are separated by subdirectories, as long as they are managed by a single repository. Updating several modules to the head version but letting other modules remain in a concerted version state becomes cumbersome and requires user discipline. The concept of branching actually results in multiple heads but this is a non-modularized history because branches assume a common version ancestor. Thus, branches are semantically different to multi-modularized-heads, which provide independent module versioning. Tracking the composite project state as the vector-based combination of the LU version histories is only possible with independent heads for each LU.

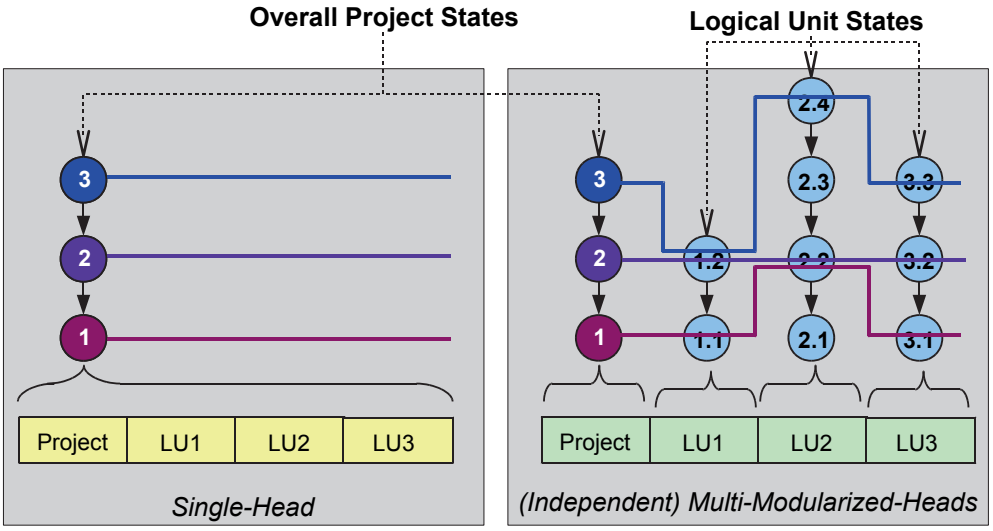

Figure 7.15: The multi-module versioning of the α-*VVS* subsystem (adapted from [343])

The second Hydra objective is to allow for (2) labelling versions by a valid / invalid flag and to enable *validity-based version navigation*. This means to provide both (i) a *system path navigation* with all version states as it is provided by common VCS navigation and (ii) a *valid path navigation* that operates only on all valid versions. The validity facilities of Hydra are primarily required by the α-*OffSync* protocol that implements optimistic concurrency control as it has been outlined in section 7.3.1. Hydra facilitates the protocol implementation: in case of global conflicts, the conflicting versions can simply be marked as invalid. The invalidated versions remain accessible by Hydra's system path navigation, and can be used for manual user-managed reconciliation. The valid path provides eased navigational access to the latest globally valid (i.e. non-conflicting) version.

The Hydra versioning is inspired by Git (e.g. [345]) and its object model[14]. Hydra also adopts full copy storage and content-addressable storage via hashing from Git. In the *α-VVS*/Hydra project, up-front, we analysed the history and evolution of versioning systems (e.g., [346]). The Source Code Control System (SCCS) by Marc Rochkind invented a mainframe-based concept of repository with forward deltas and versions; by its lock–modify–unlock workflow it did proactively prevent contradictory changes, i.e. pessimistic concurrency control [347]. Then Revision Control System (RCS) by Walter Tichy supported the concept of the working copy, applied reverse deltas for reducing the checkout delay, and allowed for branching, still in a mainframe environment. Tichys 80s publication [348] is a profound analysis on parallel development, customer modifications, supporting temporary fixes, and conflicting updates. Then Concurrent Versions System (CVS) provided a solution for client-server environments in 1986. CVS implemented optimistic concurrency control instead of pessimistic locking; resulting in a copy–modify–merge workflow. It also introduced the treatment of a coherent set of artefacts as a single unit of version progress. Up to CVS, the prime intent was for text data storage only, and histories were kept on individual files and not on the overall tree, e.g., breaking history when moving files. For a CVS coherent set no concept of transaction is available, thus, it is possible that the check-in operation will be completed only for some files. Then Subversion, or Subversion (SVN), was introduces in 2001, again for client-server environments. SVN provided a common repository version number and change sets with atomic commits, i.e. change sets are checked-in either all or nothing. In addition, SVN specifically allowed for all file types and eased versioning of binary files. In the mid-2000s, a number of new distributed Version Control System (dVCS) systems were introduced, like Monotone, Mercurial, and Git. They employed a peer-to-peer repository paradigm, in which each peer repository maintains its own redundant copy of the data and maintains its own perception of the artefact's evolution; each repository provides collaborative services to other peer repositories, resulting in a pull-copy-modify-merge-push workflow. Best practice for managing peer-based collaboration is designating a so-called blessed repository that may only be updated by designated individuals, the so-called integration managers. Blessed repositories are updated after reviewing and accepting proposed changes, resulting in a update/pull–modify/push–propose/pull–accept/push workflow, which is illustrated in the appendix section B.3 by figure B.6.

The dVCS scenario requires two repositories to be online at the same time. Synchronization operates between two VCS nodes, and push-based synchronization is only supported unilateral and not multilateral for a group of nodes. The *α-Flow* cannot assume two *α-Doc* replicates to be online at the same time. Thus, *α-OffSync* provides offline-capable synchronization and provides synchronization between multiple nodes.

14 e.g. `http://eagain.net/articles/git-for-computer-scientists`

Support for the independent versioning of multiple modules is missing even if it is required in any modular data architecture. For substitution of this feature, modules are sometimes separated in distinct repositories. Then, supporting a "super-project" repository requires mechanisms to reference distinct external repositories and to virtually merge them into a single working space[15]. The distinct-repositories approach does not provide a module-comprehensive version state. In particular, it encumbers restructuring between the modules and interrupts version history at relocations. In contrast, a VCS implementation with multi-modularized-heads needs to provide three levels of versioning granularity: the atomic artefacts, the logical unit, and the overall repository.

Hydra extends the Git object model, which is illustrated in figure 7.16. The original model consists of the class Object with subtypes Commit, Tree, and Blob as well as Reference as a named relationship. Trees and blobs form a hierarchical structure; trees have reflexive parent associations. Commit objects link to multiple trees; additionally, commits have reflexive previous associations. The Blob has a Unique Identifier (UID) via an SHA-1 fingerprint on the binary content. The Tree content is a composite (a map data structure) of its sub-trees and artefacts; it is recursively calculated. The UID of a Tree is the fingerprint on this content. The Commit object represents a committed snapshot and stores metadata about the commit.

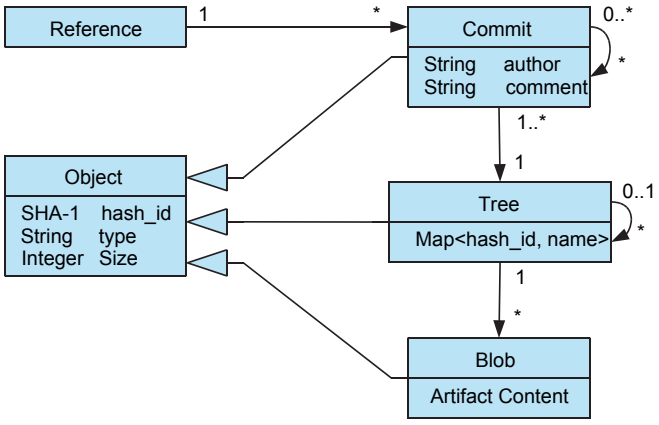

Figure 7.16: The Git object model (adapted from [343])

We reimplemented the versioning object model in Java. First, we refined the Git Reference class into subtypes LogicalUnit and Stage. The stage references LUs, each with an arbitrary state, thus, the stage manages the module-interdependency and represents concerted super-project progress. The Git Commit class is refined into two according

15 For such purposes, SVN provides the `svn:externals` property and Git similarly offers the `git-submodule` commands. Both system extensions have several documented pitfalls (e.g., `http://book.git-scm.com/5_submodules.html`).

state types, Stage State and Logical Unit State. The differentiation between references and states is necessary because logical references like head and current successively point at different state objects. A simplified illustration of the meta-model is outlined in figure 7.17. The Container is the equivalent to the Git Tree but the sub-structures of the container are skipped in figure 7.17. The comprehensive object model of Hydra VCS is illustrated in the appendix section B.3 by figure B.7. Validity tracking is implemented by adding a second reflexive validPrevious association between Git Commit classes, i.e. both Hydra state classes. A detailed Hydra description is provided in [343].

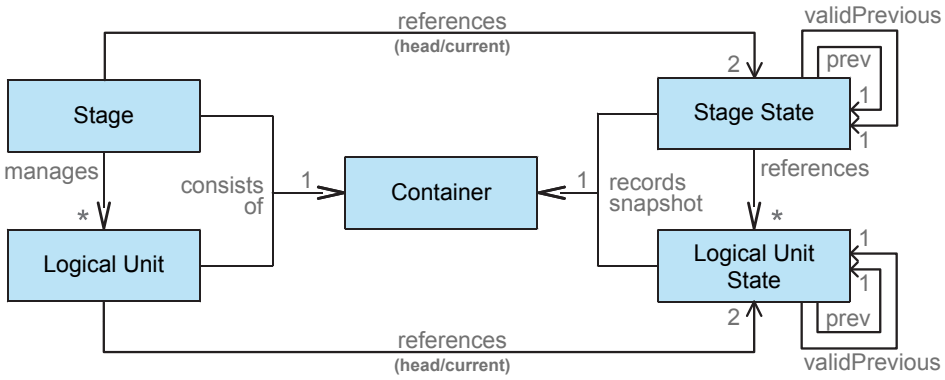

Figure 7.17: The meta-model of the Hydra VCS subsystem (adapted from [343])

The Hydra library is also made available as a stand-alone VCS application that is extended by a command-line interface. The *α-VVS* module is, similarly to the command-line interface, a wrapper for the Hydra library. *α-VVS* and Hydra implement an embedded multi-module and validity-aware versioning. Its executable size is 208 kb instead of Git's 19 MB. A stress test in section 8.4 will compare basic performance characteristics of Hydra VCS with SVN and Git.

7.4 Local System Integration

Integration of *α-Doc*s with local systems is a challenge. Based on the document-oriented paradigm the systems interaction between a local workflow engine and the distributed Document-oriented Process Management (dDPM) engine inside the *α-Doc* has the same semantics as the user interaction: it basically consists of put-card/get-card, put-payload/get-payload, and set-adornment (cf. chap. 4, fig. 4.1, p. 155). A command-line interface to the *α-Doc* may be used for the interaction of a local workflow systems or Clinical Decision Support System (CDSS) (cf. sect.3.1.3) with an *α-Doc*. For example, a CDSS can suggest two different types of decisions, domain-oriented decisions (like a

differential diagnosis), and process-oriented decisions (like a therapy plan). Domain-oriented decisions can be communicated via a payload of an *α-Card*. Process-oriented decisions can be articulated by creating *α-Card* descriptors as requests for the necessary treatment steps.

A dDPM hypothesis for paper-based inter-institutional cooperation is "anything that must be exchanged must be printable". The simplest form of content contribution would be to use a freely available Portable Document Format (PDF) printer-driver and to extract the inter-institutionally relevant paper-based information into an electronic equivalent PDF file (cf. sect. 2.2.2). However, this requires the user to determine a temporary file location for the PDF output. Then, the user needs to navigate to the PDF file location. Next, he or she needs to drag-and-drop the PDF file onto an *α-Doc*. This sequence can be improved; thus, we developed *α-PrintPut*.

α-PrintPut: Windows Printer-Driver for "1-Act Contribution"

The objective of *α-PrintPut* is to provide a direct interaction between any MS Windows application that allows printing and an *α-Doc*. The Windows platform is supported because it is the most popular one. The *α-PrintPut* provides a Windows printer-driver in which the user does not select an output PDF file but an existing *α-Doc*. The *α-PrintPut* temporarily creates a PDF file, integrating an open-source PDF printer library, and hands the file over to the user-selected *α-Doc* per command-line interface. The elimination of the user-handled PDF file creation results in a kind of user experience that is considered as a "1-Act Contribution", in analogy to the Amazon "1-Click® Ordering" [349]. The *α-PrintPut* system was constructed by Konstantin Tsysin for his bachelor thesis [350].

The *α-PrintPut* is developed in C#. It is not part of the *α-Flow* engine; instead, *α-PrintPut* is a MS Windows printer-driver. The primary challenge for *α-PrintPut* is that a printer-driver necessarily runs with system privileges but the execution of the *α-Doc* should be performed with user privileges. The construction of a PDF printer-driver requires expert knowledge about the Windows printing architecture. We used the open-source tool PDFCreator[16] as primary reference for implementing *α-PrintPut*. Hence, *α-PrintPut* creates the PDF file using a natively provided PostScript driver, together with RedMon[17] that captures the resulting PostScript data stream as print monitor. RedMon allows for redirecting the PostScript into a GhostScript library for PDF conversion, which finally results in a temporarily created PDF. In contrast to tools like PDFCreator, the *α-PrintPut* has to hand its result to the *α-Doc* as a Java application. *α-PrintPut*

16 http://sourceforge.net/projects/pdfcreator/
17 http://pages.cs.wisc.edu/~ghost/redmon/

supports Windows Vista and Windows 7[18] because it uses the Windows User Account Control (UAC) facilities to prepare the user-based security-context for the Java-driven *α-Doc* execution. Konstantin Tsysin's thesis provides the technical details [350]. In the end, the *α-PrintPut* supports the use case of the 1-act contribution, for contemporary Windows systems. *α-PrintPut* is bundled in form of an autonomous Windows installer.

7.5 Summary

This chapter has provided an overview on the implementation challenges for constructing a distributed case file and case handling engine in form of an active document. Various methodical and technical aspects have been discussed. The *α-Flow* implementation is of considerable size, thus, only issues of general interest have been illustrated. All *α-Flow* subsystems have been outlined and the design choices have been characterized.

18 *α-PrintPut* supports both 32bit and 64bit systems.

8 Evaluation of the Implementation

> “In theory, there is no difference between theory and practice. But, in practice, there is.”
>
> (Jan L. A. van de Snepscheut)

This chapter provides a technical evaluation of the implemented *α-Flow* system and its subsystems. The first section outlines the hard disk footprint of the system as well as library dependencies. Code metrics are applied to measure the quality of the system implementation. The code value is estimated by applying a parametric cost model. A performance benchmark test is outlined for the storage subsystem. Finally, current system limitations of the *α-Flow* implementation are discussed.

8.1 Executable Artefacts & Hard Disk Footprint

The implementation efforts in the context of this thesis are released as three artefacts that can be used as autonomous applications. The *α-Flow* engine, the Hydra Version Control System (VCS), and the *α-Forms*. Each artefact is a single executable Java Archive (JAR) file. The three artefacts are completely independent of the others at run-time. Yet, the *α-Flow* engine embeds Hydra VCS and *α-Forms* as libraries. However, Hydra VCS and *α-Forms* can also be used on their own. Table 8.1 provides an overview of the three executable artefacts and their file size[1]. The build automation is based on the Apache Maven™ build system[2].

The primary artefact is the *α-Flow* engine. The *α-Forms* module as an autonomous artefact independently fulfils the active document metaphor. It provides both the form composing and form editing capabilities for a single form instance. Hydra VCS can be

1 All technical evaluation is based upon the internal Subversion (SVN) project repository, rev. 3853, 12th June, 2012

2 `http://maven.apache.org/`

System name	Build tool	Artefact name	JAR file size
α-*Flow* engine	Maven	alph-o-matic-1.0.jar	14.73 MB
Hydra VCS	Maven	hydra-cli-1.0.one-jar.jar	208.13 kB
α-*Forms*	Maven	alphaforms-1.0.one-jar.jar	264.76 kB

Table 8.1: Deployment artefacts of the autonomous α-*Flow* applications

used as a text console application that provides a command-line interface with commands similarly to a `cvs`, `svn`, or `git` executable. The Hydra command-line interface provides self-documentary usage help, all commands and their usage are explained on the console at run-time. In addition, Scott Hady has documented the Hydra usage and commands in the appendix sections[3] A.2 to A.6 of his thesis [343].

Build Configuration

The project uses a Maven™ multi-module configuration. Table 8.1 references the build module, which is the subdirectory of the source code repository that contains the Maven configuration for bundling the artefact as an executable JAR file.

The One-JAR™ tool[4] is used for packaging all dependent libraries into a single deployable and executable unit (cf. sect. 7.1). The Hydra VCS and α-*Forms* artefact names accordingly have an "one-jar" infix. The artefact for the α-*Flow* engine is technically also a One-JAR™ but its artefact name is configured to be spelled as "alph-o-matic".

Transitive External Dependencies

The α-*Flow* engine uses several external libraries. All of them are included in the executable artefact, resulting in the hard disk footprint in table 8.1. Notably, both Hydra VCS and α-*Forms* have none external dependencies at all. All transitive dependencies, each with its version and according JAR library file size, are listed in table 8.2. Amongst others, the list of libraries will be relevant for the discussion in section 8.5.7.

The list is alphabetically sorted by the Maven groupId that uniquely identifies the original developer group. The logback libraries as well as the slf4j libraries are for logging purposes. The logging facilities are used by all subsystems. The jaxb, xstream, xpp3, and jsr173 libraries are for Extended Markup Language (XML) serialization, which is used by the α-*Kernel*, α-*OffSync*, and α-*VVS* subsystems as well as by the Drools™

3 `http://www6.cs.fau.de/research/projects/promed/theses/DA_sischady.pdf`
4 `http://one-jar.sourceforge.net/`

GroupId	ArtifactId	Version	JAR file size
ch.qos.logback	logback-classic	0.9.27	238.06 kB
ch.qos.logback	logback-core	0.9.27	301.81 kB
com.sun.xml.bind	jaxb-impl	2.0.3	765.82 kB
com.thoughtworks.xstream	xstream	1.2.2	364.45 kB
com.toedter	jcalendar	1.3.2	123.66 kB
javax.activation	activation	1.1	61.51 kB
javax.mail	mail	1.4.4	483.32 kB
javax.xml.bind	jaxb-api	2.0	71.37 kB
javax.xml.bind	jsr173_api	1.0	48.51 kB
joda-time	joda-time	1.6	522.29 kB
org.antlr	antlr-runtime	3.1.1	113.48 kB
org.bouncycastle	bcpg	1.47	315.79 kB
org.bouncycastle	bcprov	1.47	2.25 MB
org.drools	drools-api	5.0.1	124.19 kB
org.drools	drools-compiler	5.0.1	1.10 MB
org.drools	drools-core	5.0.1	1.71 MB
org.drools	drools-transformer-jaxb	5.0.1	12.88 kB
org.eclipse.jdt	core	3.4.2.v_883_R34x	4.16 MB
org.mvel	mvel2	2.0.10	593.79 kB
org.slf4j	jcl-over-slf4j	1.6.1	16.90 kB
org.slf4j	jul-to-slf4j	1.6.1	4.36 kB
org.slf4j	log4j-over-slf4j	1.6.1	12.07 kB
org.slf4j	slf4j-api	1.6.1	24.90 kB
org.springframework	spring-context	3.0.5.RELEASE	653.18 kB
org.springframework	spring-asm	3.0.5.RELEASE	51.84 kB
org.springframework	spring-beans	3.0.5.RELEASE	542.39 kB
org.springframework	spring-core	3.0.5.RELEASE	373.48 kB
org.springframework	spring-expression	3.0.5.RELEASE	165.77 kB
uk.org.lidalia	sysout-over-slf4j	1.0.2	43.11 kB
xpp3	xpp3_min	1.1.3.4.O	24.10 kB

Table 8.2: Transitive external dependencies of the *α-Flow* engine

rule engine. The jcalendar library provides a graphical widget for date selection within the *α-Editor* subsystem. The javax.mail library is required for messaging via Simple Mail Transfer Protocol (SMTP) and Internet Message Access Protocol (IMAP) within the *α-OffSync* subsystem. The joda-time library provides helper classes that ease date and time handling in Java. It is used by various *α-Flow* subsystems but is also required by the Drools™ rule engine. The antlr, eclipse.jdt, and mvel libraries are required by the Drools™ rule engine for parsing purposes of its Domain Specific Language (DSL) rule file format. The bouncycastle libraries are for cryptographic purposes. *α-Flow* uses it as OpenPGP implementation within the *α-OffSync* subsystem. The drools libraries

implement the rule engine within the *α-Kernel* subsystem. Finally, the spring libraries provide an inversion of control container [351] that allows for management of Java objects via dependency injection [352, 353] and that fosters system modularization.

8.2 Code Metrics

For assessing the *α-Flow* system implementation, I follow the methodology of Lanza and Marinescu in their book "Object-oriented metrics in practice" [354]. Lanza and Marinescu use a system of seven direct calculable metrics, and eight derived metrics. They also measured dozens of existing systems in Java and C++, thus, gaining an empirical reference system for the metrics. For example, these metrics are lines of code, number of classes, or cyclomatic complexity (cf. [355]). Lanza and Marinescu use abbreviations for the code metrics, which are listed in table 8.3.

Abbr.	Description
CYCLO	Cyclomatic complexity
LOC	Lines of code
NOM	Number of methods
NOC	Number of classes
NOP	Number of packages
CALL	Calls (per method)
FOUT	Fan out (number of other methods called by a given method)
ANDC	Average Number of Derived Classes
AHH	Average Hierarchy Height

Table 8.3: Code metrics and their abbreviations (adopted from Lanza and Marinescu [354])

The light yellow metrics characterize size & complexity of the system under evaluation. The direct metrics for size & complexity are CYCLO, LOC, NOM, NOC, and NOP (cf. tab. 8.3). The derived metrics for size & complexity are NOC/package, NOM/class, LOC/method, CYCLO/line (cf. tab. 8.4). The light blue metrics characterize coupling. The direct metrics for coupling are CALLS and FANOUT (cf. tab. 8.3). The derived metrics for coupling are CALLS/method and FANOUT/call (cf. tab. 8.4) The light green metrics characterize usage of inheritance. The inheritance metrics are ANDC and AHH (cf. tab. 8.3 & 8.4). Both are derived metrics because each is an average, the former over all root classes (of inheritance hierarchies) and the latter over all inheritance hierarchies.

All direct metrics will be articulated as integer numbers. All derived metrics will be articulated as floating point number. The derived metrics are indicators that are independent

of the size of the system. This is a premise for enabling an objective assessment of a system. A reference point is required. Thus, Lanza and Marinescu measured industry systems and derived empiric ranges for **average** values and thresholds for **low** and **high** values. The ranges that are given in [354] are reproduced in table 8.4.

	Low	Average	High
CYCLO/line	$x \leq$ 0.16	0.20	0.24 $\leq x$
LOC/method	$x \leq$ 7.0	10.0	13.0 $\leq x$
NOM/class	$x \leq$ 4.0	7.0	10.0 $\leq x$
NOC/package	$x \leq$ 6.0	17.0	26.0 $\leq x$
CALL/method	$x \leq$ 2.01	2.62	3.20 $\leq x$
FOUT/call	$x \leq$ 0.56	0.62	0.68 $\leq x$
ANDC	$x \leq$ 0.25	0.41	0.57 $\leq x$
AHH	$x \leq$ 0.09	0.21	0.32 $\leq x$

Table 8.4: Derived code metrics: empiric industry ranges for Java projects (adopted from Lanza and Marinescu [354])

Assessment of the *α-Flow* Implementation

For measuring the direct metrics of the *α-Flow* implementation, a tool named iPlasma [356] has been used. A notable aspect about the methodology of Lanza and Marinescu is the presentation of the metrics in form of an "*overview pyramid*", which has "steps" in its two-dimensional triangular form. Table 8.5 provides this pyramid-style presentation of the code metrics for the *α-Flow* implementation.

				ANDC	0.07		high
				AHH	0.20		avg.
			4.45	NOP	101		low
		7.95	NOC		450		
	10.62	NOM			3580	NOM 3.56	
0.13	LOC				38027	12745 CALL	0.57
CYCLO					5273	7265	FOUT

Table 8.5: Pyramid-style presentation of the code metrics: implementation details of the *α-Flow* engine

The metrics for size & complexity and for coupling form the basis of the pyramid, the metrics for the usage of inheritance are on top of the pyramid. In the middle of the pyramid the integer-based direct metrics are listed. For example, the number of classes is 450 and the number of methods is 3580.

The derived metrics are written at both "edges" of the pyramid. For example, the cyclomatic complexity per line is 0.13 and the number of lines of code per method is 10.62. Similarly on the other side of the pyramid, the fan out per call is 0.57 and the calls per method is 3.56. The values for the inheritance metrics on top of the pyramid are calculated directly by the tool.

The derived metrics are coloured according to the reference by Lanza and Marinescu for Java projects (cf. tab. 8.4). For example, the cyclomatic complexity per line is below the threshold of 0.16. Thus, it is considered a low value and it is coloured in blue. The number of lines of code per method is between the thresholds 7.0 and 13.0. Thus, it is considered average and it is coloured in green.

Interpretation of the Overview Pyramid of the *α-Flow* Implementation

The derived metrics allow for an objective assessment of a system. The overview pyramid of the *α-Flow* implementation can be interpreted according to the reference system by Lanza and Marinescu. Hence, the *α-Flow* implementation can be described as follows: *Class hierarchies* tend to be of **average height** and to be **narrow** (i.e. inheritance trees tend to have base-classes with few directly derived sub-classes). *Classes* tend to contain an **average** number of methods. Classes also tend to be organized in rather **fine-grained packages** (i.e. few classes per package). *Methods* tend to be **average** in length and having a rather **simple logic** (i.e. few conditional branches). Methods also tend to call **many methods** (high coupling intensity) from **few other classes** (low coupling dispersion).

In conclusion, the only critical value relates to the high coupling intensity. However, the low coupling dispersion makes this a manageable aspect. Finally, from the perspective of Lanza and Marinescu's industrial reference, the design and implementation of the *α-Flow* engine can be considered in harmony. This *α-Flow* system trait is a necessary precondition for applying an industrial model of cost estimation to the project's code base.

8.3 Code Value

The so-called *substitution costs* are an estimation of efforts in person-months for a given project. It describes how much development time and costs would be necessary to substitute an implementation with an own one. The substitution costs measure the value of a project. The question about substitution costs has been discussed in the

context of open-source software. For example, Wheeler uses the COnstructive COst MOdel (COCOMO) [357] to estimate the value of the GNU Linux Kernel in [358].

The COCOMO model is a form of parametric model. Its purpose is to provide a statistical reference for an a priori prediction of software development projects. Parametric models are built up using regression analysis on available, empirical data. Other parametric models for cost estimation are, for example, the Parametric Review of Information for Costing and Evaluation – Software (PRICE-S), cf. [359], or the Software Evaluation and Estimation of Resources – Software Estimating Model (SEER-SEM) [360].

In 2006, the European Commission's Directorate General for Enterprise and Industry commissioned a study for calculating substitution costs for open-source systems. Amongst others, the executing authority applied COCOMO in the context of several case studies. One key indicator that is calculated by COCOMO are person-months. In order to calculate a monetary value the study declares: "Salaries are used to compute the value of primary production [..], mapping person-months to monetary values by multiplying them with appropriate salary levels" [361, p. 48]. The study warns that COCOMO-like estimation techniques are designed for classical software generation processes and that the results that COCOMO gives when applied to open-source implementations should be viewed with caution. There are some fundamental differences between open-source development and classical software development (e.g., [362]) and the statistical calibration of COCOMO-like models did neither consider highly distributed development processes nor the influence of sporadic contributions by volunteer programmers. Creating a cost evaluation model for open-source software is an unsolved scientific problem (e.g., [363]).

In the absence of a suitable cost model, the basic application of the COCOMO model remains popular. For example, the Ohloh[5] web application provides analytic services to open-source projects, using COCOMO to calculate an estimated project value.

In academic software development projects, the programming is accomplished in the context of a PhD thesis as well as supervised bachelor or master theses. For example, I supervised twenty-four student theses. However, literature research and the writing of the thesis takes significant amount of time. Thus, the programming is not a full-time job for any of the involved developers. The theoretical fraction in contrast to the engineering fraction varies for each thesis and student project. For the same reasons, it is impossible to me to account my own contributions to the source code in person-months of software engineering. In addition, in a lab course called "SWAT"[6] another six students did implementation work in the context of *α-Flow* during summer term 2011. Hence, a

5 `https://www.ohloh.net/`

6 SWAT is a recursive acronym for "SWAT is a Web Application Tutorial"

valuation solely based on the number, type, and duration of all related projects would be without meaning. Instead, an a posteriori evaluation of the implementation efforts provides additional information about the accomplished pilot implementation.

Applying COCOMO to the *α-Flow* Implementation

As a form of disclaimer, I must stress that the COCOMO model provides only a very rough estimation of the effort needed to generate software of a given size. Since this estimation technique is designed for classical software generation processes, the results it gives when applied to academic pilot implementations should be treated with caution. However, the previous system evaluation of the *α-Flow* engine based on code metrics in section 8.2 has shown that the design and implementation can be considered in harmony from the perspective of Lanza and Marinescu's industrial reference. Thus, it seems fair to similarly apply COCOMO as an industry reference for substitution cost estimation.

Boehm introduces a simple formula for estimating the "Effort Applied" (E) in dependency of the source code size and two parameters a_b and b_b. The COCOMO model involves other formulas but equation 8.1 is used to calculate substitution costs (cf. [358, 360]). The "Effort Applied" is measured in person-months and the value is meant to include the overhead for system design, specification drafting, reviewing, and management.

$$E = a_b * (\text{KLOC})^{b_b} \tag{8.1}$$

KLOC is the number of delivered lines of code for a project, being expressed in thousands. KLOC comprises not only programming language lines of code but also configuration for the build automation, deployment descriptors, or logging framework configurations.

For the formula parameters a_b and b_b, Boehm distinguishes three classes of software projects for COCOMO: organic projects, semi-detached projects, and embedded projects [357]. The organic projects represent small teams with least rigid requirements. Thus, it seems most appropriate to consider academic pilot implementations as COCOMO organic projects. The theoretical foundation of the formula is rather complex and well-documented in [357]. In conclusion, the statistical calibration of his model provides parameter configurations for each project type, summarized in table 8.6.

The *α-Flow* system implementation reuses several libraries, yet, the resulting source code and configuration files are completely handcrafted. For calculating the KLOC value, the analytical tool cloc[7] has been used. The cloc application is configured to count Drool™

7 http://cloc.sourceforge.net/

Class of software project	a_b	b_b
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

Table 8.6: The COCOMO parameter configurations for each COCOMO project type (adopted from Boehm [357])

rule files with the file extension "`*.drl`" as Java files using the command-line option "`force-lang=Java,drl`". Hence, it reports 102,582 lines in total on a source repository export: 10,472 blank lines, 39,257 comment lines, and 52,853 effective lines. Thus, the KLOC value for the *α-Flow* system is 52.85 thousand lines.

$$E_\alpha = 2.4 * (\text{KLOC}_\alpha)^{1.05} \ : \ \text{KLOC}_\alpha := 52.85 \tag{8.2}$$

$$E_\alpha = 154 \ \text{ in person months} \tag{8.3}$$

In conclusion, the applied development effort equals 12.9 person-years. The salary in the engineering sector for bachelor and master graduates, in Germany, is reported annually by the workers' union "IG Metall"[8]. It seems appropriate to use the values for bachelor graduates without working experience. The median is approximately €43,711 p.a., accordingly. Finally, the roughly estimated value of the *α-Flow* pilot is about €560,000 or rather the substitution of the *α-Flow* modules by means of an industrial project would cost so much money, approximately.

8.4 Performance Aspects

The *α-Flow* pilot emphasizes the introduction of new functionality and not performance optimization. However, to provide an objective analysis of the system, some performance aspects can be measured. An important influence on the overall *α-Flow* system performance has the storage subsystem. Consequently, an according performance assessment has been made. The *α-Flow* subsystem that provides an overlay network for peer transfer & remote synchronization is not assessed in terms of response time or throughput. Its performance characteristics rely on the underlying messaging infrastructure, which is currently SMTP and IMAP. However, all peer network nodes that belong to a distributed

8 The report is titled "Entgelte in der ITK-Branche 2012". In 2012, the analysis has been based on 28,200 personnel records from 132 companies. An excerpt is available online: `http://www.igmetall-itk.de/files/was_sind_sie_wert_cebit_2012__2_.pdf`

case file synchronization are assumed to be offline at time of message dispatching. Network performance characteristics like response time and throughput are of minor interest because the actual benchmark is still postal delivery of paper-based documents. In conclusion, other subsystems of the *α-Flow* engine have not been evaluated in respect of performance.

Performance Assessment of the α-Flow Storage Engine

The *α-Flow* storage engine is Hydra VCS subsystem. It was implemented and evaluated by Scott Hady in [343]. He assessed its performance in two areas: 1) data transfer between workspace and repository, and 2) data compression, i.e. hard disk footprint of the repository. Testing the (1) data transfer rate was accomplished by measuring the time[9] that the system needed to execute the VCS tasks of a) adding file system artefacts, b) committing the state of the file system artefacts, and c) returning the file system artefacts to a previously persisted state. Testing the (2) data compression was accomplished by measuring the resulting repository size after committing a large file system artefact set.

Hydra can be configured to store all repository file system artefacts uncompressed ("Hydra$^{\mathbf{U}}$") or compressed ("Hydra$^{\mathbf{C}}$"), cf. [343]. In the uncompressed mode, the Java NIO features are used for advanced input/output in Java (e.g., [364]). Since the Java 1.4 release they are provided by the java.nio package and sub-packages. For compression, the zip compression features provided by the java.util.zip package is additionally used to compress the persisted data to reduce the size of the repository. The performance result depends on the compression mode and both modes have been measured.

The stress test data consisted of 2,874 files comprising 983 MB of mixed binary and text documents, cf. [343]. The stress test execution was automatized with a script file. After initializing a new repository, the testing script adds the stress test file set and commits the workspace, for writing performance testing. Then it deletes the complete stress test file set from the working space (which is not timed) and reverts the workspace to the committed state, for reading performance testing. As a benchmark, Git[10] and SVN[11] have been measured using the same setting[12]. Both apply zlib-compression, Git applies

9 The tests were scripted using Linux' time command.
10 The Linux implementation of Git, by Torvalds et al. in version 1.7.6, was used.
11 The Linux implementation of SVN, by CollabNet in version 1.6.17, was used.
12 Particularly for SVN, the initialized repository must be local to exclude the influence of network transfer from the calculations. Git and Hydra use a local repository by default.

full-copy storage and applies zlib-compression to full files[13] whereas SVN uses delta-based storage and applies zlib-compression on the deltas[14].

Several files in the original file-set are identical binaries at different path positions. This is the reason why even an uncompressed Hydra repository requires less than 983 MB. Hydra applies, like Git, content-addressable identification and internally stores duplicate binaries only once, independently of their paths as positions in the file system.

All runs of the stress test were executed on the same system[15]. Git and SVN are implemented in C/C++ and Hydra is implemented in Java. Still, the Hydra executable size is 208 kB instead of Subversion's ∼12 MB or Git's ∼19 MB. The purpose of the stress test was to get an approximate performance estimation. All tests were executed five times and the slowest and fastest times were removed to reduce the effect of outliers caused by the Java garbage collection. A larger-scale evaluation with hundreds of runs on different hardware systems is not intended because performance optimization is not the objective of Hydra. Table 8.7 provides an overview of the stress test results.

Task	Git	SVN	Hydra^{U}	Hydra^{U} vs. Git	Hydra^{U} vs. SVN	Hydra^{C}	Hydra^{C} vs. Git	Hydra^{C} vs. SVN
Add	17.749s	7.465s	30.343s	x1.7	x4.1	49.669s	x2.8	x6.7
Commit	5.372s	56.881s	26.859s	x5.0	x0.5	26.857s	x5.0	x0.5
Retrieve	7.158s	37.063s	7.960s	x1.7	x0.3	11.812s	x1.7	x0.3
Summary	**Git**	**SVN**	**Hydra^{U}**			**Hydra^{C}**		
Σ time	30.279s	101.409s	65.162s	x2.2	x0.6	88.338s	x2.9	x0.9
Size	187.1MB	201.1MB	844.1MB	(x4.51)	(x4.19)	173.2MB	x0.92	x0.86

Table 8.7: Stress test results: Hydra VCS performance in comparison to Git and SVN

Git requires about 30 seconds to accomplish all three timed tasks and SVN requires about 101 seconds. Hydra requires about 65 seconds, in its mode without compression. Thus, Hydra is 2.2 times slower than Git but 40 percent faster than SVN. However, Hydra's repository size is more than four times bigger than both, in this mode. For this reason, the most comparative results are the ones for Hydra applying compression.

13 cf. `http://git-scm.com/book/en/Git-Internals-Git-Objects/`. The set-up uses Git's default *loose object format*. The *packfile* optimization does not apply because neither `git gc` command has been run manually nor has a push to a remote server been invoked, which would trigger the packfile-based repository reorganization automatically.

14 cf. `http://svnbook.red-bean.com/nightly/en/svn.reposadmin.maint.html`

15 The hardware environment was an Intel Quad 4 processor at 2.66GHz, 8GB random access memory, and a Western Digital 500GB Blue Edition hard drive. The operating system was an Ubuntu Linux distribution in version 10.4.

Consequently, Hydra is nearly three times slower than Git but it is still slightly faster than SVN and Hydra's repository size, hence, is slightly smaller than both ones of Git and SVN.

In conclusion, Hydra's performance characteristics are close enough to mature versioning systems. Hydra provides sufficient performance as the storage subsystem of the *α-Flow* engine and has a very low hard disk footprint[16] as a library.

8.5 System Limitations

The implementation of the *α-Flow* engine is quite substantial. Still, the pilot is a proof of concept and has several limitations. The following sections discuss possible improvements for the *α-Flow* system.

8.5.1 Automatic Merging of Process Artefacts

The overall chance for distributed parallel changes is systematically reduced by the dedicated ownership of content cards. The coordination cards Process Structure Artifact (PSA), Collaboration Resource Artifact (CRA), and Adornment Prototype Artifact (APA) are shared but based on the use case scenario the overall probability of any parallel conflicts is low. The logic timestamps of the synchronization protocol guarantees the conflict detection. For conflict resolution, the implemented default strategy merely identifies the latest globally valid version and will reset to it. The advantage of such strategy is that it can be applied for arbitrary content types and its reconciliation action can be automatized without further knowledge about the content semantics or its format. However, for artefacts that are managed by the *α-Flow* engine itself, like the shared work-list or even *α-Forms*, other reconciliation strategies are possible that can be automatized.

One strategy could be based on automated file difference calculation & merging. The coordination cards are XML-based lists. There are special file difference algorithms for XML, a survey is available by Rönnau [365] or by Peters [366]. Java libraries for XML difference calculation are, for example, DiffX[17], diffxml[18], or Eracaton's XOperator[19].

16 The embedded Hydra library without its command-line user interface and without its own OneJAR™ wrapper is only 154 kb of file size.

17 `http://www.topologi.com/diffx/`

18 `http://diffxml.sourceforge.net/`

19 `http://www.living-pages.de/de/projects/xop/`

Using an XML-merging strategy would allow to automatize parallel changes that affect different sections of the process structure, for example, if work items are inserted in the middle of the work-list.

A special case is the appending of new entries at the end of the XML list. For example, new *α-Card* entries at the end of the PSA work-list. Two parallel append-changes result in a conflicting difference calculation because the same section, i.e. the end of the file, would be affected. The favoured behaviour of the reconciliation would be to keep both appended entries. In this concurrent case, an additional merge criterion is necessary to ascertain the order of the merged-appends on all nodes. From the perspective of the users, the best criterion would be real-world time. However, information about physical clocks is unreliable and the logical clocks provide no ordering criterion for parallel changes. If we assume that the users accept any order of two parallel work items in the rare case of a parallel PSA append, the merge strategy could easily use the technical *α-Card* ID, which is a Universally Unique IDentifier (UUID), as unambiguous criterion to order parallel append-changes.

Currently, the conflicting versions remain in the VCS, basically for provenance purpose. A manual reconciliation strategy that displays conflicting versions to the user could be implemented. This strategy is compatible to the default reset strategy because it can be applied at any time. However, the necessary user interaction becomes quiet complex and it is not implemented at the moment.

8.5.2 Single-Shot Contributions

In general, it is possible for each actor to end his or her participation by simply removing his or her electronic post-box information from their *α-Doc* actor profile. The removal is propagated to all *α-Doc* replicas as usual and the content synchronization stops. Optionally, the actor can delete any copy he or she has of the *α-Doc*.

However, there is a special form of contribution that may be called *single-shot contribution*. An example is the pathologist during the episode of pre-therapeutic diagnostics for breast cancer (cf. sect. 5.2.1, p. 161). He is only involved in the case shortly. He just provides his or her histology report and is not concerned any further with the case. Strictly speaking, there is no necessity for the pathologist as a single-shot contributor to provide his electronic post-box information. The *α-Doc* already contains the electronic return addresses of all other participants and he could just contribute his or her report without formally joining the team.

The *α-Flow* implementation, at the moment, requires any contributors to enter their post-box information and it always carries out the join protocol. Only then, the single-

shot contributors can add their report and would immediately remove the post-box information before they delete the *α-Doc*. Laboratories are also notable single-shot contributors. Improved support for this use case could be provided, for example, by a graphical dialogue selection after the drag-and-drop of a content file on the *α-Doc*.

8.5.3 In-Memory Cache

Using the embedded editor sporadically shows noticeable delays. A first analysis has indicated that frequent store and retrieve interactions with the Hydra VCS is a primal cause. The performance evaluation of Hydra has shown that its implementation is not the bottle-net. Instead, the data access strategy is rather unoptimized at the moment.

The implementation guideline is "any data access has to be done via the *α-Kernel* subsystem, which uses the *α-VVS* subsystem as the only authority on the latest data state". The strict separation between editor, active property kernel, and storage engine prevents race conditions between user-triggered changes and network-triggered changes. Both sources of change, the *α-Editor* and the *α-OverNet*, are channelled through the *α-Kernel* subsystem that controls and synchronizes any data access. The implications of the current data access strategy can be considered in the context of any data update notification (of a payload or of a descriptor; from the user or from the network) because it triggers a refresh of the editor display. The implementation guideline implicates that each display refresh will re-load all descriptors via *α-Kernel*, each re-load queries the *α-VVS*, and each *α-VVS* query reads the data unit via Hydra VCS from hard disk. At the moment, the *α-VVS* subsystem uses synchronous delegation to the embedded Hydra module.

In conclusion, the *α-Kernel* or the *α-VVS* could implement read-caches to improve the user-experienced performance. For example, the *α-VVS* could implement a basic caching strategy for all `head` versions. Using Drools™ facilities for caching purposes would be another design option. The rule engine already provides a working memory and session concept. Currently, its working memory and session is cleaned after each *α-Kernel* invocation. The clearance is applied to prevent any side effects in the rule evaluation and each *α-Kernel* invocation loads only those data units that are necessary in an invocation context. However, this rather defensive application of the Drools™ rule engine could be changed by carefully reviewing the implemented rule conditions and actions for any potential rule execution conflicts. Thus, the rule engine could provide a natural in-memory cache.

8.5.4 Dynamic Rules Management

In chapter 5.5 on page 180, user-defined rules and actions have been discussed in the context of user-defined adornments. An automatic reaction on state changes of user-defined adornments would require configuring user-defined rules at run-time. An important consideration in selecting the Drools™ rule engine for implementing the active property kernel was its built-in ability to dynamically load rules at run-time. Thus, the *α-Flow* pilot is generally prepared for user-defined rules. However, a graphical rule editor for the end-users is currently not available.

Dynamic rule management implicates several open questions. For a start, users cannot be expected to provide rule actions in form of a programming language. For easing the specification of user-defined actions, an action library could be supported. There are several open research questions like "Which actions should be supported?", "How can they be parameterised by the user?", "Should user-defined rules be propagated/synchronized between the *α-Doc* clones of a case?", or "Which security issues are implicated by propagating user-defined rules?". In conclusion, dynamic rule management is not provided in the context of this thesis.

8.5.5 Secure User Authentication

The *α-Doc*s currently do not authenticate the user. Hence, access to each *α-Doc* must be restricted by the operating systems and the user desktop environments as it would be the case for passive electronic documents. Still, *α-Doc*s are hybrids of electronic documents and applications, by their very nature. It is an open question whether such active documents should be required to enforce secure user authentication on their own, as an application, for data privacy protection purposes. However, this question is beyond the scope of this thesis.

8.5.6 Content-Oriented Process Templates with embedded binary Content Templates

The process templates that can be created via the *α-Templates* subsystem are currently restricted to process-related information. The work-list is included in form of the filtered PSA export, participant and role templates are included in form of the filtered CRA export, an adornment prototype is included in form of the filtered APA export, and the work item descriptors are included in form of filtered *α-Card* descriptors. However, inclusion of binary templates for the payload files is currently not supported.

Most paper-based cases in healthcare are assumed to have content templates for referrals or various result reports. In order to support the inclusion of binary content templates into the content-oriented process templates of *α-Flow*, the graphical interface of the *α-Templates* exporter and importer must be extended. From a technical perspective, a process template file, which is exported, is a single data file in XML format. XML Schema supports binary content for XML elements in form of the xs:base64Binary content type (cf. [367]). The *α-Templates* importer could easily transform the embedded content template into an initial private-invalid version of the according *α-Card*.

8.5.7 Footprint Reduction

Some approximate storage calculation is required for the *α-Flow* approach because applying *α-Doc*s as distributed case files requires additional hard disk storage for each patient. An estimation can be made to exemplify the overhead for an institution. In 2009, the case numbers for breast cancer in Germany[20] was 71,874. The current number of breast cancer centres in Germany[21] is 255. Thus, each breast cancer centre has to handle about 282 cases each year. For each case, additional hard disk storage would be required of 14.73 MB, the binary file size of the *α-Flow* engine (cf. sect. 8.1). Thus, the storage overhead that is implicated by using *α-Doc*s would amount to approximately 4 GB per breast cancer centre each year.

In order to reduce the binary footprint, the size of the *α-Flow* engine could be reduced. There are several options. First, some library dependencies could be eliminated. For example, the decision to use the Spring framework has primarily been made because the framework mechanisms educate students about system modularization. The five Spring libraries (cf. sect. 8.1, tab. 8.2) amount to 1.27 MB. Ultimately, the Spring framework provides some convenience but is dispensable. Its 1.27 MB could be substituted with a few handcrafted factory classes.

The core-3.4.2[..].jar library of org.eclipse.jdt is another large library. It is an open-source Java parser of the Eclipse foundation that is used by Drools™ to parse its rule files. The Eclipse parser requires 4.16 MB. However, the parser could be substituted with the open-source JANINO[22] parser by Arno Unkrig and Matt Fowles. JANINO requires only about 600 kB. The substitution would require some customizations but guidance for several versions of Drools™ can be found online.

20 cf. GEKID-Atlas: http://www.gekid.de/
21 cf. http://www.onkozert.de/deutschland/karte.htm
22 http://docs.codehaus.org/display/JANINO/Home

Another large library is bouncycastle, which is used for cryptographic purposes. It requires 2.56 MB. A variant that has appeared just recently is Spongy Castle[23]. It is directly derived from the Bouncy Castle library but it is intended for the Android platform. Spongy Castle only requires about 1.21 MB. However, it is currently unclear whether Spongy Castle could replace Bouncy Castle in *α-Flow* because the limitations of Spongy Castle have not been evaluated.

Finally, all used third party libraries contain some classes that are not required by the *α-Flow* engine. It would be possible to use Java class file shrinkers or so-called obfuscators to detect and remove transitively unused classes, fields, methods, and attributes. Such shrinkers or obfuscators directly optimize the Java byte-code. A free tool for such purposes is ProGuard[24]. However, any usage of Java reflection by the own code or by any libraries requires to provide a so-called seed configuration that indicates code entry points to ProGuard. A first experimental application of ProGuard to the *α-Flow* engine has reduced the JAR file from 14.73 MB to about 3 MB. Yet, the experimental seed configuration is not perfect and some run-time functionalities result in exceptions because some classes are missing. The preparation of a full-fledged seed configuration can be quite intricate. Still, a carefully prepared seed configuration can reduce the binary footprint of the *α-Flow* engine significantly.

Programming active documents requires library economy. During the implementation of the *α-Flow* pilot, the library dependencies have been carefully monitored. Yet, the pilot is a proof of concept and does not emphasize binary footprint optimization. Still, the current executable size is small enough in relation to the supported functionality to be considered as a lightweight application. Thus, it fulfils the role model being implied by the active document metaphor.

8.6 Summary

This chapter has provided a technical evaluation of the *α-Flow* system. Initially, the executable artefacts have been described with their hard disk footprint. The included transitive external library dependencies have been accounted for.

The design and implementation of the *α-Flow* engine have been assessed via code metrics. The methodology of Lanza and Marinescu has been used. The resulting values of the measured code metrics have been interpreted. All metrics considered, the implementation is is well engineered.

23 http://rtyley.github.com/spongycastle/
24 http://proguard.sourceforge.net/

The system is of considerable size and a rough estimation of the effort needed to generate software of the size of the *α-Flow* implementation has been calculated. The methodology of the COCOMO model for cost estimation has been used. The substitution costs for the *α-Flow* system are about 12.9 person years of development and about €560,000.

In regard to performance evaluation, the Hydra VCS subsystem as the *α-Flow* storage engine has been assessed. As a benchmark, Git and SVN have been measured. The performance of Hydra is close to the performance of these systems. Finally, current system limitations of the *α-Flow* implementation have been discussed.

IV

Epilogue

9 | Evaluation of Capabilities

> “It is not the strongest of the species that survives, nor the most intelligent that survives. It is the one that is the most adaptable to change.”
>
> (Charles Darwin)

This chapter provides a conceptual evaluation of the *α-Flow* system. In the first part, a comparative analysis will oppose *α-Flow* to related approaches. The second part reviews the central themes of *α-Flow* and discusses the fitness for use. The third part provides a discussion of open issues and future work.

9.1 Comparative Analysis

For the selection of related approaches, there are three prime distributed Document-oriented Process Management (dDPM) qualities: to provide shared data access in inter-institutional environments, to provide workflow articulation and a model for ad hoc processes, and to fulfil the active document metaphor.

Each quality, individually, is targeted by a multitude of approaches. However, the underlying assumptions of each approach are significant. Thus, the combination of qualities is important and reduces the selection. In conclusion, for the comparative analysis approaches are considered that share at least two of the qualities. In the end, three projects have been selected that exhibit similar qualities as dDPM. First, the characteristics of the *α-Flow* system are summarized. Then the three related projects are outlined. Finally, a comparison is provided, which is based on the characteristics of content-oriented workflow models and active document approaches. In addition, the process model requirements of dDPM are used to compare the systems' capabilities for case handling.

9.1.1 α-Flow Characteristics

In comparison to the active document approaches, each *α-Doc* for dDPM has the following characteristics:

- a file-based active document like the original idea from Placeless documents
- portable like a TiddlyWiki, also containing an embedded editor for tasks lists and for basic forms
- like the Microsoft Active Documents containments it contains binary file types and supports editing delegation in order to instrument locally available applications
- (not as versatile as Ercatons)
- self-synchronization between its replicated copies is provided, similar to AXML

In comparison to the content-oriented workflow approaches, the *α-Flow* engine shares the concept of articulating workflow progress in form of content units. Each content unit has data attributes and a content life-cycle that is represented by status attributes. In addition, circulations and their dynamic characteristics have a strong influence on *α-Flow*. The lack of a content-oriented model for ad hoc processes resulted in the conceptual design of the process model of dDPM, which is implemented by *α-Flow*.

9.1.2 Component-Based Approaches to Distributed Circulation Folders

Electronic Circulation Folder (ECF) approaches like ProMInanD and POLITeam have been discussed in section 3.3.4. There are two more ECF projects both implementing an active document infrastructure. Both infrastructures are based on a distributed component middleware. They are named very similar, if not to say identically: XFolders and X-Folders. The XFolders project appeared in 2002 by Castellani and Pacull [368] from Xerox Research Centre Europe in France. The X-Folders project appeared in 2004 by Rossi [369, 370] from the University of Bologna in Italy. Both approaches are not directly related to each other. However, the concepts are very similar. For the sake of completeness, it must be mentioned that the Xerox "XFolders" approach is sometimes referred to as "X-Folders" likewise, which increases the confusion; in contrast, I will use the hyphen ('-') as the distinguishing trait between the two approaches.

Castellani and Pacull: XFolders

The XFolders project [368] implements electronic circulation folders. The XFolders are built upon an academic middleware for distributed components named CLF/Mekano

(cf. [371]) that also provides a scripting language. Forwarding an ECF to another user implies the ECF component migration from one component run-time container into another. Neither CLF/Mekano nor XFolders are publicly available.

An outline of the XFolders distributed architecture is provided in figure 9.1. However, a specification of the illustrated components or a detailed description on their functionality is missing. The connectional subsystems in the middle of the illustration, i.e. the XFolder Manager and the Document Migrator, are centrally installed and administered software components.

The XFolders project manages a list of tasks as its workflow schema and allows for adaptations at circulation time. In conclusion, the XFolders system by Castellani and Pacull can be considered as a content-oriented workflow system for ad hoc processes. The representation of the ECF as a distributed component being migrated between remote sites fulfils the active document metaphor. However, the XFolders project requires a uniform environment of component run-time containers at participating sites. Thus, it is suited for a distributed but still well defined institutional scope because the active document infrastructure is tightly coupled to its middleware environment.

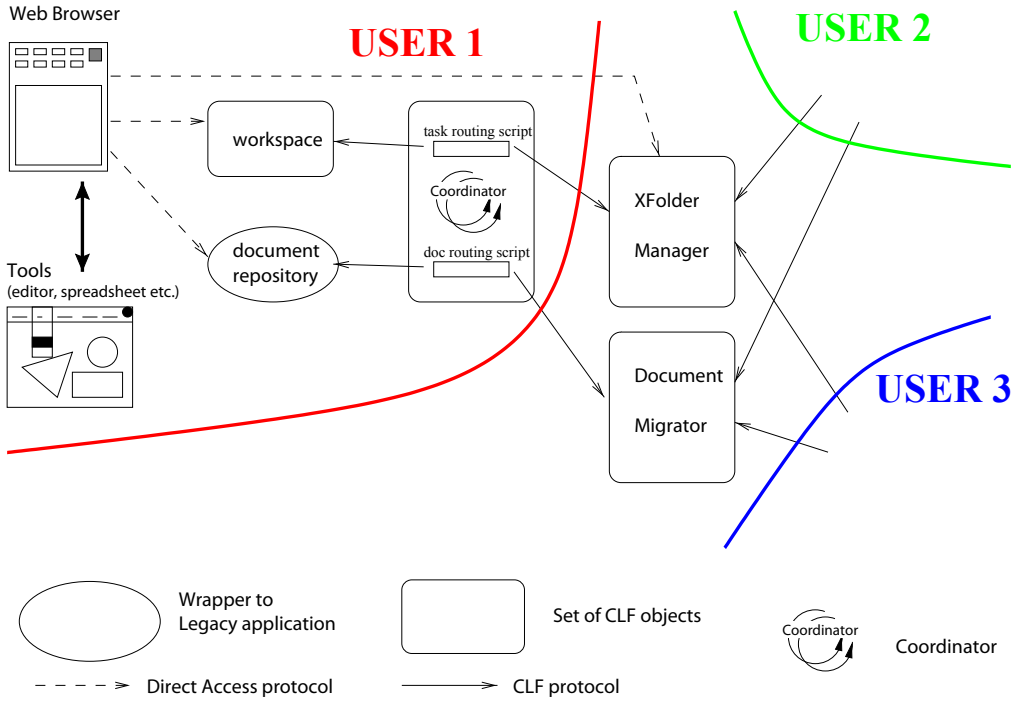

Figure 9.1: The Xerox XFolders architecture based on CLF/Mekano middleware for distributed components (adopted from Andreoli et al. [371])

Rossi: X-Folders

The X-Folders project [369, 370] also implements ECFs. It aims for decentralized circulation folders and uses Web-based Distributed Authoring and Versioning (WebDAV) technology. The author explicitly references the *active document* conception. The active property of an X-Folder is a programming script that is stored as a string in the WebDAV property of a WebDAV folder. Multiple X-Folders can be combined to a site-spanning application because WebDAV folders are intrinsically distributed. The X-Folders project describes in [369] a circulation that uses one folder to accept new forms, another folder with pending forms, and a last folder with accepted forms. These three folders are at different sites for three users with a different process role. The active property for each folder is handcrafted and only one folder of each type exists.

In principle, X-Folders scripts are similar to AppleScript Folder Actions (cf. sect. 3.5.1). The scripting language is a proprietary Extended Markup Language (XML) dialect. It eases the invocation of Simple Object Access Protocol (SOAP) web services as well as the dispatching of e-mail messages for user notifications. The guards and triggers are based on WebDAV copy and move operations, hard-coded for each X-Folder. The triggering and execution of WebDAV-embedded SOAP actions require a non-standard WebDAV server. The implementation is built upon a J2EE environment and servlet technology is used to expose X-Folders as WebDAV folders. The X-Folders project neither provides a formal representation of a workflow schema nor a dynamic adaptation. However, it implements a Content Management System (CMS) and the author casually mentions an application of X-Folders for implementing a conference management system. A key aspect of instrumenting WebDAV technology is to resolve firewall barriers in distributed scenarios.

The X-Folders project by Rossi emphasizes inter-institutional scenarios. This is a fundamental quality that the approach shares with dDPM and α-*Flow*, and which contrasts the approach to the XFolders project by Castellani and Pacull and all other active document approaches in section 3.5 (except AXML). However, the X-Folders system can hardly be considered a content-oriented workflow system because any formal process model is missing. In conclusion, the X-Folder project is primarily an active document approach that supports the implementation of basic circulations in inter-institutional environments.

9.1.3 Component-Based Active Document Approaches in Healthcare

One recent project implements an electronic patient file by an active document infrastructure. The use case is based on a clinical scenario. It is based on a distributed active component middleware environment.

Wilczek's Approach to Patient Files via Active Documents

The PhD thesis of Stephan Wilczek [372] describes an approach to implement electronic patient files as active documents. The objective of the Java-based patient file implementation is to provide an editor that resembles the paper-based Kardex™ [373] filing system. Accordingly, an internal Java object representation has been designed that follows the idea of the archetype concept of openEHR. The project does not use openEHR libraries or serialization formats, technologically. Yet, it implements a similarly flexible data structure, i.e. a tree-structure that allows recursive composition of archetype attribute groups. Each patient file is exactly one tree-structure. Several organizational and medical concepts are predefined as archetypes, like the patient master data or vitals like blood pressure, pulse, and temperature. In the end, the editor is capable to display the tree-structure, i.e. the patient file data, in emulation of the Kardex™ style.

The approach uses Open Services Gateway Initiative (OSGi) as Java component framework. Each participant has a host system installed that includes a customized OSGi run-time environment. The installation is eased by providing an installer based on Java Web Start[1]. Furthermore, OSGi allows the dynamic reloading of code modules at run-time. The system is not publicly available.

Each patient file is represented as one OSGi bundle instance. Wilczek explicitly argues the *active document* metaphor. The editor component and the patient file data form a single OSGi instance unit. Both can be serialized in XML: the tree-structure implementation supports XML serialization and the logic is serialized by encoding the underlying Java Archive (JAR) file of the OSGi component using Base64 [374]. Based on the combined XML serialization, the overall framework allows migrating such an active document via SOAP web services between remote run-time environments. For look-up purposes between peer hosts a service registry is implemented based on JavaSpaces [375] technology. However, the patient file in form of its active document is only exactly at one host at one time. Synchronization is not supported and the patient file may be lost if its current user host system becomes corrupted.

1 cf. `http://java.com/en/download/faq/java_webstart.xml`

The active document includes several additional active properties besides its editing capabilities. For example, a DICOM-to-JPG-converter is implemented and used for viewing purposes. In addition, a conversion of the electronic patient file into a printable representation is supported via Extensible Stylesheet Language Transformations (XSLT) [376] technology. An embedded XSLT-based converter allows for transforming the XML serialization into a Portable Document Format (PDF) format. Furthermore, the editor allows for user-specific configurations, like user-specific filter settings or the creation of user-specific views.

Wilczek's OSGi approach to active documents in healthcare is similar to XFolders by Castellani and Pacull, yet, it has no central system components and can be used offline. In conclusion, it provides circulations based on a uniform distributed component environment. Thus, Wilczek's OSGi approach is suited for a distributed but still well defined institutional scope.

An interesting aspect about the approach is its healthcare-related Java implementation of active properties, which fit well with *α-Flow*. The Eclipse Equinox[2] implementation of a basic OSGi component container requires only 1.3 MB as a JAR library. However, OSGi containers require special Java class-loaders. Thus, it is currently unclear whether an OSGi container could be embedded into the JAR bundle of an *α-Doc*. Still, the embedding of Wilczek's Kardex™-style XML serializations as cards into dDPM case files seems promising.

9.1.4 Evaluation: Characteristics of Content-Oriented Workflow Approaches and Active Document Approaches

The *α-Flow* system is evaluated in comparison to XFolders by Castellani and Pacull, X-Folders by Rossi, and Wilczek's OSGi approach to patient files. As the basis for comparison, the characteristics for content-oriented workflow approaches (sect. 3.4.5, fig. 3.14, p. 127) are used in combination with the characteristics for active document approaches (sect. 3.5.5, figure 3.22, p. 149). The resulting overview is illustrated by table 9.1.

The *α-Flow* system supports cards as simply typed content-oriented workflow elements. Yet, support for the adornment prototype and its inheritance relationship to card descriptors is a form of complex typed element. Consequently, elements are uniquely instantiated and the descriptors are instantiated by cloning the adornment prototype.

2 http://eclipse.org/equinox/

			approaches			
			α-Flow	XFolders [368]	X-Folders [370]	Wilczek's OSGi
content-or. workflow characteristics	element type system	simply typed	X	X	X	X
		complex typed	X			X
	element instantiation	unique elements	X	X	X	X
		class-based				X
		prototype-based	X			
	content progression	activity diagrams				
		rule-based	X			
		macro step graphs				
		circulation	X	X	X	X
	content accessibility scope	activity-wise				
		perpetual	X	X		
		occasional			X	X
	process assembly	flat	X	X	X	X
		layered				
active document characteristics	direct interaction	file operations	X		X	
		editing appl.	X			X
		HTML-editor		X		
		command-line	X			
	operability	file system				
		windowing system				
		web browser				
		component container		X	X	X
		cross-plattform lang.	X			
	portability	intrinsic	X			
		extrinsic			X	X
		unportable		X		
	remoteness	isolated				
		remote access		X	X	X
		distrib. scopes	X			
	run-time versatility	editing	X	X		X
		logic			X	

Table 9.1: Comparative analysis: characteristics of content-oriented workflow approaches and active document approaches

The content progression of cards as work items is rule-based and the current implementation provides a basic content progression model based on visibility and validity. The complete case file artefact is cloned or moved between sites, which can be considered as a circulation. It is possible to define a customized enumeration of content progression states in form of user-defined adornments. Content accessibility is provided perpetually, independent from activity execution and at any time. The process assembly is flat and based on a work-list. The dDPM concepts of sub-lists to support layered process assemblies have not been implemented at the moment.

The active document characteristics of the *α-Flow* implementation are essentially the ones required in chapter 4. Direct interactions are possible by file operation triggers like drag-and-drop, by an embedded editing application, or by a command-line interface. The operability is based on Java as a cross-platform programming language. A standard Java virtual machine installation is sufficient. The portability is intrinsically provided as the *α-Doc*s can be moved and copied like ordinary files. About remoteness, the *α-Doc*s of the same episode are self-synchronizing distributed scopes. The versatility is currently limited to ready-made editing. The enhancement of active properties in form of arbitrary logic at run-time is not possible. Still, the general *α-Flow* architecture is prepared for run-time system extensions because the underlying Drools™ engine supports dynamic reloading of rules and actions.

As content-oriented systems, the XFolders, X-Folders, and Wilczek's OSGi approach also support electronic documents as content-oriented workflow elements, i.e. simply typed elements. In addition, Wilczek provides flexible composite tree-structures, which are considered as complex typed elements, similarly to the "object-aware" approach. The three approaches all provide unique element instantiation. Yet, Wilczek's archetype-related data structures allow for class-based element instantiation as well. Concerning content progression, all three approaches provide circulations. The XFolders approach allows all participants perpetual access because it uses centralized system components. In contrast, the X-Folders approach allows only for occasional access because the content units are physically moving from one WebDAV site to another. In Wilczek's approach, the access is also only occasional because the patient file is successively migrated between participants but only available at one host at a time. Concerning process assembly, the XFolders system provides an explicit circulation model, which is still a flatly configured ECF routing without nested sub-processes. The X-Folders approach has no explicit circulation model, the remote folders are linked with WebDAV move or copy operations and nested sub-processes are not considered. Wilczek's approach has an explicit but very basic model for flat circulations.

As active document systems, the XFolders, X-Folders, and Wilczek's OSGi approach have different characteristics. For Castellani and Paculls's XFolders, the distributed composite

ECF is considered as a molecular active document. It provides direct interaction in form of an HTML editor. Its operability is limited to a CLF/Mekano environment. The content units of an ECF are stored locally in content repositories and an export of the ECF as an independent file unit is not considered, thus, the ECF is not portable. However, by means of CLF/Mekano middleware, remote interactions with an ECF seem possible beyond the scope of its run-time execution environment. In regard to run-time versatility, the editing of the ECF is supported but general-purpose logic cannot be added dynamically.

In the X-Folders approach by Rossi, each WebDAV folder can be considered as a molecular active document. The direct interaction is provided by file operations like WebDAV move or copy. The operability is limited by the custom X-Folders WebDAV middleware. It is possible to migrate an X-Folder from one run-time execution environment to another, extrinsically, because the active properties are stored as strings in the WebDAV properties, which can be exported. The application of WebDAV allows for remote access. The X-Folders approach does not provide ready-made editing but allows for general-purpose logic, for example invoking web services. The scripts within the WebDAV properties can be changed by versed end-users at run-time.

In Wilczek's approach, the patient file is the active document and the direct interaction is provided by an embedded editing application. The operability is limited to an OSGi component container environment. Portability is provided extrinsically by XML file extraction. The OSGi components provide means to migrate the active document to a remote container. Thus, remote access is provided virtually. Similar to *α-Flow* and XFolders, Wilczek's approach provides ready-made editing. Even if OSGi allows dynamically loading code modules at run-tim, like Drools™, the end-user has no means to add or change active properties at run-time.

9.1.5 Evaluation: Process Model Characteristics

This section evaluates the systems' capabilities of inter-institutional case handling. For this purpose, the dDPM process model requirements (sect. 5.7, tab. 5.2, p. 186) are used. The resulting overview is illustrated by table 9.2.

It has to be stressed that *α-Flow* has been implemented specifically to fulfil the dDPM requirements. The *α-Flow* engine provides an implementation of all core requirements. It also implements some extended requirements, like process roles, the adornment prototype, consensus scopes, or process templates. It does not yet implement extended requirements like sub-lists and content relationships, extended dDPM case termination concepts, or user-defined rules and actions. Ultimately, *α-Flow* is included in the list to illustrate specifically that it still lacks some process model requirements. The evaluation

in this section does not imply any superiority of *α-Flow* because each of the three related approaches has other capabilities beyond the scope of *α-Flow*. In conclusion, the purpose of the dDPM-related evaluation is to illustrate whether the other approaches allow articulating process coordination in an inter-institutional environment.

A constitutive classification about XFolders by Castellani and Pacull, X-Folders by Rossi, and Wilczek's OSGi approach is provided by assessing their universal inter-institutional and case-driven process characteristics, which are outlined as the two top sections in table 9.2, highlighted in light blue. The XFolders approach lacks basic considerations for inter-institutional scenarios but emphasizes content-oriented cooperation, which fits well with case-driven processes. The authors do not consider autonomous participants from different institutions with equal rights or integration of heterogeneous systems. The mindset of the X-Folders approach is somewhat the opposite. It emphasises on inter-institutional scenarios with autonomous participants and heterogeneous systems. Workflow support for scenarios that emulate paper-based working practice is the purpose of the X-Folders approach. However, it provides no intrinsic process model and the coordination must be handcrafted by programmers, similarly to CMS systems. Wilczek's approach is close to dDPM, naturally. However, it still emphasizes a clinical environment and integration must be solved a priori.

The core process requirements of dDPM for inter-institutional case-handling in healthcare are highlighted in light green in table 9.2. All three related approaches support distributed circulations. Thus, they fulfil the "partial results from various actors" criterion as well as "location-independent access". The card metaphor implies that the separation of content units is based on organizational accountability. Both the XFolders and X-Folders approaches fulfil the card metaphor. They allow for electronic document contributions to their respective ECF and manage an organizational context of each contribution. In Wilczek's approach, there is no organizational accountability for separate content units because the clinical team takes responsibility. An actor and role model is provided for user-specific display configurations. However, Wilczek's explanations about managing editing history particularly demonstrate that organizational accountability for content units is not considered in his pilot implementation. Still, in his initial case study, at the beginning of his thesis, he distinguishes data objects and relates them explicitly with distinct user roles. The archetype concept would easily provide a well-suited platform to separate content units and to relate them explicitly with organizational accountability. Overall, Wilczek's approach virtually fulfils the card metaphor.

The only approach that provides means to a priori plan contributions like *α-Flow* is the XFolders approach since it allows for planning circulations, which qualifies as the equivalent to a result-oriented work-list. It even manages state attributes for its ECF contributions like adornments for card progression states. The X-Folders approach does

	approaches			
	α-Flow	XFolders [368]	X-Folders [370]	Wilczek's OSGi
Site-Spanning Processes	X	X	X	X
Autonomous Participants	X	-	X	X
Heterogeneous IT-Systems	X	-	X	-
Decentralized Coordination	X	X	X	X
Paper-Based Working Practice	X	X	X	X
Dynamic Teams	X	X	-	X
Knowledge-Driven Processes	X	X	-	X
Team-Wide Information Demands	X	X	X	X
Result-Oriented Work-List	X	X	-	-
Partial Results from Various Actors	X	X	X	X
Card Metaphor	X	X	X	(X)
Adornments as Card Progression States	X	X	-	-
Domain User Roles	X	-	-	X
Parallel Work, by Default	X	-	X	-
Tacit Order by Non-Binding Prioritization	X	X	-	(X)
Location-Independent Access	X	X	X	X
Case Episodes as Process Scopes	X	X	-	-
User-Defined Indicators and Annotations	X	-	X	(X)
Separation of Visibility and Validity	X	-	-	-
Versioning and Process History	X	-	(X)	X
Open-Ended Case Termination	X	-	-	X
Cohesive-Content Relationships	-	-	-	-
Required-Content Dependencies	-	-	-	-
Episode Sub-Lists	-	-	-	-
Process Responsibility Roles	X	-	-	-
Adornment Prototype	X	-	-	-
Consensus Scopes	X	-	-	-
Process Templates	X	X	-	-
User-Defined Rules and Actions	-	-	-	-
Case Sealing and Closing	-	-	-	-
Sub-List Ceasing and Completion	-	-	-	-

Table 9.2: Comparative analysis: dDPM process model requirements for inter-institutional case handling

not provide a work-list directly. Wilczek's approach integrates a basic task list, like groupware application, but task items are narrative and are not related to content units. As mentioned above, Wilczek's approach supports domain roles, primarily for its display configurations. Both other approaches do not provide domain roles but only manage identities of individual actors. The XFolders approach seemingly allows only sequential circulations and parallel work is not supported. The X-Folders approach considers parallel work and provides WebDAV support for locking. Wilczek's approach allows access to each patient file only for one participant at any time. In regard to work order, the tacit order criterion means that no strict control flow patronizes the users, but support for prospective, non-binding prioritization is still provided. The XFolders approach implements sequential circulations that are strictly controlled. However, they can be changed dynamically. Thus, any remaining steps in a circulation configuration can be considered as a prioritized work-list that can be re-prioritized any time, which fulfils the tacit order criterion. In the X-Folders approach, traditional control flow constructs are used to strictly execute any programmed activities. Wilczek's approach does not restrict the contribution order but it also does not support a priori prioritization of content contributions. However, it is a criterion that has originally been derived from paper-based working practice and Wilczek's approach shares this mindset because it is generally discussed in his thesis. In conclusion, his approach is considered to fulfil the tacit order criterion.

The XFolders approach provides process scopes by allowing for explicit circulation schemes. The X-Folders approach and Wilczek's approach do not provide concepts to distinguish process scopes. Support for user-defined indicators is not provided by XFolders. The X-Folders approach could generally provide them by user-defined WebDAV properties. In Wilczek's approach there exists a so-called "general comment" archetype that can be instantiated anywhere in the patient file tree-structure. However, any such annotation is not typed like the ones by *α-Flow*. Separation of visibility and validity is not supported by any of the related approaches. Versioning is not provided by XFolders. The WebDAV facilities of the X-Folders approach support versioning implicitly but versioning is not discussed by the author at all. Wilczek's approach manages an editing history. The last core dDPM criterion is about open-ended case termination. In the end, the XFolders and X-Folders approaches assume that after a circulation is instantiated a specified number of steps is executed and then the circulation terminates. In contrast, dDPM's and Wilczek's conception of patient files or case files do not a priori assume any finishing time of the overall treatment.

The extended requirements of dDPM are listed in the bottom section of table 9.2 and are highlighted in light yellow. Support for the extended requirements is generally not

provided by the other approaches. However, the XFolders approach provides support for process templates in form of so-called "ECF Models".

In conclusion, for each of the core requirements of dDPM (all but the separation of visibility and validity) at least one of the other approaches can be identified that shares similar capabilities with *α-Flow*. Still, for inter-institutional scenarios that require emergent data exchange and ad hoc coordination, the *α-Flow* system provides a unique set of functionality.

9.2 Fitness for Use

The *α-Flow* system has multiple facets and the comparative analysis provides only a fragmentary evaluation of its purpose. The integral purpose of *α-Flow* is to improve IT assistance in distributed scenarios with autonomous participants that are willing to cooperate. Conventional tools only partially solve prime challenges. The *α-Flow* system comprises a multi-level architecture with the following aspects:

- to bring workflow articulation as close to the user as possible
- to choose the granularity for activities and processes according to paper-based working practice
- to support cooperation without the need to do data integration, still enabling deferred and demand-driven data integration in a pay-as-you-go style
- to synchronize multiple content repositories
- to provide flexibility by means of run-time adaptability

These aspects are discussed in the following in form of two constitutive challenges. The first challenge is to provide an infrastructure that allows the healthcare professionals to establish electronic information exchange without prior system integration. The subsequent challenge is to enable ad hoc processes by supporting workflow articulation.

The first challenge is about integration, which implies several degrees of data integration and functional integration. In inter-institutional scenarios, any tight system integration requires many efforts in time and money, even between only a few systems. As a dDPM hypothesis, two universal integration rules can be formulated (for IT-supported distributed cooperation between knowledge workers). The first rule is "anything that must be exchanged must be printable". This is crucial because anything that is printable can be captured as a digital document. The second rule is "the single concept that is universally shared between IT applications is the concept of electronic documents in form of data files". Notably, it would be deceptive to interpret the second rule such that solely dumping information as data files solves integration. The problem of syntactic and

semantic data integration will still require many efforts. Yet, using a content-agnostic data scheme as the basis of an exchange platform allows deferring data integration. Besides, specifications like Health Level 7 (HL7) Clinical Document Architecture (CDA) enable semantically rich content standards for document-oriented data integration in healthcare. The second rule only concerns the architectural style of inter-institutional data exchange. Instead of messaging or remote invocations, each content artefact must be representable in form of an electronic document independent of an application system.

In dDPM, lessons from case handling and from active documents are combined. Information can be shared by contributing it into a case file. The dDPM application that manages a case file (e.g., the *α-Flow* engine) synchronizes the contributed content units between remote sites. Thus, a dDPM implementation is considered an augmentation of the local healthcare systems. It provides the augmentation externally to the local systems. This can be understood as a target-oriented and demand-driven retrofitting of messaging and synchronization capabilities. In dDPM, the lesson from the second rule is also applied to the case files themselves. Paper-based case files are dossiers that are tangibly accessible on wooden desktops and stored in filing cabinet as single units of paperboard containers. Hence, electronic case files should provide a similar user experience. In dDPM, the case file is a composite of electronic documents but it must still be a single unit itself. Thus, the case file is turned into a molecular document. This property becomes convenient when some connection is required between distributed case files and local patient records in the site-specific Electronic Medical Record (EMR) systems: If case files are handled as molecular documents it is possible to integrate them into patient records basically as binary file attachments. Another dDPM assumption is "do not take it for granted that there is an application for handling a case file at another site". Thus, dDPM learns from active documents. The lesson is: "implement necessary software as a lightweight and installation-free application that can be bundled together with the molecular document". Thus, handing over the document coevally hands over the required application.

The subsequent challenge is to provide workflow articulation and an explicit process model. Traditional workflow management patronizes its users, which is necessary to some degree because workflow automation is an important purpose. However, cooperating knowledge-workers must not be patronized.

In dDPM, lessons from the diagnostic-therapeutic cycle and from agile methodologies are combined. In dDPM, users employ cards in order to indicate information demands. In addition, users can apply and adopt markers on the cards to articulate process-related status indication. The dDPM approach considers the content units themselves sufficient as triggers for workflow progression as they articulate work items in a shared work-list. Thus, dDPM provides a content-oriented process model that achieves flexible coordination support for human-oriented workflows.

As a form of final disclaimer, the dDPM conception has two additional assumptions: 1) "Doctors and nurses do not depend on information technology". This means that the dDPM mindset honours the fact that health service providers effectively use their experience or knowledge to cure patients without any software automations. 2) "Cooperation does not depend on information technology". This means that knowledge-workers use the telephone or paper-based postal deliveries to coordinate their work with others effectively in the absence of software support. However, doctors may benefit from the right type of information technology. The purpose of *α-Doc*s is to make distributed case handling more efficient, for example, by providing process planning, process history, participant management, and process template creation.

9.3 Discussion & Future Work

The breast cancer treatment was chosen as the basis of the dDPM requirement analysis because breast cancer involves various forms of cooperation and there is a basic documented consensus about the overall inter-institutional treatment process. However, the true power of the *α-Flow* system appears in genuinely ad hoc scenarios. In gynaecology, for example, there are cases that are initially diagnosed as "unbestimmte Unterleibsbeschwerden" in German, which translates into "undetermined abdomen discomfort". This is actually a non-diagnosis; still, the gynaecologist can guess several causes. For example, the cause could be an internal ailment that requires an internist or a urological ailment that requires an urologist. The gynaecologist could also refer the patient to a surgeon for a laparoscopy if he still suspects a gynaecological ailment. Probably the patient is multimorbid and several problems are the cause of her pain. The resulting treatment processes are somewhat elusive because they are inevitably inter-institutional and the unfolding depends, above the ordinary, on the intuition of the participating doctors. Cases of high initial medical uncertainty that have many potential participants are genuine beneficiaries of *α-Flow*.

The *α-Flow* pilot has not been employed in a healthcare field test, at present. A small-scale field test at our department has demonstrated its functionality. For productive purposes, several system limitations from section 8.5 should still be solved. As the current implementation is equivalent to about 154 person months (cf. sect. 8.3), from my personal experience I would estimate that another 30 to 60 person months would be required to polish the *α-Flow* implementation for practical application. Some functionalities were out of scope of this thesis but would be required for productive purposes. Field tests are required to gain user feedback for improvements of the user interface. The responsiveness must be improved, implementing an in-memory caching strategy for the storage and

messaging facilities. Nevertheless, the prototype is designed for using *α-Flow* in a real world context.

The purpose of the *α-Flow* system is to provide an interim solution, in the medium-term, until an ideal implementation of dDPM can substitute *α-Flow*, in the long-term (as has been outlined in section 5.8). The ideal dDPM implementation requires pre-integration, for example, a full-fledged and site-spanning Electronical Health Record (EHR) environment, which is not in sight. In contrast, the prime objective of *α-Flow* has been to bridge the current gap between primary and secondary care. Still, there are regional scenarios in which time and money are available to achieve an a priori well-integrated system environment. In fact, this is progressively pursued by breast cancer centres, and state subsidy is provided for many regional cancer-related system integration projects (e.g., [377]). In the long term, it will become imperative to standardize document-oriented interactions between tightly-integrated regional federations and loosely-coupled wide-area case handling systems. Specifications from Integrating the Healthcare Enterprise (IHE) like the Patient Care Coordination (PCC) specification and its Exchange of Personal Health Record Content (XPHR) chapter are possible candidates. It is an open challenge whether XPHR could technologically be supported by an *α-Doc*.

Similar challenges would arise if the *α-Flow* systems should solely be used in clinical settings for cross-departmental cooperation. Considerations about teamwork between doctors and nurses, as they are documented in Wilczek's thesis [372], are currently not addressed by *α-Flow*. For example, user-specific display configurations are not supported. It seems also promising to design an interaction with a clinical system that manages Problem-Oriented Medical Records (POMRs) (cf. sect. 1.2.7) and to use SOAP-formatted progress notes as dDPM cards (cf. sect. 2.2.11). Moreover, the scope disclaimer in section 1.4.2 has mentioned Master Patient Index (MPI) systems. Yet, a particular clinical context would provide good cause to implement basic support for interactions between an *α-Doc* and some local MPI. In order to maintain the active document idea, all functions require adherence to an economy of libraries keeping the sub-modules as small as possible.

Mobile devices are of ascending importance in healthcare (e.g., [378]). Active documents match well with the limited resources of a mobile device. Furthermore, an *α-Doc* is an autonomous application and does not require continuous network availability. Peer deployments of *α-Doc*s to an arbitrary number of mobile devices are possible. In conclusion, the *α-Flow* approach has considerable potential in the context of mobile devices.

9.4 Summary

This chapter has provided a conceptual evaluation of the *α-Flow* system. Initially, approaches have been identified that share at least two qualities with *α-Flow*. Three approaches have been selected and together with *α-Flow* itself have been compared according to the characteristics for content-oriented workflow approaches and active document approaches. In addition, the capabilities for distributed case handling of the four systems have been evaluated using the process model requirements of dDPM as a reference. Subsequently, a debate of the fitness for use has reviewed the central themes of *α-Flow*. The last section has discussed some open issues of dDPM and *α-Flow*. Finally, some recommendations on future research have been outlined.

10 | Conclusion

> “Do not fear to be eccentric in opinion, for every opinion now accepted was once eccentric.”
>
> (Bertrand Russell)

Healthcare processes are intrinsically ad hoc and inter-institutional medical treatments implicate decentralized workflows. This thesis analysed the healthcare requirements for distributed and ad hoc process support with initially unknown sets of actors and institutions. As a use case that covers the inherent complexity of medical cooperation, breast-cancer treatment was analysed. The paper-based working practice provides a frame of reference for inter-institutional data integration as well as modelling human-oriented cooperative processes. The distributed Document-oriented Process Management (dDPM) approach adopts electronic documents as the primary means of information exchange. Even if the dDPM approach is primarily discussed in the context of healthcare, all presented methods are applicable to other domains with case characteristics like law (legal case management), sales (lead acquisition), insurance (claim handling), or science (e.g., research funding processes).

The workflow-related challenge of dDPM was to adapt documents to carry the workflow context in addition to the domain content. The paper-based working practice is used as a reference for case handling. In the end, dDPM provides a work-list concept that combines process articulation derived from agile methodologies like Kanban and Scrum with concepts for content progression derived from content-oriented workflow approaches. In the dDPM work-list, cards represent tasks and cards are used to prospectively plan cooperative processes. Workflow progress is equivalent to the successive contribution of content documents like reports.

The challenge related to integration was to bridge the gap between institutions of the primary and secondary care and to foster the availability of patient information. The fundamental boundary condition in the dDPM scenario is the strict autonomy of the participating sites in a healthcare network. For the purpose of data integration, the document-oriented method was discussed. The essential argument for document-oriented

integration over interface-orientated integration lies in its capacity to support deferred system design. A survey on the theory of semantic integration and software evolution was conducted. In conclusion, deferred system design is necessary for healthcare information systems due to their evolutionary character. The document-oriented integration supports semantically heterogeneous and even informal content types for distributed, large-scale scenarios. Further data integration, in order to provide advanced functionality, can be applied in a demand-driven manner.

Concerning the workflow-related information exchange, the rationale behind separating content, decision support, and coordination work was explained. The separation of concerns is driven by the objective to provide process support in spite of missing system integration. Decision support, in particular, requires a profound semantic understanding of rich medical content. In order to support heterogeneous systems, we need to decouple collaboration functionality from the existing applications. The collaboration is considered as a feature of the autonomous case file artefact and not of the local application systems. The case handling engine primarily administers coordination artefacts. Still, it allows access, viewing, and editing of the embedded content documents through common editors in the local information system without corrupting the process semantics of the distributed case engine.

The challenge related to operative embedding was to bring the tool for distributed case handling as close to the end-user as possible. Thus, dDPM applies the idea of active documents. One of their most appreciated properties is simplicity because they can be handled like common desktop files and no pre-installed system components are required to interact with an active document. By their active properties, the documents themselves become the driving force of cooperation.

The implementation of the dDPM process conception is provided by the *α-Flow* engine. The active documents that implement dDPM case files are so-called *α-Doc*s. The *α-Flow* engine provides a reference architecture for Java-based active documents. The architecture includes facilities for direct interaction and reactivity, similar to agents. As a result, each *α-Doc* carries autonomous coordination logic and process status information. The *α-Flow* artefacts were formalized and the various facets of implementation challenges and design choices were outlined. The *α-Doc*s minimize the initial work for establishing an information exchange between different process participants. For the end user, they embed a functional fusion of group-based instant messaging, shared work-list composing, and version control.

For process status indications, a run-time adaptive attribute model is provided for work items in form of *α-Adornment*s. Furthermore, the *α-Doc* embeds a rule engine that guards status changes and executes actions as the kernel of the active document. In addition, the boundary conditions impose unique requirements for versioning and

distributed synchronisation, which were analysed and explained. As as result, an offline-capable synchronization protocol based on Adaptive Vector Clocks (AVCs) has been designed and implemented, as well as an embedded multi-module and validity-aware Version Control System (VCS). Interactions with an *α-Doc* are possible without immediate network availability, even though a trusted network infrastructure is needed for the background synchronization. The work-list editor provides users with the means to gain shared knowledge about each other's documents and on that account about their activities. In the end, *α-Doc*s provide workflow benefits like process planning, process history, participant management, and process template creation.

In conclusion, the dDPM process model and the *α-Flow* implementation provide process support for inter-institutional physician teams and allow for patient-centred document management. The *α-Doc* is a case file that is bundled with an embedded engine for distributed case handling according to the active document metaphor. The *α-Doc* artefacts enable a seamless enhancement of existing healthcare information systems with inter-institutional collaboration facilities.

Appendices

A | Explanatory Notes

A.1 Security Concepts

The specifications for the "Elektronische Gesundheitskarte" (eGK), [72, p. 54] and [73, sect. 6.4.4.1], describe several security concepts for protection targets as a matter of legal boundary conditions. Table A.1 provides a translation of the eGK description.

Protection Target	Definition
Confidentiality ("Vertraulichkeit")	Guarantee that data and information are accessible exclusively to authorized users in the permissible way.
Integrity ("Integrität")	Soundness of information and data. Data cannot be modified undetectably, e.g. during the transmission of electronic communication.
Authenticity ("Authentizität")	Refers to the truthfulness of origins to ensure the data is genuine. Within electronic communication it is required to validate that both parties involved are who they claim they are.
Non-Repudiation ("Nichtabstreitbarkeit")	Guarantee that the dispatch and receipt of data and information cannot be denied. In other words it is about the liability and the proof of a transaction.
Availability ("Verfügbarkeit")	Guarantee that information and services, if these are called upon by the users, can be used at any time and in the intended speed.

Table A.1: Protection targets defined by the eGK specifications (adapted from [72])

A.2 CDA Example

```
<?xml version="1.0" encoding="UTF-8"?>
<ClinicalDocument xmlns="urn:hl7-org:v3"
 xmlns:voc="urn:hl7-org:v3/voc"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 templateId="2.16.84.11383.3.27.1776">
  <!-- CDA Header -->
  <id extension="c266" root="2.16.84.11383.3.933"/>
  <code code="11488-4" codeSystem="2.16.84.11383.6.1"
    displayName="Consultation␣note"/>
  <title>Health Clinic Consultation Note</title>
  <effectiveTime value="20000407"/>
  <setId extension="BB35" root="2.16.84.11383.3.933"/>
  <versionNumber value="2"/>
  ...
  <recordTarget><patientRole>
    <id extension="12345" root="2.16.84.11383.3.933"/>
    <patientPatient>
      <name><given>Henry</given>
            <family>Levin</family></name>
      <administrativeGenderCode code="M" codeSystem="2.16.84.11383.5.1"/
         >
      <birthTime value="19320924"/>
    </patientPatient>
  </patientRole></recordTarget>
<!-- CDA Body -->
  <component>
    <bodyChoice><StructuredBody>
      <component><section>
       <code code="10164-2" codeSystem="2.16.84.11383.6.1"
          codeSystemName="LOINC"/>
       <title>History of Present Illness</title>
       <text>Henry Levin, the 7th is a 67 year old male
       referred for further asthma management. ...</text>
      </section></component>
    </StructuredBody></bodyChoice>
  </component>
</ClinicalDocument>
```

Listing A.1: HL7 v3 CDA example

A.3 Workflow Management Coalition: Terminology

The Workflow Management Coalition (WfMC) provides a reference document "Terminology & Glossary" [143]. It contains several figures that are reproduced in figure A.1, A.2, A.3, A.4, and A.5 for illustrative purposes.

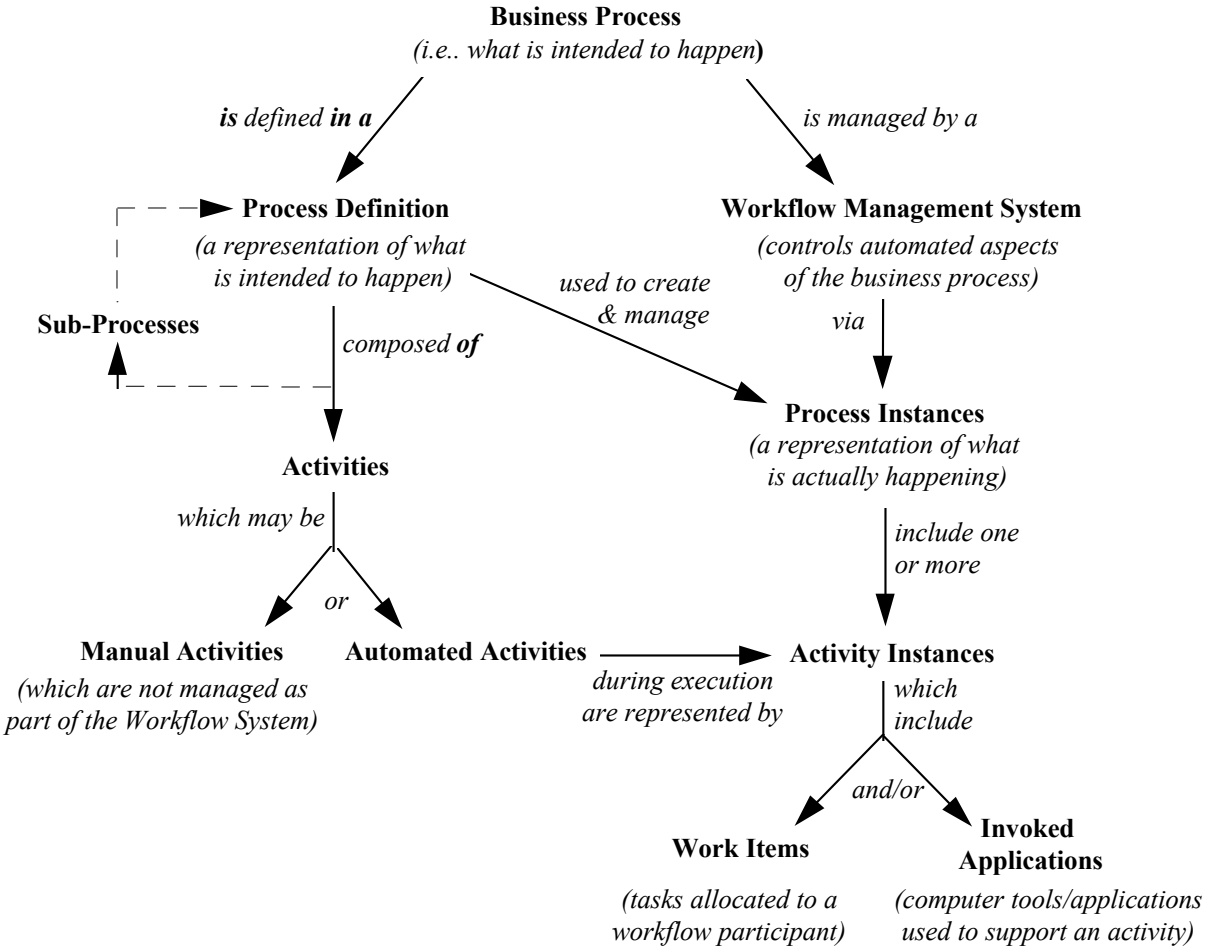

Figure A.1: WfMC Terminology & Glossary: Relationships between basic terminology (cf. [143, p. 7])

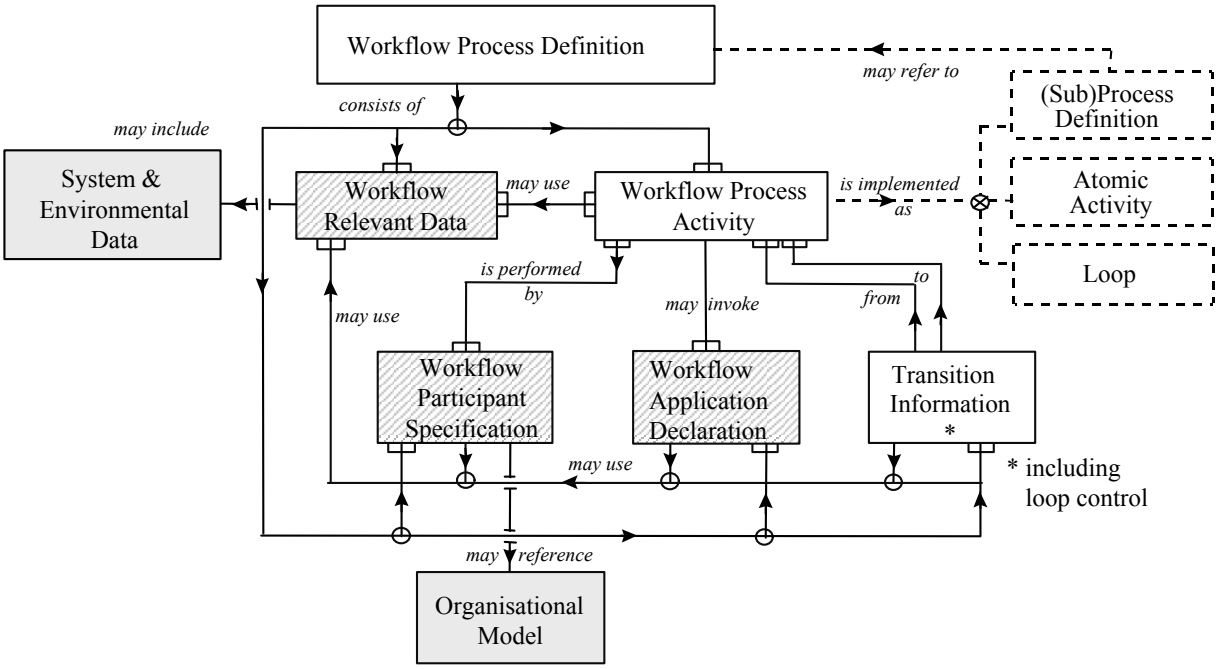

Figure A.2: WfMC Terminology & Glossary: WfMC process definition meta-model (cf. [143, p. 12])

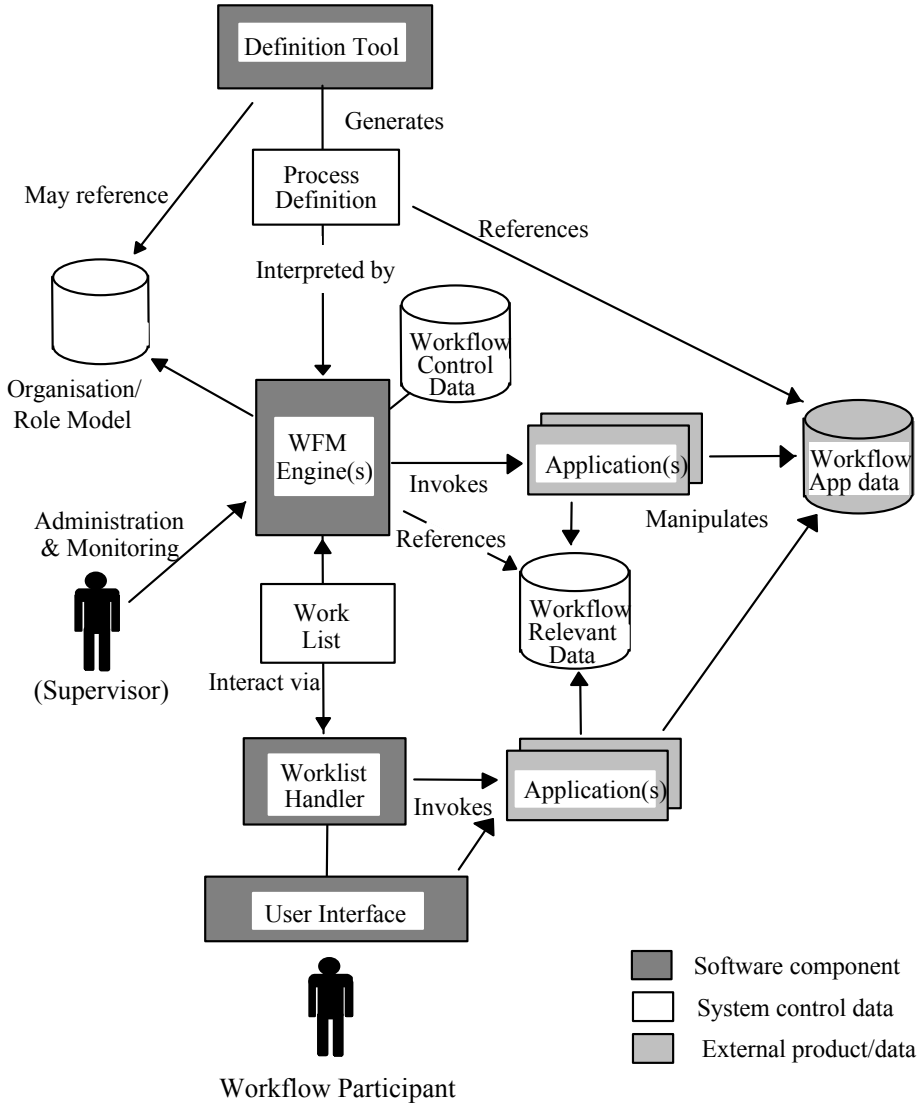

Figure A.3: WfMC Terminology & Glossary: Generic workflow product structure (cf. [143, p. 39])

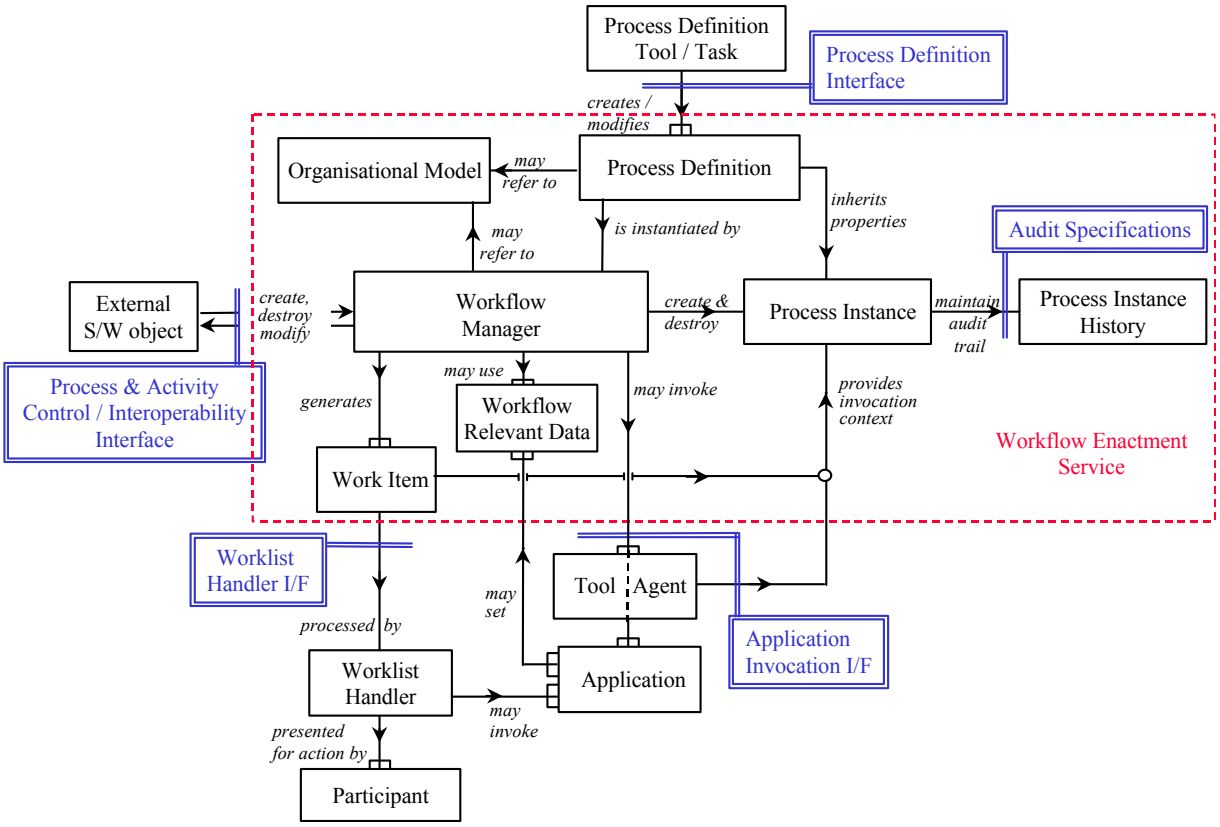

Figure A.4: WfMC Terminology & Glossary: WfMS components & interfaces (cf. [143, p. 40])

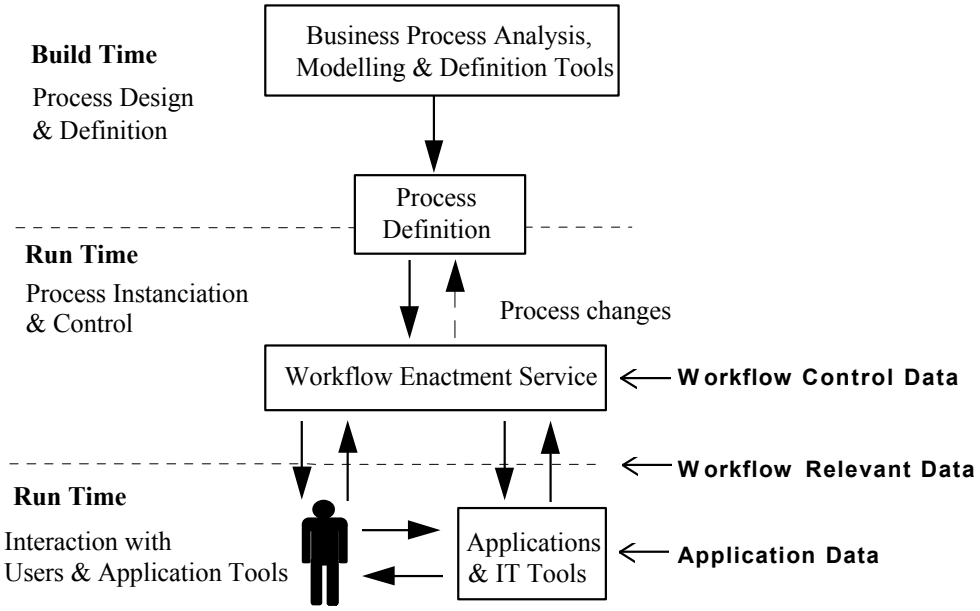

Figure A.5: WfMC Terminology & Glossary: Types of data in WfMSs (cf. [143, p. 44])

A.4 Active Document Technology

This section provides additional illustrative material about some active document approaches: a screen-shot of the original Placeless document publication as well as the XReference specification of the Ercaton approach.

A.4.1 Placeless Documents

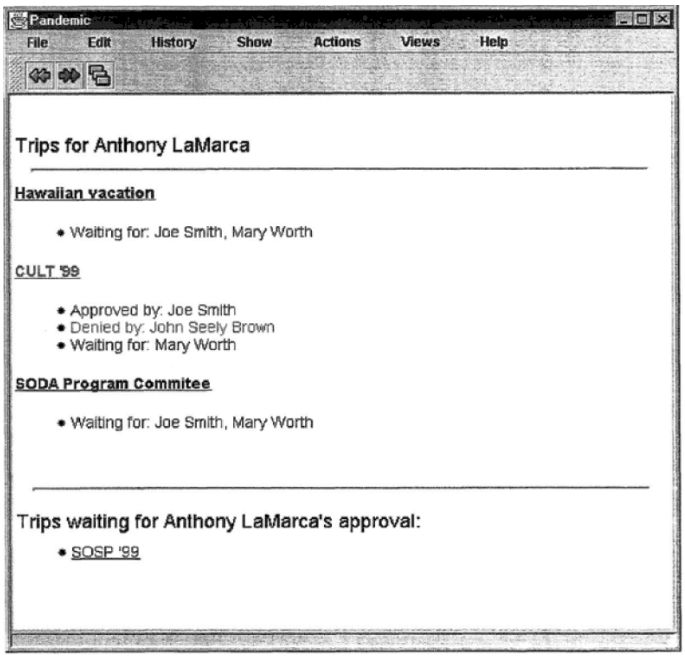

Figure A.6: LaMarca's example for content-oriented workflows based on Placeless documents: the trip status document (adopted from [273]). *Remark: None of LaMarca's screenshots provides insight on the system design or implementation.*

A.4.2 Ercatons: XReference Identification Scheme

```
<XReference> := <global_XReference> | <local_XReference>

<global_XReference> := [:<engine>:]<id>[,<version>]{<qualifier>[(<
   parameterlist>)]}[:[<target>][:[<erclet>]]]

<id> := << any legal ercaton-id >>
<version> := v | << any legal version string >>
<target> := null | << any target name >>
<erclet> := null | << any erclet name >>
<qualifier> := ! [(<qtype>)][<qexpression>]
<qtype> := action | target | trigger | xp | xq | scope
    -- a missing qtype defaults to qtype 'action'.
    -- xp is an abbreviation for XPath; xq for XQuery
<qexpression> := main | << any expression name, e.g. an action name >>
    -- a missing qexpression for qtype 'action' defaults to qname 'main'
<parameterlist> := <parametername>=<parametervalue> [,<parameterlist>]
<parametername> := << quoted or unquoted name of a parameter >>
<parametervalue> := [()<XReference>[)] | \$[()<variable>[)] | << quoted
   or unquoted value >>
<engine> := local | << hostname >>[,<engine_id>]
    -- a missing engine defaults to engine 'local' which is local to the
        current ercaton.
<engine_id> := << port offset with respect to default ports >>
    -- e.g., port 80; a missing engine_id defaults to '0'

<local_XReference> := [<path>[,<version>]]{<qualifier>[(<parameterlist>)
   ]}[:[<target>][:[<erclet>]]]
<path> := self | this | << id path relative to local ercaton >>
    -- e.g., '../mysiblingercaton'; a missing path defaults to 'self'.
```

Listing A.2: XReference identification scheme for Ercatons (cf. [301])

B | α-Flow

B.1 User Interface

The following section provides an early design of the work-list dashboard and several additional screenshots of the *α-Flow* prototype.

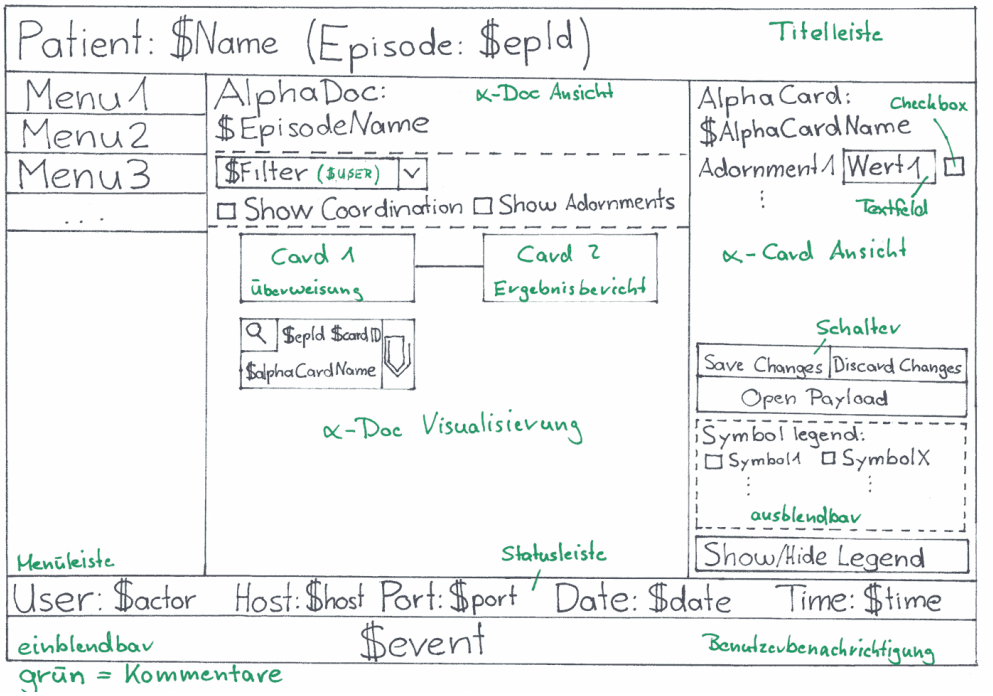

Figure B.1: Early GUI sketch for the *α-Editor* (adopted from [312])

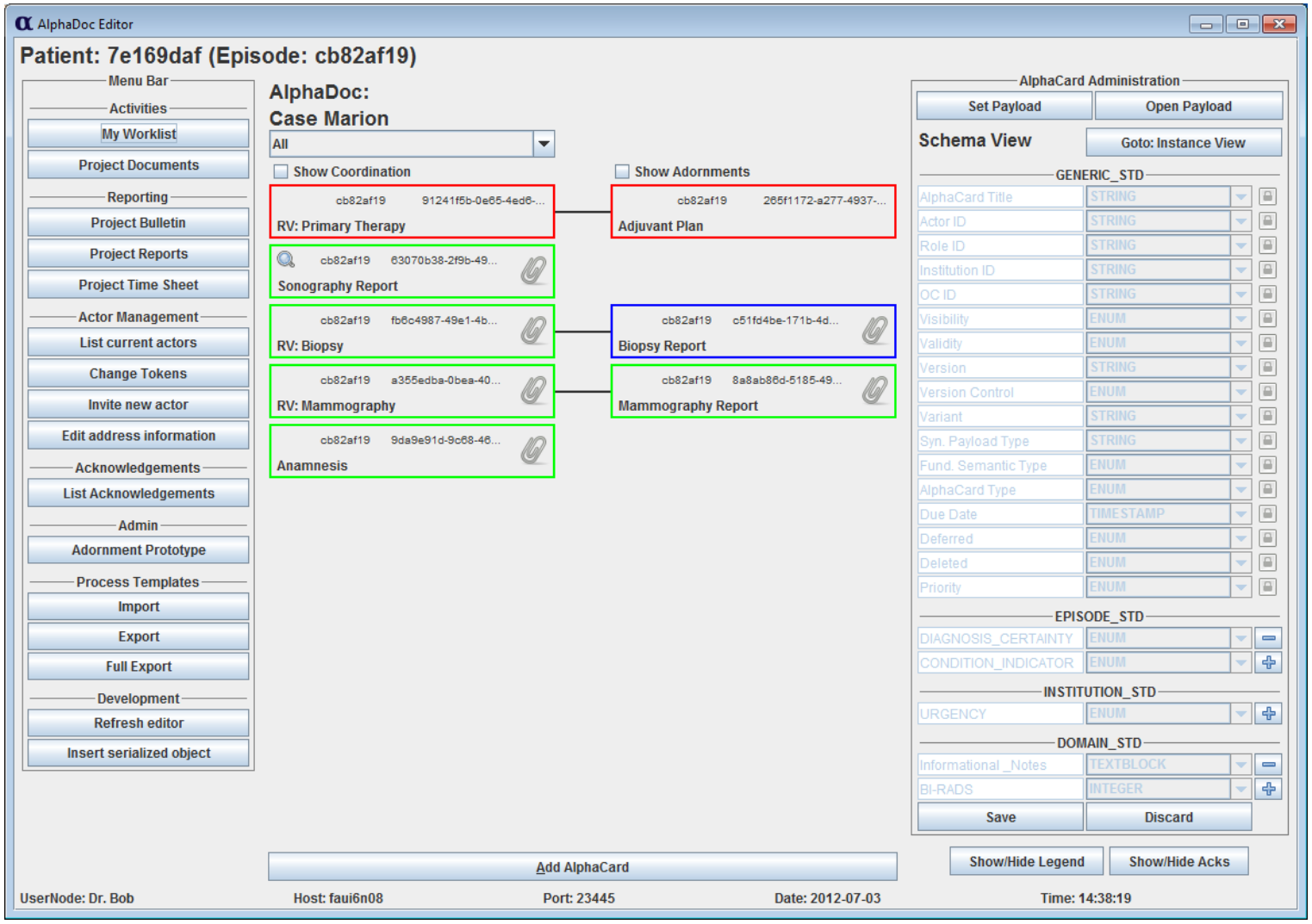

Figure B.2: The right half of the dashboard is switched into the adornment schema selection mode that is available for each *α-Card* descriptor, provided by the *α-Adaptive* extension

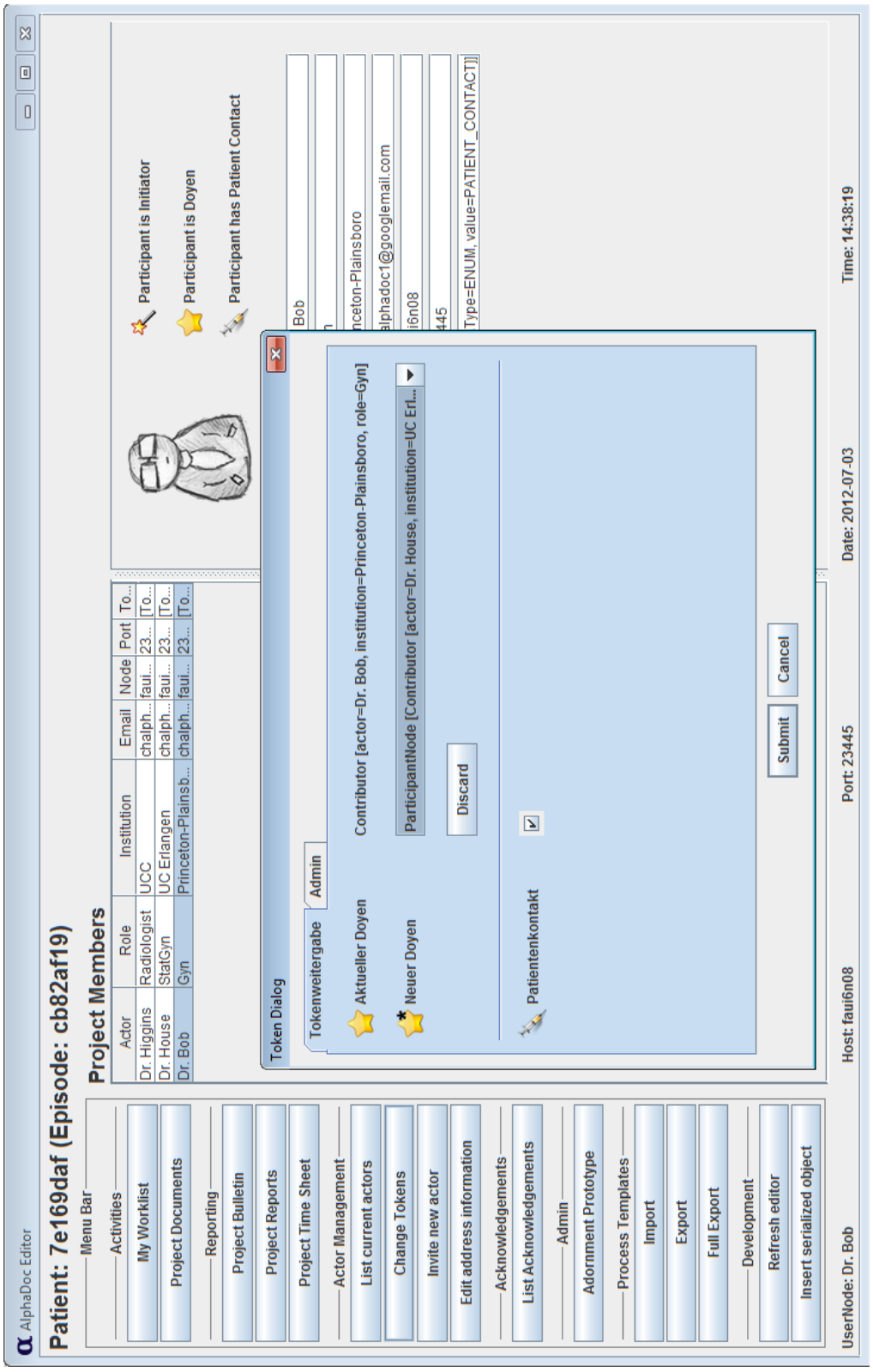

Figure B.3: A screenshot of the CRA editor provided by the *α-Doyen* subsystem

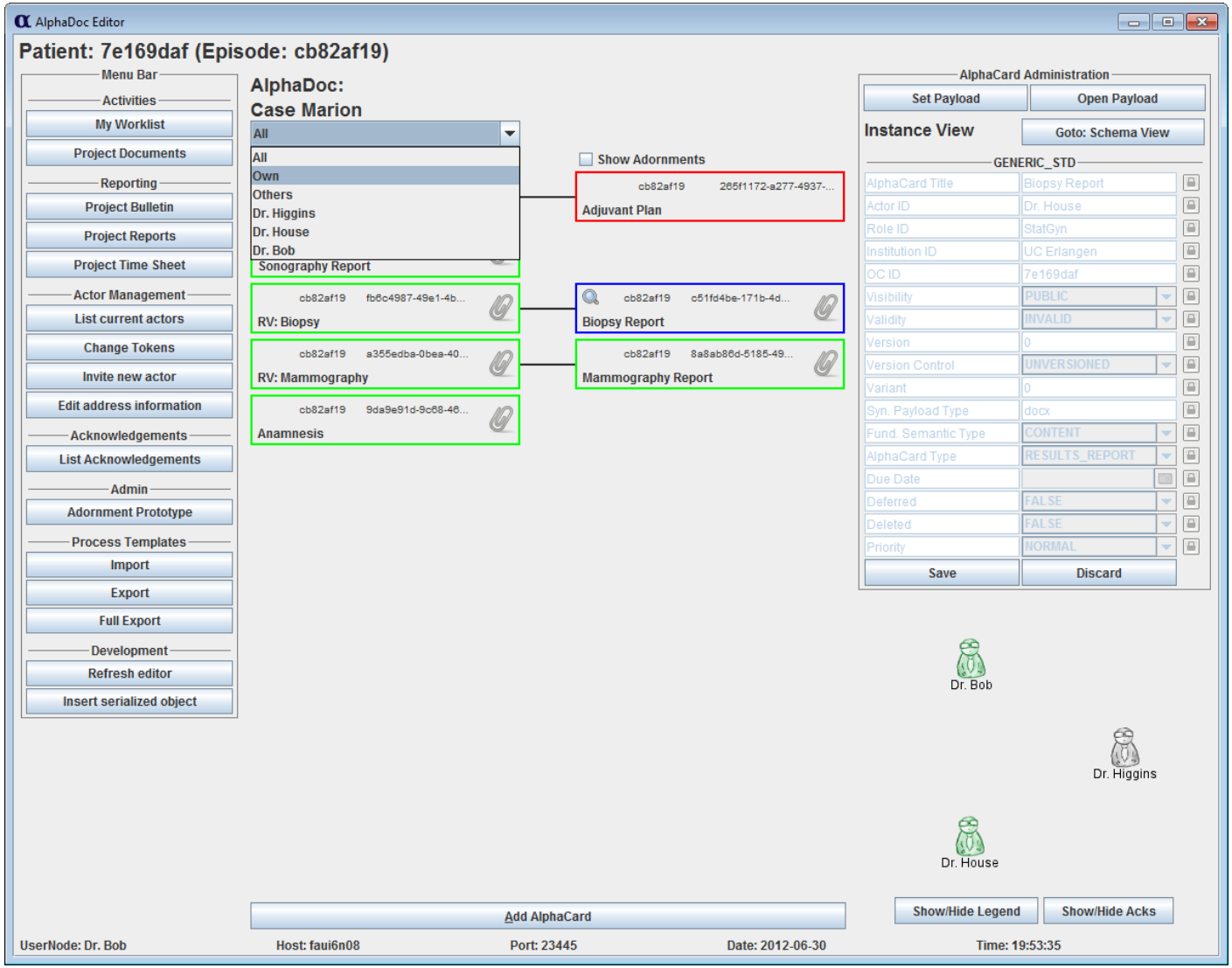

Figure B.4: A screenshot of the work-list dashboard with the *α-Doyen* extension for receipt acknowledgement indications

B.2 α-Adaptive

The reference model of *α-Adaptive* for run-time adaptive status attributes is outlined as UML class diagram in figure B.5. An AlphaCardDescriptor contains an AlphaCardID and a list of adaptive adornments that can be managed by methods to read, update, create, or delete adornments. An adaptive adornment (AdpAdornment) consists of a name, a value, an instance flag, an AdornmentDataType, and a ConsensusScope.

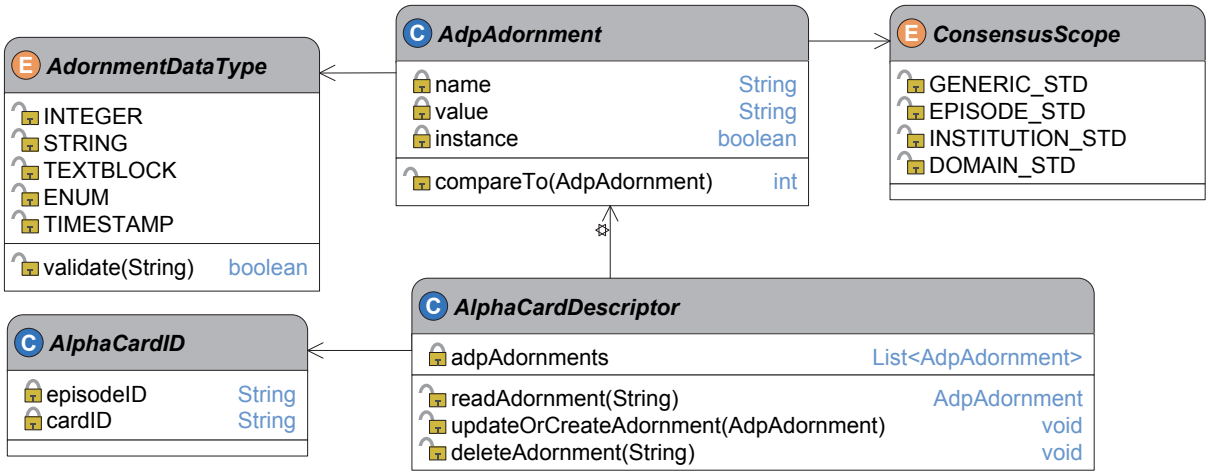

Figure B.5: The *α-Adaptive* classes of the adaptive adornment implementation (adopted from [330])

B.3 Hydra Version Control

Figure B.6 illustrates the update/pull–modify/push–propose/pull–accept/push workflow for distributed Version Control Systems (dVCSs). The Hydra VCS object model has been derived from the Git object model as a conceptual extension. The Hydra classes that implement the multi-headed and validity-aware versioning are outlined in figure B.7.

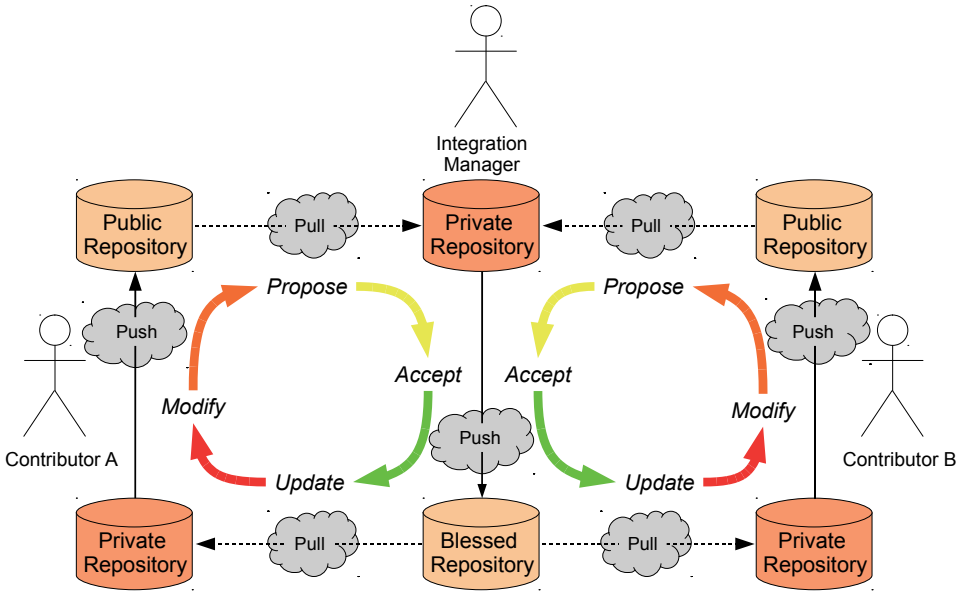

Figure B.6: The dVCS repository integration by a blessed repository (adopted from [343])

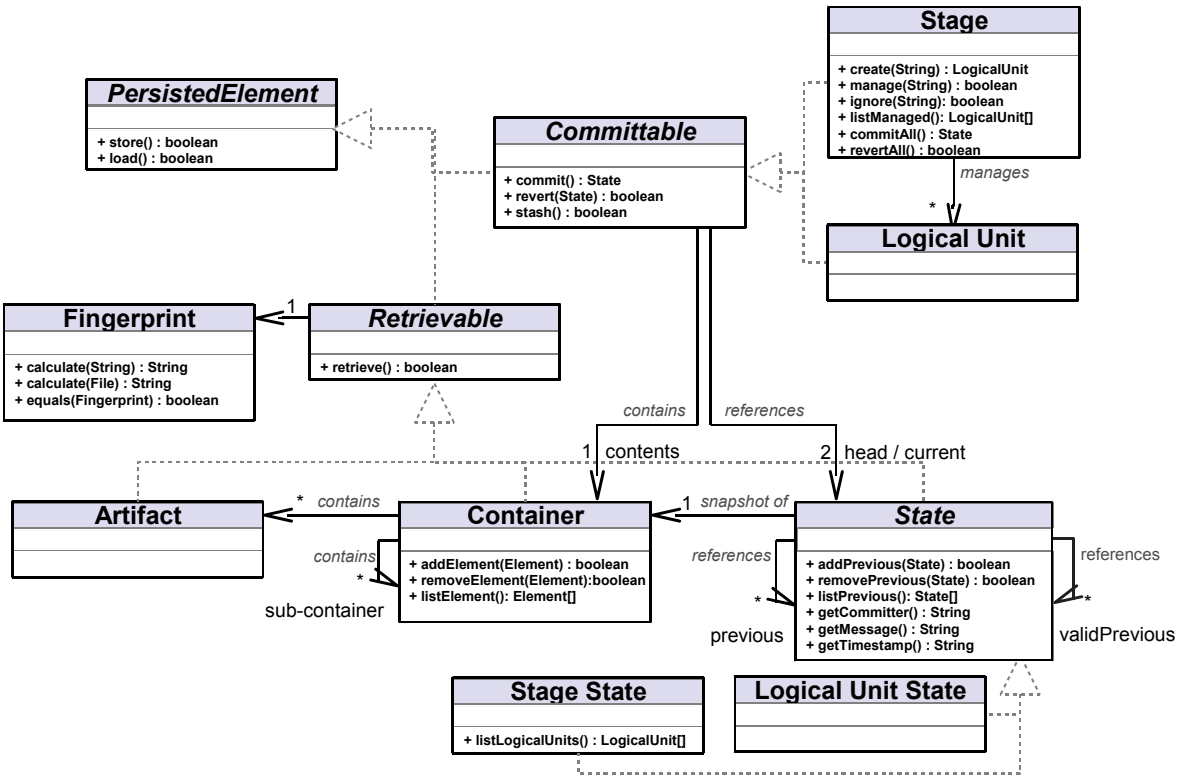

Figure B.7: The Hydra VCS classes that implement the multi-headed and validity-aware versioning (adapted from [343])

B.4 The α-Flow Source Code

The master development repository has been a private Subversion (SVN) repository at our institute. At the end of the project, the system will be made publicly available. The master repository will be migrated from the private Subversion to a public repository. The homepageof the α-*Flow* project will detail any information:
`http://www6.cs.fau.de/research/projects/promed/alphaFlow/`

Bibliography

[1] Linda T. Kohn, Janet M. Corrigan, Molla S. Donaldson, et al. *To Err Is Human: Building a Safer Health System*. IOM Report. Washington, D.C., USA: National Academy Press, Nov. 1999. ISBN: 0309068371.

[2] Linda T. Kohn, Janet M. Corrigan, Molla S. Donaldson, et al. *Crossing the quality chasm: A new health system for the 21st century*. IOM Report. Washington, D.C., USA: National Academy Press, Mar. 2001. ISBN: 0309072808.

[3] L. L. Leape, D. W. Bates, D. J. Cullen, J. Cooper, H. J. Demonaco, T. Gallivan, R. Hallisey, J. Ives, N. Laird, and G. Laffel. 'Systems analysis of adverse drug events. ADE Prevention Study Group'. In: *Journal of the American Medical Association (JAMA)* 274.1 (1995), pp. 35–43.

[4] J. J. Nobel and G. K. Norman. 'Emerging information management technologies and the future of disease management'. In: *Disease Management* 6.4 (2003), pp. 219–231.

[5] A. Maaz, M. H. J. Winter, and A. Kuhlmey. 'Der Wandel des Krankheitspanoramas und die Bedeutung chronischer Erkrankungen (Epidemiologie, Kosten)'. In: *Fehlzeiten-Report 2006*. Ed. by Bernhard Badura, Henner Schellschmidt, and Christian Vetter. Springer, pp. 5–23. ISBN: 3540343679.

[6] L. G. Glynn, J. M. Valderas, P. Healy, E. Burke, J. Newell, P. Gillespie, and A. W. Murphy. 'The prevalence of multimorbidity in primary care and its effect on health care utilization and cost'. In: *Family Practice* (2011).

[7] J. A. Barondess. 'Specialization and the physician workforce'. In: *Journal of the American Medical Association (JAMA)* 284.10 (2000), p. 1299.

[8] K. E. Thorpe, L. L. Ogden, and K. Galactionova. 'Chronic conditions account for rise in Medicare spending from 1987 to 2006'. In: *Health Affairs* 29.4 (2010), p. 718.

[9] P. R. Dexter, D. K. Miller, D. O. Clark, M. Weiner, L. E. Harris, L. Livin, I. Myers, D. Shaw, L. A. Blue, J. Kunzer, et al. 'Preparing for an Aging Population and Improving Chronic Disease Management'. In: *Proc of the AMIA Annual Symposium*. Vol. 2010. American Medical Informatics Association. 2010, p. 162.

[10] R. Lenz. 'Information Systems in Healthcare – State and Steps towards Sustainability'. In: *IMIA Yearbook 2009 – Yearbook of Medical Informatics as a supplement of Methods of Information in Medicine* (2009). Ed. by Antoine Geissbuhler and Casimir Kulikowski, pp. 63–70.

[11] Bernd Sippel. 'Evaluation und Integration von Standards zum Datenaustausch im medizinischen Umfeld'. Diplomarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, Mar. 2005.

[12] Philip Kotler, Kevin Lane Keller, and Friedhelm Bliemel. *Marketing-Management: Strategien für wertschaffendes Handeln*. Pearson Studium, 2007. ISBN: 9783827372291.

[13] M. A. Musen and J. H. van Bemmel. *Handbook of medical informatics*. Bohn Stafleu Van Loghum, 1997. ISBN: 3540633510.

[14] Richard Lenz and Manfred Reichert. 'IT Support for Healthcare Processes'. In: *Proc of the 3rd Int'l Conf on Business Process Management (BPM'05)*. Ed. by Wil M. P. van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera. Vol. 3649. Nancy, France, Sept. 2005, pp. 354–363.

[15] D. L. Sackett, W. Rosenberg, JA Gray, R. B. Haynes, and W. S. Richardson. 'Evidence based medicine: what it is and what it isn't'. In: *British Medical Journal (BMJ)* 312.7023 (1996), pp. 71–72.

[16] Ikujiro Nonaka. 'The Knowledge-Creating Company'. In: *Harvard Business Review* 69 (1991), pp. 96–104.

[17] Ikujiro Nonaka and Hirotaka Takeuchi. *The knowledge-creating company: How Japanese companies create the dynamics of innovation*. Oxford University Press, 1995. ISBN: 0195092694.

[18] Mario Stefanelli. 'The socio-organizational age of artificial intelligence in medicine'. In: *Artificial Intelligence in Medicine* 23.1 (2001), pp. 25–47.

[19] K. Lorig, D. Sobel, D. Laurent, and V. Gonzalez. *Living a Healthy Life With Chronic Conditions: Self-management of Heart Disease, Arthritis, Diabetes, Asthma, Bronchitis, Emphysema & Others*. Bull Publishing Company, 2000.

[20] M. H. Tattersall, P. N. Butow, J. E. Brown, and J. F. Thompson. 'Improving doctors' letters'. In: *Medical Journal of Australia* 177.9 (2002), pp. 516–522.

[21] Josie Samers. *Report on Integrated Care in Advanced Cancer Project*. Tech. rep. Inner and Eastern Melbourne BreastCare Consortium, Mar. 2004.

[22] R. Lenz, M. Beyer, C. Meiler, S. Jablonski, and K. A. Kuhn. 'Informationsintegration in Gesundheitsversorgungsnetzen'. In: *Informatik-Spektrum* 28.2 (2005), pp. 105–119.

[23] M. Oschem, V. Mahler, and H.-U. Prokosch. 'Objectifying user critique: a means of continuous quality assurance for physician discharge letter composition'. In: *Methods Inf Med* 50 (2011), pp. 23–35.

[24] J. G. Anderson. 'Clearing the way for physicians' use of clinical information systems'. In: *Communications of the ACM* 40.8 (1997), pp. 83–90.

[25] A. Oxman, GRADE Working Group, et al. 'Grading quality of evidence and strength of recommendations'. In: *British Medical Journal (BMJ)* 328.19 (2004), pp. 1490–4.

[26] J. C. Wyatt. 'Management of explicit and tacit knowledge'. In: *Journal of the Royal Society of Medicine (JRSM)* 94.1 (2001), pp. 6–9.

[27] R. Lenz and M. Reichert. 'IT support for healthcare processes – premises, challenges, perspectives'. In: *Data & Knowledge Engineering (DKE)* 61.1 (2006), pp. 39–58.

[28] I. Kopp, A. Encke, and W. Lorenz. 'Leitlinien als Instrument der Qualitätssicherung in der Medizin – Das Leitlinienprogramm der Arbeitsgemeinschaft Wissenschaftlicher Medizinischer Fachgesellschaften (AWMF)'. In: *Bundesgesundheitsblatt - Gesundheitsforschung - Gesundheitsschutz* 45 (3 2002), pp. 223–233.

[29] R. Kreienberg, Deutsche Krebsgesellschaft, et al. *Interdisziplinäre S3-Leitlinie für die Diagnostik, Therapie und Nachsorge des Mammakarzinoms*. Zuckschwerdt, 2008. ISBN: 978-3886039340.

[30] Sascha Müller. 'Modellbasierte IT-Unterstützung von wissensintensiven Prozessen – Dargestellt am Beispiel medizinischer Forschungsprozesse'. PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, 2007.

[31] R. Lenz, R. Blaser, M. Beyer, O. Heger, C. Biber, M. Bäumlein, and M. Schnabel. 'IT support for clinical pathways – Lessons Learned'. In: *Int'l Journal of Medical Informatics* 76.3 (2007), pp. 397–402.

[32] R. H. Fletcher, M. S. O'Malley, S. W. Fletcher, J. A. Earp, and J. P. Alexander. 'Measuring the continuity and coordination of medical care in a system involving multiple providers'. In: *Med Care* 22 (May 1984), pp. 403–411.

[33] M. L. Müller, F. Ückert, T. Bürkle, and H.-U. Prokosch. 'Cross-institutional data exchange using the clinical document architecture (CDA)'. In: *Int'l Journal of Medical Informatics* 74.2-4 (2005), pp. 245–256.

[34] C. van Walraven, M. Taljaard, C. M. Bell, E. Etchells, K. B. Zarnke, I. G. Stiell, and A. J. Forster. 'Information exchange among physicians caring for the same patient in the community'. In: *Canadian Medical Association Journal* 179.10 (2008), p. 1013.

[35] T. Ganslandt, U. Kunzmann, K. Diesch, P. Palffy, and H.-U. Prokosch. 'Semantic Challenges in Database Federation: Lessons Learned'. In: *Proc of the 19th Int'l Congress of the European Federation for Medical Informatics (MIE2005)*. Sept. 2005, pp. 551–556.

[36] M. H. Williams, G. Venters, and D. Marwick. 'Developing a regional healthcare information network'. In: *IEEE Transactions on Information Technology in Biomedicine* 5.2 (2001), pp. 177–180.

[37] J. Powell and I. Buchan. 'Electronic Health Records Should Support Clinical Research'. In: *Journal of Medical Internet Research* 7.1 (2005).

[38] C. P. Waegemann. 'The five levels of electronic health records.' In: *MD computing: computers in medical practice* 13.3 (1996), p. 199.

[39] Thomas M. Lehmann and Erdmuthe Meyer zu Bexten. *Handbuch der medizinischen Informatik*. Hanser, 2002. ISBN: 3446215891.

[40] Alexander Dobrev, Karl A. Stroetmann, Veli N. Stroetmann, Jörg Artmann, Tom Jones, and Reinhard Hammerschmidt. *The conceptual framework of interoperable electronic health record and ePrescribing systems*. `http://www.ehr-impact.eu/downloads/documents/EHRI_D1_2_Conceptual_framework_v1_0.pdf`. Apr. 2008.

[41] P. C. Tang, J. S. Ash, D. W. Bates, J. M. Overhage, and D. Z. Sands. 'Personal Health Records: Definitions, Benefits, and Strategies for Overcoming Barriers to Adoption'. In: *Journal of the American Medical Informatics Association (JAMIA)* 13.2 (2006), p. 121.

[42] M. Franklin, A. Halevy, and D. Maier. 'From databases to dataspaces: a new abstraction for information management'. In: *ACM Sigmod Record* 34.4 (2005), pp. 27–33.

[43] David Nueschler. *David's Model: A guide for content modeling*. `http://wiki.apache.org/jackrabbit/DavidsModel/`. 2007.

[44] Stefano Mazzocchi. *Data First vs. Structure First*. `http://www.betaversion.org/~stefano/linotype/news/93/`. 2005.

[45] S. K. Rothschild and S. Lapidos. 'Virtual Integrated Practice: Integrating Teams and Technology to Manage Chronic Disease in Primary Care'. In: *Journal of Medical Systems* 27.1 (2003), pp. 85–93.

[46] S. Brucker, U. Krainick, M. Bamberg, B. Aydeniz, U. Wagner, A. DuBois, C. Claussen, R. Kreienberg, and D. Wallwiener. 'Rationale, funktionelles Konzept, Definition und Zertifizierung'. In: *Der Gynäkologe* 10 (2003), p. 862.

[47] H.-U. Prokosch, T. Ganslandt, R. C. Dumitru, and F. Ückert. 'Telemedicine and Collaborative Health Information Systems'. In: *it – Information Technology* 48.1 (2006), pp. 12–23.

[48] Theresia Theurl and Stefanie Franz. *Benchmark integrierte Versorgung im Gesundheitswesen: Erste empirische Ergebnisse*. Tech. rep. Institut für Genossenschaftswesen der Westfälischen Wilhelms-Universität Münster, 2007.

[49] A. Lützenkirchen. 'Interdisziplinäre Kooperation und Vernetzung im Gesundheitswesen—eine aktuelle Bestandsaufnahme'. In: *Gruppendynamik und Organisationsberatung* 36.3 (2005), pp. 311–324.

[50] R. Kreienberg, D. Alt, W. Jonat, V. Möbus, and T. Volm. *Management des Mammakarzinoms*. 3rd. Springer, 2006. ISBN: 3540317473.

[51] Alison Clarke. *Open Text Case Management Framework*. `http://www.slideshare.net/opentextcorp/open-text-ecm-suite-case-management/`. Oct. 2010.

[52] David Roe. *Enterprise CMS Usage Scenario: Case Management Frameworks*. `http://www.cmswire.com/cms/enterprise-cms/enterprise-cms-usage-scenario-case-management-frameworks-006071.php`. Nov. 2009.

[53] Hajo A. Reijers, JHM Rigter, and Wil M. P. van der Aalst. 'The Case Handling Case'. In: *Int'l Journal of Cooperative Information Systems* 12.3 (2003), pp. 365–392.

[54] K. D. Swenson et al. *Mastering the Unpredictable. How Adaptive Case Management Will Revolutionize The Way That Knowledge Workers Get Things Done*. Meghan-Kiffer, 2010. ISBN: 0929652126.

[55] Christoph P. Neumann and Richard Lenz. 'alpha-Flow: A Document-based Approach to Inter-Institutional Process Support in Healthcare'. In: *Proc of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'09) in conjunction with the 7th Int'l Conf on Business Process Management (BPM'09)*. Ulm, DE, Sept. 2009.

[56] C. Altenhofen, H. Hofmann, T. Kieninger, and M. Stanišić-Petrović. 'Results of a Survey about the Use of Tools in the Area of Document Management'. In: *Reading and Learning* (2004), pp. 328–354.

[57] J. A. Mülle, K. Böhm, N. Röper, and T. Sünder. 'Building conference proceedings requires adaptable workflow and content management'. In: *Proc of the 32nd Int'l Conf on Very Large Data Bases (VLDB'06)*. Sept. 2006, pp. 1129–1139.

[58] Santhosh Kumaran, Rong Liu, and Frederick Wu. 'On the Duality of Information-Centric and Activity-Centric Models of Business Processes'. In: *Advanced Information Systems Engineering*. Ed. by Zohra Bellahsène and Michel Léonard. Vol. 5074. Lecture Notes in Computer Science. Springer, 2008, pp. 32–47.

[59] Lawrence L. Weed. 'Medical records that guide and teach'. In: *New England Journal of Medicine* 278.12 (1968), pp. 652–657.

[60] J. F. Fries. 'Time-oriented patient records and a computer databank'. In: *Journal of the American Medical Association (JAMA)* 222.12 (1972), pp. 1536–1542.

[61] A. L. Rector, W. A. Nowlan, S. Kay, et al. 'Foundations for an electronic medical record'. In: *Methods Inf Med* 30.3 (1991), pp. 179–86.

[62] J. R. Schultz, S. V. Cantrill, and K. G. Morgan. 'An initial operational problem oriented medical record system: for storage, manipulation and retrieval of medical data'. In: *Proc of the AFIPS Spring Joint Jomputer Conference*. May 1971, pp. 239–264.

[63] M. Bainbridge, P. Salmon, A. Rappaport, G. Hayes, J. Williams, and S. Teasdale. 'The Problem Oriented Medical Record – just a little more structure to help the world go round?' In: *Proc of the 1996 Annual Conf of the Primary Health Care Specialist Group (PHCSG) of the British Computer Society*. Sept. 1996.

[64] I. Carey, D. Cook, S. De Wilde, S. Bremner, N. Richards, S. Caine, D. Strachan, and S. Hilton. 'Implications of the problem orientated medical record (POMR) for research using electronic GP databases: a comparison of the Doctors Independent Network Database (DIN) and the General Practice Research Database (GPRD)'. In: *BMC Family Practice* 4.1 (2003), p. 14.

[65] Richard Lenz. 'Information Management in Distributed Healthcare Networks'. In: *Data Management in a Connected World (Essays Dedicated to Hartmut Wedekind on the Occasion of His 70th Birthday)*. Ed. by Theo Härder and Wolfgang Lehner. Vol. 3551. Lecture Notes in Computer Science. Springer, 2005, pp. 315–334.

[66] Lawrence L. Weed. 'The problem oriented record as a basic tool in medical education, patient care and clinical research'. In: *Annals of Clinical Research* 3.3 (1971), p. 131.

[67] E. Bayegan and O. Nytro. 'A problem-oriented, knowledge-based patient record system'. In: *Studies in health technology and informatics* (2002), pp. 272–276.

[68] Pallas Athena. *Case Handling with FLOWer: Beyond workflow*. Formerly available as `http://www.pallas-athena.com/downloads/eng_flower/flowerwp.pdf`. 2002.

[69] Wil M. P. van der Aalst, M. Weske, and D. Grünbauer. 'Case handling: a new paradigm for business process support'. In: *Data & Knowledge Engineering* 53.2 (2005), pp. 129–162.

[70] Lutz J. Heinrich. *Wirtschaftsinformatik: Einführung und Grundlegung*. Oldenbourg, 1993. ISBN: 3486224638.

[71] Richard Lenz. 'Evolutionäre Informationssysteme'. Habilitation thesis. Philipps-Universität Marburg, 2005.

[72] Fraunhofer Institut. *Spezifikation der Lösungsarchitektur zur Umsetzung der Anwendungen der elektronischen Gesundheitskarte*. `http://www.ehealthopen.com/FuE/PDF/eGK-Spez-Loesarch-v1.0.pdf`. Version 1.0. Mar. 2005.

[73] TU-Wien. *T-Stich: Architektur der dezentralen Dienste und Komponenten – Eine operable Architektur*. `http://www.ehealthopen.com/FuE/PDF/eGK-T-Stich-v1.0.pdf`. Version 1.0. Mar. 2005.

[74] T. C. Thomas Rindfleisch. 'Privacy, information technology, and health care'. In: *Communications of the ACM* 40.8 (1997), pp. 92–100.

[75] F. K. Ueckert and H.-U. Prokosch. 'Implementing security and access control mechanisms for an electronic healthcare record.' In: *Proc of the AMIA Annual Symposium*. American Medical Informatics Association. 2002, p. 825.

[76] Fraunhofer Institute for Biomedical Engineering (IBMT). *PaDok – Patientenbegleitende Dokumentation*. `http://www.ibmt.fraunhofer.de/fhg/Images/MT_padoknetzkonzept_de_tcm266-68980.pdf`. 2000.

[77] ACC, HIMSS, and RSNA. *IHE IT Infrastructure Technical Framework, vol. 1 (ITI TF-1): Integration Profiles, Rev 4.0*. `http://www.ihe.net/Technical_Framework/upload/IHE_ITI_TF_4_0_Vol1_FT_2007_08_22.pdf`. Aug. 2007.

[78] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. *(RFC 2440): OpenPGP message format*. Tech. rep. Internet Engineering Task Force (IETF), Nov. 1998.

[79] M. Gasser. *Building a secure computer system*. New York, NY, USA: Van Nostrand Reinhold, 1988. ISBN: 0442230222.

[80] J. H. Saltzer and M. D. Schroeder. 'The protection of information in computer systems'. In: *Proceedings of the IEEE* 63.9 (1975), pp. 1278–1308.

[81] Object Management Group, Inc. (OMG). *Person Identification Service (PIDS) Specification*. `http://www.omg.org/technology/documents/formal/index.htm`. Apr. 2001.

[82] Gernot Roth. 'Konzeption und Realisierung eines verteilten Master Patient Index'. Diplomarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, June 2009.

[83] P. J. Leach, M. Mealling, and R. Salz. *(RFC 4122:) A Universally Unique IDentifier (UUID) URN Namespace*. Tech. rep. Internet Engineering Task Force (IETF), July 2005.

[84] Oracle and/or its affiliates. *API Specification for Class java.util.UUID*. `http://docs.oracle.com/javase/6/docs/api/java/util/UUID.html`. 2011.

[85] Christoph Bussler. 'Organisationsverwaltung in Workflow-Management-Systemen'. PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, 1997.

[86] F. Dawson and T. Howes. *(RFC 2426:) vCard MIME directory profile*. Tech. rep. Internet Engineering Task Force (IETF), Sept. 1998.

[87] Igor Engel. 'Konzeption und Implementierung einer verteilten Institutionsverwaltung als anwendungsspezifische Form eines verteilten Metadaten-Repository'. Studienarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, July 2011.

[88] Alistair Miles and Sean Becchofer. *SKOS: Simple Knowledge Organization System Reference. W3C Recmmendation*. Tech. rep. World Wide Web Consortium (W3C), Aug. 2009.

[89] R. Lenz, M. Beyer, and K. A. Kuhn. 'Semantic integration in healthcare networks'. In: *Int'l Journal of Medical Informatics* 76.2-3 (2006), pp. 201–207.

[90] Florian Wagner. 'Entwurf und Realisierung eines IHE XDS Komponenten-Teststands'. Studienarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Jan. 2009.

[91] Christoph P. Neumann, Florian Wagner, and Richard Lenz. 'XdsRig – Eine Open-Source IHE XDS Testumgebung'. In: *Tagungsband der 54. GMDS-Jahrestagung*. Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS). Essen, DE, Sept. 2009.

[92] Christoph P. Neumann, Stefan Hanisch, Bernhard Schiemann, and Richard Lenz. 'OXDBS – Erweiterung einer nativen XML-Datenbank um die Validierung und Konsistenzprüfung gegen eine OWL-Ontologie'. In: *Tagungsband der 54. GMDS-Jahrestagung*. Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS). Essen, DE, Sept. 2009.

[93] Christoph P. Neumann, Thomas Fischer, and Richard Lenz. 'OXDBS – Extension of a native XML Database System with Validation by Consistency Checking of OWL-DL Ontologies'. In: *Proc of the 14th International Database Engineering & Applications Symposium (IDEAS'10)*. Montreal, QC, CA, Aug. 2010.

[94] Florian Rampp. 'Design and Implementation of a Distributed Address Database Following a Publish/Subscribe Architecture to Share Patient Data Among Autonomous Healthcare Information Systems'. Diplomarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Mar. 2009.

[95] Christoph P. Neumann and Richard Lenz. 'A Light-Weight System Extension Supporting Document-based Processes in Healthcare'. In: *Proc of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'09) in conjunction with the 7th Int'l Conf on Business Process Management (BPM'09)*. Ulm, DE, Sept. 2009.

[96] Roy T. Fielding. 'Architectural Styles and the Design of Network-based Software Architectures'. PhD thesis. University of California, USA, 2000.

[97] Christoph P. Neumann, Florian Rampp, Michael Daum, and Richard Lenz. 'A Mediated Publish-Subscribe System for Inter-Institutional Process Support in Healthcare'. In: *Proc of the 3rd ACM Int'l Conf on Distributed Event-Based Systems (DEBS 2009)*. Nashville, TN, USA, July 2009.

[98] eHealth. *Ministerial Declaration, Brussels*. `http://europa.eu.int/information_society/eeurope/ehealth/Conf/2003/doc/min_dec_22_may_03.pdf`. May 2003.

[99] S. Santana, B. Lausen, M. Bujnowska-Fedak, C. Chronaki, H.-U. Prokosch, and R. Wynn. 'Informed citizen and empowered citizen in health: results from an European survey'. In: *BMC Family Practice* 12.1 (2011), p. 20.

[100] Eclipse project. *Higgins Open Source Identity Framework*. `http://www.eclipse.org/higgins/`. 2007.

[101] Christoph P. Neumann, Florian Rampp, and Richard Lenz. *DEUS: Distributed Electronic Patient File Update System*. Tech. rep. CS-2012-02. University of Erlangen, Dept. of Computer Science, Mar. 2012.

[102] G. Britain. 'Computerisation of personal health records'. In: *Health Visit* 51 (1978), p. 227.

[103] Dean F. Sittig. 'Personal health records on the internet: a snapshot of the pioneers at the end of the 20th Century'. In: *Int'l Journal of Medical Informatics* 65.1 (2002), pp. 1–6.

[104] Google. *Google Health Data API CCR Reference*. Tech. rep. The Google Health Data API has been fully retired as of January 1, 2012 and is no longer available. June 2011.

[105] IHE: Integrating the Healthcare Enterprise. *IHE Patient Care Coordination (PCC), Technical Framework, Volume 1, Revision 6.0*. `www.ihe.net/Technical_framework/upload/IHE_PCC_TF_Rev6-0_Vol_1_2010-08-30.pdf`. Aug. 2010.

[106] J. C. Schwarze, S. Tessmann, C. Sassenberg, M. Müller, H.-U. Prokosch, and F. Ückert. 'Eine modulare Gesundheitsakte als Antwort auf Kommunikationsprobleme im Gesundheitswesen'. In: *Wirtschaftsinformatik* 47.3 (2005), pp. 187–195.

[107] Drummond Reed and Dave McAlpin. *Extensible Resource Identifier (XRI) Syntax V2.0 Committee Specification*. OASIS XRI Technical Committee. Nov. 2005.

[108] Drummond Reed and Geoffrey Strongin. *The Dataweb: An Introduction to XDI. A White Paper for the OASIS XDI Technical Committee v2*. OASIS XDI Technical Committee. Apr. 2004.

[109] Drummond Reed and Markus Sabadello. *The XDI RDF Model*. OASIS XDI Technical Committee. Jan. 2010.

[110] T. Weichert. 'Die elektronische Gesundheitskarte'. In: *Datenschutz und Datensicherheit* 28.7 (2004), pp. 391–403.

[111] Peter Mertens. 'Fehlschläge bei IT-Großprojekten der öffentlichen Verwaltung–ein Beitrag zur Misserfolgsforschung in der Wirtschaftsinformatik'. In: *Multikonferenz Wirtschaftsinformatik (MKWI 2008): Die Wirtschaftsinformatik im Spannungsfeld zwischen Vielfalt und Profilbildung – Auf der Suche nach den Kernkompetenzen einer vielfältigen Disziplin*. Ed. by M. Bichler. Feb. 2008, pp. 1085–1100.

[112] Fraunhofer Institut. *Fachlogische Modellierung und spezifische Anwendungsdienste der elektronischen Gesundheitskarte*. `http://www.ehealthopen.com/FuE/PDF/eGK-Fachmodlanw-v1.0.pdf`. Mar. 2005.

[113] C. Batini, M. Lenzerini, and S. B. Navathe. 'A comparative analysis of methodologies for database schema integration'. In: *ACM Computing Surveys (CSUR)* 18.4 (1986), pp. 323–364.

[114] S. Heiler. 'Semantic interoperability'. In: *ACM Computing Surveys (CSUR)* 27.2 (1995), pp. 271–273.

[115] Robert M. Colomb. 'Impact of semantic heterogeneity on federating databases'. In: *The Computer Journal* 40.5 (1997), pp. 235–244.

[116] R. G. Berger and J. Baba. 'The realities of implementation of Clinical Context Object Workgroup (CCOW) standards for integration of vendor disparate clinical software in a large medical center'. In: *International Journal of Medical Informatics* 78.6 (2009), pp. 386–390.

[117] R. Lenz, T. Elstner, H. Siegele, and K. A. Kuhn. 'A practical approach to process support in health information systems'. In: *Journal of the American Medical Informatics Association* 9.6 (2002), pp. 571–585.

[118] C. J. McDonald, J. Marc Overhage, P. Dexter, B. Takesue, and J. G. Suico. 'What is done, what is needed and what is realistic to expect from medical informatics standards'. In: *Int'l Journal of Medical Informatics* 48.1-3 (1998), pp. 5–12.

[119] C. Szyperski, D. Gruntz, and S. Murer. *Component software: beyond object-oriented programming*. Addison-Wesley Professional, 2002.

[120] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall, 2005. ISBN: 0131858580.

[121] R. Lenz and K. A. Kuhn. 'A strategic approach for business-IT alignment in health information systems'. In: *Lecture notes in computer science* (2003), pp. 178–195.

[122] David S. Linthicum. 'B2B Process Integration'. In: *eAI Journal* (2000), pp. 50–56.

[123] David S. Linthicum. *Next generation application integration: from simple information to Web services*. Addison-Wesley, 2004.

[124] C. Hentrich and U. Zdun. 'Patterns for process-oriented integration in service-oriented architectures'. In: *Proc of 11th European Conf on Pattern Languages of Programs (EuroPlop 2006)*. 2006, pp. 141–189.

[125] T. Kobayashi, M. Tamaki, and N. Komoda. 'Business process integration as a solution to the implementation of supply chain management systems'. In: *Information & Management* 40.8 (2003), pp. 769–780.

[126] S. Kumaran and K. Bhaskaran. 'Business Process Integration'. In: *Supply Chain Management on Demand* (2005), pp. 211–232.

[127] Jerry Luftman. 'Assessing Business-IT Alignment Maturity'. In: *Strategies for Information Technology Governance*. Ed. by Wim van Grembergen. IGI Global, 2003. Chap. 4, p. 99. ISBN: 1591402840.

[128] James Martin. *Rapid Application Development*. Macmillan, 1991. ISBN: 0023767758.

[129] Meir M. Lehman. 'Programs, life cycles, and laws of software evolution'. In: *Proceedings of the IEEE* 68.9 (1980), pp. 1060–1076.

[130] Meir M. Lehman. 'Program evolution'. In: *Information Processing & Management* 20.1 (1984), pp. 19–36.

[131] Meir M. Lehman and L. A. Belady. 'Program Evolution – Processes of Software Change'. In: (1985).

[132] S. Cook, R. Harrison, M. M. Lehman, and P. Wernick. 'Evolution in software systems: foundations of the SPE classification scheme'. In: *Journal of Software Maintenance and Evolution: Research and Practice* 18.1 (2006), pp. 1–35.

[133] Meir M. Lehman and Juan F. Ramil. 'Software Evolution – Background, Theory, Practice'. In: *Information Processing Letters* 88.1-2 (2003), pp. 33–44.

[134] Meir M. Lehman. 'Laws of software evolution revisited'. In: *Software Process Technology* (1996), pp. 108–124.

[135] Adam Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations*. 1776.

[136] J. Becker, L. Algermissen, and B. Niehaves. 'Processes in e-government focus: a procedure model for process oriented reorganisation in public administrations on the local level'. In: *Electronic Government* (2003), pp. 1062–1062.

[137] R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. 'The action workflow approach to workflow management technology'. In: *Proceedings of the 1992 ACM conference on Computer-supported cooperative work*. ACM. 1992, pp. 281–288.

[138] D. Georgakopoulos, M. Hornick, and A. Sheth. 'An overview of workflow management: From process modeling to workflow automation infrastructure'. In: *Distributed and parallel Databases* 3.2 (1995), pp. 119–153.

[139] Michael Hammer. 'Reengineering work: don't automate, obliterate'. In: *Harvard business review* 68.4 (1990), pp. 104–112.

[140] Michael Hammer and James Champy. *Reengineering the corporation: A manifesto for business revolution*. HarperBusiness, 1993. ISBN: 1863735054.

[141] Mathias Weske and Gottfried Vossen. *Workflow Languages*. Tech. rep. `http://www.xforms-editor.org/pub/Public/PaperArchive/springer.pdf`. Westfälische Welhelms-Universität Münster, 1997.

[142] Chris Peltz. 'Web services orchestration and choreography'. In: *Computer* 36.10 (2003), pp. 46–52.

[143] Workflow Management Coalition. *WFMC-TC-1011 Ver 3 Terminology and Glossary English*. `http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html`. Feb. 1999.

[144] M. Rusinkiewicz and A. Sheth. 'Specification and execution of transactional workflows'. In: *Modern Database Systems: The Object Model, Interoperability, and Beyond*. Ed. by Won Kim. Addison-Wesley, 1994, pp. 592–620. ISBN: 0-201-59098-0.

[145] S. McCready. 'There is more than one kind of workflow software'. In: *Computerworld* 2 (1992), pp. 86–90.

[146] Y. Kubera, P. Mathieu, and S. Picault. 'Everything can be Agent!' In: *Proc of the 9th Int'l Conf on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems. 2010, pp. 1547–1548.

[147] Nandish V. Patel. *Adaptive Evolutionary Information Systems*. Idea Group Inc, 2002. ISBN: 1591400341.

[148] Liora Alschuler and Kai U. Heitmann. 'CDA Introductory Tutorial'. In: *HL7 International CDA Conference*. Oct. 2002.

[149] Paul Krill. *JavaScript creator ponders past, future – Mozilla's Brendan Eich describes JavaScript's history, the upcoming upgrade, and disagreements with Microsoft*. `http://www.infoworld.com/d/developer-world/javascript-creator-ponders-past-future-704/`. June 2008.

[150] Ecma International. *ECMAScript Language Specification (ECMA-262), 5.1 Edition*. `http://www.ecma-international.org/publications/standards/Ecma-262.htm`. June 2011.

[151] C. Dony, J. Malenfant, and P. Cointe. 'Prototype-based languages: from a new taxonomy to constructive proposals and their validation'. In: *Proc of the 7th Int'l Conf on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA'92)*. Oct. 1992, pp. 201–217.

[152] Peter Michaux. *Transitioning from Java Classes to JavaScript Prototypes*. `http://michaux.ca/index`. Oct. 2007.

[153] W. P. Stevens, G. J. Myers, and L. L. Constantine. 'Structured Design'. In: *IBM Systems Journal* 13.2 (1974), pp. 115–139.

[154] C. Ghezzi, M. Jazayeri, and D. Mandrioli. *Fundamentals of software engineering*. Vol. 2. Prentice Hall Indianapolis, 1991. ISBN: 0133056996.

[155] C/S2ESC Software and Systems Engineering Standards Committee. *(IEEE Std. 610.12-1990:) Standard Glossary of Software Engineering Terminology*. Tech. rep. Institute of Electrical and Electronics Engineers (IEEE), 1990.

[156] B. Nitzberg and V. Lo. 'Distributed shared memory: A survey of issues and algorithms'. In: *Computer* 24.8 (1991), pp. 52–60.

[157] D. Krafzig, K. Banke, and D. Slama. *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall PTR, 2005. ISBN: 0131465759.

[158] N. Josuttis. *SOA in Practice: The Art of Distributed System Design*. O'reilly, 2007. ISBN: 0596529554.

[159] Volker Stiehl. 'Composite Application Systems'. PhD thesis. Technische Universität Darmstadt, Germany, 2011.

[160] Russell Butek. *Which style of WSDL should I use*. `http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/`. May 2005.

[161] G. Hohpe and B. Woolf. *Enterprise Integration Patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley, 2003.

[162] William Stallings. *Operating Systems: Internals and Design Principles*. Prentice Hall, 2000. ISBN: 0130319996.

[163] David Cummings. *Push Vs. Pull – The Battle for the Best CMS*. `http://www.sitepoint.com/push-pull-best-cms/`. Feb. 2005.

[164] Z. Duan, K. Gopalan, and Y. Dong. 'Push vs. Pull: Implications of protocol design on controlling unwanted traffic'. In: *Proc of USENIX Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI 2005)*. July 2005.

[165] Markus Lorch, Seth Proctor, Rebekah Lepro, Dennis Kafura, and Sumit Shah. 'First experiences using XACML for access control in distributed systems'. In: *Proc of the ACM Workshop on XML security (XMLSEC'03)*. Fairfax, VA, USA, Oct. 2003, pp. 25–37.

[166] S. Quaglini, M. Stefanelli, A. Cavallini, G. Micieli, C. Fassino, and C. Mossa. 'Guideline-based careflow systems'. In: *Artificial Intelligence in Medicine* 20.1 (2000), pp. 5–22.

[167] L. Anselma, A. Bottrighi, G. Molino, S. Montani, P. Terenziani, and M. Torchio. 'Supporting Knowledge-Based Decision Making in the Medical Context'. In: *Int'l Journal of Knowledge-Based Organizations (IJKBO)* 1.1 (2011), pp. 42–60.

[168] Erich Ortner. 'Component-based application architecture for enterprise information systems'. In: *Data Management in a Connected World (Essays Dedicated to Hartmut Wedekind on the Occasion of His 70th Birthday)*. Ed. by Theo Härder and Wolfgang Lehner. Vol. 3551. Lecture Notes in Computer Science. Springer, 2005, pp. 315–334.

[169] Jesper Boeg. *Priming Kanban – A 10 step guid to optimizing flow in your software delivery system*. Trifork A/S, 2011.

[170] H. Kniberg. *Scrum and XP from the Trenches*. Enterprise Software Development. Lulu Enterprises, 2007. ISBN: 1430322640.

[171] Corey Ladas. *Scrumban – Essays on Kanban Systems for Lean Software Development*. Modus Cooperandi Press, 2009. ISBN: 0578002140.

[172] K. Schwaber et al. 'Scrum development process'. In: *Proc of the OOPSLA Business Object Design and Implementation Workshop*. Vol. 27. Austin, TX, USA, 1995, pp. 10–19.

[173] D. K. Sobek, A. C. Ward, and J. K. Liker. 'Toyota's principles of set-based concurrent engineering'. In: *Sloan Management Review* 40.2 (1999), pp. 67–84.

[174] J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov. 'Distributed Scrum: Agile Project Management with Outsourced Development Teams'. In: *Proc of the 40th Annual Hawaii Int'l Conf on System Sciences (HICSS-40)*. Jan. 2007, 274a–274a.

[175] Anne Brüggemann-Klein. *Document Engineering im World-Wide Web*. `http://www11.in.tum.de/dokument.php?id_dokument=255`. 2002.

[176] David M. Levy. 'Topics in document research'. In: *Proc of the ACM Conf on Document Processing Systems (DOCPROCS'88)*. Nov. 1988, pp. 187–193.

[177] Linda Schamber. 'What is a document? Rethinking the concept in uneasy times'. In: *Journal of the American Society for Information Science* 47.9 (1996), pp. 669–671.

[178] P. Dourish, W. K. Edwards, A. LaMarca, J. Lamping, K. Petersen, M. Salisbury, D. B. Terry, and J. Thornton. 'Extending document management systems with user-specific active properties'. In: *ACM Transactions on Information Systems (TOIS)* 18.2 (2000), pp. 140–170.

[179] W. K. Edwards, J. P. Dourish, A. G. Lamarca, J. O. Lamping, K. Petersen, M. F. Salisbury, D. B. Terry, J. D. Thornton, et al. *Extending application behavior through active properties attached to a document in a document management system*. `http://www.google.de/patents?id=KNAIAAAAEBAJ`. Xerox Corporation, US Patent No.: 6,562,076 B1. May 2003.

[180] C. Fay. 'The Document Management Alliance'. In: *Bulletin of the American Society for Information Science and Technology* 25.1 (1998), pp. 20–24.

[181] Bill Nowicki. *(RFC 1094:) NFS: Network File System Protocol Specification*. Tech. rep. Internet Engineering Task Force (IETF), Mar. 1989.

[182] D. Hitz, J. Lau, and M. Malcolm. 'File system design for an NFS file server appliance'. In: *Proc of the USENIX Winter Technical Conference*. Jan. 1994, pp. 19–19.

[183] Y. Goland, E. Whitehead, A. Faizi, S. Carter, and D. Jensen. *(RFC 2518:) HTTP Extensions for Distributed Authoring – WEBDAV*. Tech. rep. Internet Engineering Task Force (IETF), Feb. 1999.

[184] D. Giampaolo. *Practical file system design with the Be File System*. Morgan Kaufmann Publishers Inc., 1998. ISBN: 1558604979.

[185] E. Heinrich and H. Maurer. 'Active documents: Concept, implementation and applications'. In: *Journal of Universal Computer Science* 6.12 (2000), pp. 1197–1202.

[186] S. K. Chang and T. Znati. 'Adlet: an active document abstraction for multimedia information fusion'. In: *IEEE Transactions on Knowledge and Data Engineering* 13.1 (2001), pp. 112–123.

[187] P. Werle, F. Kilander, M. Jonsson, P. Lönnqvist, and C. Jansson. 'A ubiquitous service environment with active documents for teamwork support'. In: *Proc of the Int'l Symposium on Ubiquitous Computing (Ubicomp'2001)*. Oct. 2001, pp. 139–155.

[188] Steffen Idler. 'Recherche und vergleichende Evaluation von verfügbaren Ansätzen für 'Aktive Dokumente''. Bachelorarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Sept. 2010.

[189] Alan C. Kay. 'The Early History of Smalltalk'. In: *SIGPLAN Notices* 28.3 (Mar. 1993), pp. 69–95.

[190] B. C. Pierce. *Types and programming languages*. The MIT Press, 2002. ISBN: 0262162091.

[191] M. Wooldridge and N. R. Jennings. 'Intelligent agents: Theory and practice'. In: *Knowledge engineering review* 10.2 (1995), pp. 115–152.

[192] M. Eichelberg, T. Aden, J. Riesmeier, A. Dogac, and G. B. Laleci. 'A survey and analysis of Electronic Healthcare Record standards'. In: *ACM Computing Surveys (CSUR)* 37.4 (2005), pp. 277–315.

[193] J. Lahteenmaki, J. Leppanen, and H. Kaijanranta. 'Interoperability of personal health records'. In: *Proc of the Annual Int'l Conf of the IEEE Engineering in Medicine and Biology Society (EMBC 2009)*. Sept. 2009, pp. 1726–1729.

[194] World Health Organization. *International Classification of Diseases (ICD)*. `http://www.who.int/classifications/icd/`. Published by the U.S. Department of Health and Human Services (HSS), Centers for Disease Control and Prevention (CDC), and Health Care Financing Administration (HFCA). 2010.

[195] M. Q. Stearns, C. Price, K. A. Spackman, and A. Y. Wang. 'SNOMED clinical terms: overview of the development process and project status'. In: *Proc of the AMIA Annual Symposium*. 2001, p. 6.

[196] A. W. Forrey, C. J. McDonald, G. DeMoor, S. M. Huff, D. Leavelle, D. Leland, T. Fiers, L. Charles, B. Griffin, F. Stalling, et al. 'Logical observation identifier names and codes (LOINC) database: a public use set of codes and names for electronic reporting of clinical laboratory test results'. In: *Clinical Chemistry* 42.1 (1996), pp. 81–90.

[197] J. Brender, E. Ammenwerth, P. Nykanen, and J. Talmon. 'Factors influencing success and failure of health informatics systems–a pilot Delphi study.' In: *Methods Inf Med* 45.1 (2006), pp. 125–36.

[198] HL7. *Health Level Seven Standard Version 2.6 – An Application Protocol for Electronic Data Exchange in Healthcare Environments (ANSI/HL7 V2.6-2007)*. `http://www.hl7.org/Library/standards_non1.htm`. 2007.

[199] Richard M. Peters and Joseph H. Schneider. *ASTM E2369 - 05e2 Standard Specification for Continuity of Care Record (CCR)*. Tech. rep. ASTM International, Subcommittee E31.25, 2005.

[200] J. H. Sutanto and H. L. Seldon. 'Translation between HL7 v2.5 and CCR message formats (For communication between hospital and personal health record systems)'. In: *Proc of the IEEE Conf on Open Systems (ICOS 2011)*. Sept. 2011, pp. 406–410.

[201] Kai U. Heitmann, Guido Noelle, and Ralf Schweiger. *SCIPHOX v1.0, Working Draft 15*. `http://sciphox.hl7.de/atwork/tools/WD-sciphox-v15.pdf`. June 2002.

[202] R. H. Dolin, L. Alschuler, C. Beebe, P. V. Biron, S. L. Boyer, D. Essin, E. Kimber, T. Lincoln, and J. E. Mattison. 'The HL7 Clinical Document Architecture'. In: *Journal of the American Medical Informatics Association (JAMIA)* 8.6 (2001), pp. 552–569.

[203] Jeffrey M. Ferranti, Clayton Musser, Kensaku Kawamoto, and Ed Hammond. 'The Clinical Document Architecture and the Continuity of Care Record'. In: *Journal of the American Medical Informatics Association (JAMIA)* 13.3 (2006), p. 245.

[204] K. U. Heitmann, R. Schweiger, and J. Dudeck. 'Discharge and referral data exchange using global standards – the SCIPHOX project in Germany'. In: *Int'l Journal of Medical Informatics* 70.2-3 (2003), pp. 195–203.

[205] eArztbrief – D2D Telematik-Plattform der Kassenärztlichen Vereinigungen. *Elektronischer Arztbrief im D2D-System*. `http://www.d2d.de/index.php?id=17`. 2005.

[206] W. D. Bidgood, S. C. Horii, F. W. Prior, and D. E. Van Syckle. 'Understanding and Using DICOM, the Data Interchange Standard for Biomedical Imaging'. In: *Journal of the American Medical Informatics Association (JAMIA)* 4.3 (1997), pp. 199–212.

[207] E. L. Siegel and D. S. Channin. 'Integrating the Healthcare Enterprise: A Primer'. In: *RadioGraphics – The Journal of Continuing Medical Education in Radiology* 21.5 (2001), pp. 1339–1341.

[208] M. Tsiknakis, D. G. Katehakis, and S. C. Orphanoudakis. 'An open, component-based information infrastructure for integrated health information networks'. In: *Int'l Journal of Medical Informatics* 68.1-3 (2002), pp. 3–26.

[209] D. G. Katehakis, M. Tsiknakis, and S. C. Orphanoudakis. 'Enabling Components of HYGEIAnet'. In: *Proc. of TEPR*. 2001, pp. 146–153.

[210] Yuval Shahar. *Automated support to clinical guidelines and care plans: the intention-oriented view*. `http://www.openclinical.org/docs/int/briefingpapers/shahar.pdf`. 2002.

[211] Paul A. de Clercq, Johannes A. Blom, Hendrikus H. M. Korsten, and Arie Hasman. 'Approaches for creating computer-interpretable guidelines that facilitate decision support'. In: *Artificial Intelligence in Medicine* 31.1 (2004), pp. 1–27.

[212] M. Peleg, S. Tu, J. Bury, P. Ciccarese, J. Fox, R. A. Greenes, R. Hall, P. D. Johnson, N. Jones, A. Kumar, et al. 'Comparing computer-interpretable guideline models: a case-study approach'. In: *Journal of the American Medical Informatics Association (JAMIA)* 10.1 (2003), pp. 52–68.

[213] Ayda I. Arruda. 'A survey of Paraconsistent Logic ()'. In: *Studies in Logic and the Foundations of Mathematics* 99 (1980), pp. 1–41.

[214] K. Miller and W. MacCaull. 'Toward web-based careflow management systems'. In: *Journal of Emerging Technologies in Web Intelligence* 1.2 (2009), pp. 137–145.

[215] S. Panzarasa, S. Madde, S. Quaglini, C. Pistarini, and M. Stefanelli. 'Evidence-based careflow management systems: the case of post-stroke rehabilitation'. In: *Journal of Biomedical Informatics* 35.2 (2002), pp. 123–139.

[216] S. Panzarasa and M. Stefanelli. 'Workflow management systems for guideline implementation'. In: *Neurological Sciences* 27 (2006), pp. 245–249.

[217] A. A. Boxwala, S. Tu, M. Peleg, Q. Zeng, O. Ogunyemi, R. A. Greenes, E. H. Shortliffe, and V.L. Patel. 'Toward a representation format for sharable clinical guidelines'. In: *Journal of Biomedical Informatics* 34.3 (2001), pp. 157–169.

[218] D. Wang, M. Peleg, S. W. Tu, A. A. Boxwala, R. A. Greenes, V. L. Patel, and E. H. Shortliffe. 'Representation primitives, process models and patient data in computer-interpretable clinical practice guidelines: A literature review of guideline representation models'. In: *International Journal of Medical Informatics* 68.1-3 (2002), pp. 59–70.

[219] M. Peleg, A. A. Boxwala, S. Tu, Q. Zeng, O. Ogunyemi, D. Wang, V. L. Patel, R. A. Greenes, and E. H. Shortliffe. 'The InterMed approach to sharable computer-interpretable guidelines: a review'. In: *Journal of the American Medical Informatics Association (JAMIA)* 11.1 (2004), pp. 1–10.

[220] M. Peleg and S. Tu. 'Decision support, knowledge representation and management in medicine'. In: *IMIA Yearbook 2006 – Yearbook of Medical Informatics as a supplement of Methods of Information in Medicine* (2006). Ed. by Antoine Geissbuhler and Casimir Kulikowski, pp. 72–80.

[221] F. T. Imam, W. MacCaull, and M. A. Kennedy. 'Merging healthcare ontologies: Inconsistency tolerance and implementation issues'. In: *Proc of 20th IEEE Int'l Symp on Computer-Based Medical Systems (CBMS'07)*. June 2007, pp. 530–535.

[222] F. Imam and W. MacCaull. 'Integrating healthcare ontologies: An inconsistency tolerant approach and case study'. In: *Proc of the 2rd Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'09) in conjunction with the 6th Int'l Conf on Business Process Management (BPM'09)*. Sept. 2008, pp. 373–384.

[223] Oliver Kopp, Daniel Martin, Daniel Wutke, and Frank Leymann. 'On the Choice Between Graph-Based and Block-Structured Business Process Modeling Languages'. In: *Modellierung betrieblicher Informationssysteme (MobIS 2008)*. Vol. 141. Lecture Notes in Informatics (LNI). Gesellschaft für Informatik e.V. (GI), Nov. 2008, pp. 59–72.

[224] Object Management Group, Inc. (OMG). *Business Process Model and Notation (BPMN), Version 2.0*. `http://www.omg.org/spec/BPMN/2.0/`. Jan. 2011.

[225] Wil M. P. van der Aalst and T. Basten. 'Inheritance of workflows: an approach to tackling problems related to change'. In: *Theoretical Computer Science (TCS)* 270.1-2 (2002), pp. 125–203.

[226] Bruce Silver. *BPMN method and style*. Cody-Cassidy, 2009. ISBN: 0982368100.

[227] S. A. White and D. Miers. *BPMN modeling and reference guide*. Future Strategies Inc., 2008. ISBN: 0977752720.

[228] Thomas Allweyer. *BPMN 2.0 – Business Process Model and Notation: Einführung in den Standard für die Geschäftsprozessmodellierung*. Bod, 2009. ISBN: 3839121344.

[229] Bruce Silver. *Dialog with Dumas on Roundtripping*. `http://www.brsilver.com/2007/11/30/dialog-with-dumas-on-roundtripping/`. Nov. 2007.

[230] Michael zur Muehlen. *Business Process Management Standards Tutorial*. `http://bpm07.fit.qut.edu.au/program/slides/Thursday/Thursday-Tutorials/Muehlen.pdf`. Howe School of Technology Management. 2007.

[231] Benedikt Lempetzeder. 'Gegenüberstellung verschiedener Paradigmen zur Darstellung von Prozesseigenschaften unter Berücksichtigung von Zeit und Daten'. Bachelorarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Sept. 2011.

[232] Wil M. P. Van Der Aalst, P. Barthelmess, C. A. Eliis, and J. Wainer. 'Proclets: A framework for lightweight interacting workflow processes'. In: *Int'l Journal of Cooperative Information Systems* 10.4 (2001), pp. 443–482.

[233] R. S. Mans, N. C. Russell, Wil M. P. van der Aalst, P. J. M. Bakker, A. J. Moleman, and M. W. M. Jaspers. 'Proclets in healthcare'. In: *Journal of Biomedical Informatics* 43.4 (2010), pp. 632–649.

[234] Wil M. P. Van Der Aalst and M. Pesic. 'DecSerFlow: Towards a truly declarative service flow language'. In: *Web Services and Formal Methods* (2006), pp. 1–23.

[235] R. Gerth, D. Peled, M. Y. Vardi, and P. Wolper. 'Simple on-the-fly automatic verification of linear temporal logic'. In: *Proc of Int'l Symposium on Protocol Specification Testing and Verification (PSTV'95)*. June 1995, pp. 3–18.

[236] A. P. Sistla and E. M. Clarke. 'The complexity of propositional linear temporal logics'. In: *Journal of the ACM (JACM)* 32.3 (1985), pp. 733–749.

[237] M. Reichert and P. Dadam. 'ADEPTflex – supporting dynamic changes of workflows without losing control'. In: *Journal of Intelligent Information Systems* 10.2 (1998), pp. 93–129.

[238] C. P. Gane and T. Sarson. *Structured systems analysis: tools and techniques*. Prentice Hall Professional Technical Reference, 1979. ISBN: 0138545472.

[239] Peter Checkland and Jim Scholes. *Soft Systems Methodology in Action*. John Wiley & Sons, 1990. ISBN: 0471927686.

[240] Jean-Jacques Dubray. *The Seven Fallacies of Business Process Execution*. `http://www.infoq.com/articles/seven-fallacies-of-bpm/`. Dec. 2007.

[241] W. Prinz and S. Kolvenbach. 'Support for workflows in a ministerial environment'. In: *Proceedings of the 1996 ACM conference on Computer supported cooperative work*. 1996, pp. 199–208.

[242] Dominic Müller, Manfred Reichert, and Joachim Herbst. 'Flexibility of Data-Driven Process Structures'. In: *Business Process Management Workshops 2006: BPD, BPI, ENEI, GPWW, DPM, semantics4ws*. Ed. by Johann Eder and Schahram Dustdar. Vol. 4103. Lecture Notes in Computer Science. Springer, 2006, pp. 181–192.

[243] Dominic Müller, Manfred Reichert, and Joachim Herbst. 'Data-Driven Modeling and Coordination of Large Process Structures'. In: *On the Move to Meaningful Internet Systems 2007: CoopIS, DOA, ODBASE, GADA, and IS*. Ed. by Robert Meersman and Zahir Tari. Vol. 4803. Lecture Notes in Computer Science. Springer, 2007, pp. 131–149.

[244] Dominic Müller, Manfred Reichert, and Joachim Herbst. 'A new paradigm for the enactment and dynamic adaptation of data-driven process structures'. In: *Proc of the 20th Int'l Conf on Advanced Information Systems Engineering (CAiSE'08)*. July 2008, pp. 48–63.

[245] Jianrui Wang and Akhil Kumar. 'A Framework for Document-Driven Workflow Systems'. In: *Business Process Management*. Ed. by Wil M. P. van der Aalst, Boualem Benatallah, Fabio Casati, and Francisco Curbera. Vol. 3649. Lecture Notes in Computer Science. Springer, 2005, pp. 285–301.

[246] Akhil Kumar and Jianrui Wang. 'A Framework for Designing Resource-Driven Workflows'. In: *Handbook on Business Process Management 1*. Ed. by Jan vom Brocke and Michael Rosemann. International Handbooks on Information Systems. Springer, 2010, pp. 419–440.

[247] Kamal Bhattacharya, Cagdas Gerede, Richard Hull, Rong Liu, and Jianwen Su. 'Towards formal analysis of artifact-centric business process models'. In: *Proc of the 5th Int'l Conf on Business Process Management (BPM'07)*. Brisbane, AU, Sept. 2007, pp. 288–304.

[248] David Cohn and Richard Hull. 'Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes'. In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* (Sept. 2009).

[249] Diego Calvanese, Giuseppe De Giacomo, Richard Hull, and Jianwen Su. 'Artifact-Centric Workflow Dominance'. In: *Service-Oriented Computing*. Ed. by Luciano Baresi, Chi-Hung Chi, and Jun Suzuki. Vol. 5900. Lecture Notes in Computer Science. Springer, 2009, pp. 130–143.

[250] Vera Künzle and Manfred Reichert. 'Towards Object-Aware Process Management Systems: Issues, Challenges, Benefits'. In: *Proc of the 10th Int'l Workshop on Enterprise, Business-Process and Information Systems Modeling (BPMDS'09) in conjunction with the 14th Int'l Conf on Exploring Modelling Methods for Systems Analysis and Design (EMMSAD'09), held in conjunction with the 21st Int'l Conf on Advanced Information Systems Engineering (CAiSE'09)*. Springer, June 2009, pp. 197–210.

[251] Vera Künzle and Manfred Reichert. 'Herausforderungen bei der Integration von Benutzern in Datenorientierten Prozess-Management-Systemen'. In: *EMISA Forum* 30.1 (Feb. 2010), pp. 11–28.

[252] V. Künzle and M. Reichert. 'PHILharmonicFlows: towards a framework for object-aware process management'. In: *Journal of Software Maintenance and Evolution: Research and Practice* 23.4 (2011), pp. 205–244.

[253] Jianrui Wang. 'PiDES: a formalism for complex adaptive discrete event simulation'. PhD thesis. Pennsylvania State University, USA, 2009.

[254] A. Nigam and N.S. Caswell. 'Business artifacts: An approach to operational specification'. In: *IBM Systems Journal* 42.3 (2003), pp. 428–445.

[255] C. E. Gerede, K. Bhattacharya, and J. Su. 'Static analysis of business artifact-centric operational models'. In: *Proc of the IEEE Int'l Conf on Service-Oriented Computing and Applications (SOCA'07)*. June 2007, pp. 133–140.

[256] C. E. Gerede and J. Su. 'Specification and verification of artifact behaviors in business process models'. In: *Proc of the 5th Int'l Conf on Service-Oriented Computing (ICSOC'07)*. Sept. 2007, pp. 181–192.

[257] A. Deutsch, R. Hull, F. Patrizi, and V. Vianu. 'Automatic verification of data-centric business processes'. In: *Proc of the 12th Int'l Conf on Database Theory (ICDT'09)*. Mar. 2009, pp. 252–267.

[258] C. Fritz, R. Hull, and J. Su. 'Automatic construction of simple artifact-based business processes'. In: *Proc of the 12th Int'l Conf on Database Theory (ICDT'09)*. Mar. 2009, pp. 225–238.

[259] R. Hull, E. Damaggio, F. Fournier, M. Gupta, F. Heath, S. Hobson, M. Linehan, S. Maradugu, A. Nigam, P. Sukaviriya, et al. 'Introducing the guard-stage-milestone approach for specifying business entity lifecycles'. In: *Web Services and Formal Methods* (2011), pp. 1–24.

[260] Vera Künzle and Manfred Reichert. *PHILharmonicFlows: Research and Design Methodology*. Tech. rep. University of Ulm, Oct. 2011.

[261] Vera Künzle and Manfred Reichert. 'Integrating Users in Object-Aware Process Management Systems: Issues and Challenges'. In: *Business Process Management Workshops 2009*. Ed. by Stefanie Rinderle-Ma, Shazia Sadiq, Frank Leymann, Wil Aalst, John Mylopoulos, Michael Rosemann, Michael J. Shaw, and Clemens Szyperski. Vol. 43. Lecture Notes in Business Information Processing. Springer, 2010, pp. 29–41.

[262] Juliane Blechinger, Frank Lauterwald, and Richard Lenz. 'Metadata Categories for Supporting Concurrent Engineering'. In: *Proc of the 15th IEEE International Enterprise Distributed Object Computing Conference Workshops (EDOCW'11)*. Sept. 2011, pp. 26–33.

[263] Juliane Blechinger. 'Ein Metadatenrepositorium zum Datenqualitätsmonitoring im Concurrent Engineering'. PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany, 2012.

[264] B. Karbe, N. Ramsperger, and P. Weiss. 'Support of cooperative work by electronic circulation folders'. In: *Proc of the ACM SIGOIS and IEEE CS TC-OA Conf on Office Information Systems (COIS'90)*. Apr. 1990, pp. 109–117.

[265] B. Karbe, N. Ramsperger, and P. Vogel. 'Office Work Coordination Using a Distributed Database System'. In: *Proc of the 2nd Int'l Conf on Database Systems for Advanced Applications (DASFAA'91)*. Apr. 1991, pp. 439–443.

[266] K. Klöckner, P. Mambrey, M. Sohlenkamp, W. Prinz, L. Fuchs, S. Kolvenbach, U. Pankoke-Babatz, and A. Syri. 'POLITeam Bridging the Gap between Bonn and Berlin for and with the Users'. In: *Proc of the 4th European Conference on Computer-Supported Cooperative Work (ECSCW'95)*. Sept. 1995, pp. 17–32.

[267] M. Sohlenkamp, L. Fuchs, and A. Genau. 'Awareness and cooperative work: The PoliTeam approach'. In: *Proc of the 30th Hawaii Int'l Conf on System Sciences (HICSS-30)*. Jan. 1997, pp. 549–558.

[268] L. Fuchs, M. Sohlenkamp, A. Genau, H. Kahler, A. Pfeifer, and V. Wulf. 'Transparenz in kooperativen Prozessen: Der Ereignisdienst in POLITeam'. In: *Proc of German Conf Deutsche Computer Supported Cooperative Work (DCSCW'96)* (1996), pp. 3–16.

[269] U. Pankoke-Babatz, G. Mark, and K. Klöckner. 'Design in the PoliTeam project: evaluating user needs in real work practice'. In: *Proc of the 2nd Conf on Designing Interactive Systems: Processes, Practices, Methods, & Techniques (DIS'97)*. Aug. 1997, pp. 277–287.

[270] A. B. Cremers, H. Kahler, A. Pfeifer, O. Stiemerling, and V. Wulf. 'PoliTeam–Kokonstruktive und evolutionäre Entwicklung einer Groupware'. In: *Informatik-Spektrum* 21.4 (1998), pp. 194–202.

[271] M. Sohlenkamp, P. Mambrey, W. Prinz, L. Fuchs, A. Syri, U. Pankoke-Babatz, K. Kloeckner, and S. Kolvenbach. 'Supporting the distributed German government with POLITeam'. In: *Multimedia Tools and Applications* 12.1 (2000), pp. 39–58.

[272] Volker Wulf. 'Handling Conflicts in Groupware: Concepts and Experiences made in the POLITeam-Project'. In: *Proc of the 6th Int'l Conf on Human-Computer Interaction (INTERACT'97)*. Vol. 97. July 1997.

[273] A. LaMarca, W. K. Edwards, P. Dourish, J. Lamping, I. Smith, and J. Thornton. 'Taking the work out of workflow: mechanisms for document-centered collaboration'. In: *Proc of the 6th European Conf on Computer Supported Cooperative Work (ECSCW'99)*. Sept. 1999, pp. 1–20.

[274] Adam Goldstein. *AppleScript: the missing manual*. O'Reilly, 2005. ISBN: 0596008503.

[275] Microsoft Corporation. *Active Document Containment*. `http://msdn.microsoft.com/en-us/library/6bzz39ft(v=vs.71).aspx`. 2003.

[276] Microsoft Corporation. *Active Document Containers*. `http://preview.library.microsoft.com/de-de/library/644x1yy6(v=vs.80).aspx`. 2005.

[277] Microsoft Corporation. *Active Documents on the Internet*. `http://preview.library.microsoft.com/de-de/library/99azcxx1(v=vs.80).aspx`. 2005.

[278] Microsoft Corporation. *Programmatic Printing*. `http://preview.library.microsoft.com/de-de/library/acax0dwh(v=vs.80).aspx`. 2005.

[279] Philip Barker. 'Using Wikis and Weblogs to Enhance Human Performance'. In: *Proc of the World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education (ELEARN'08)*. Nov. 2008, pp. 581–589.

[280] Oliver Imbusch, Falk Langhammer, and Guido von Walter. 'Ercatons: Thing-Oriented Programming'. In: *Proc of the 5th Int'l Conf on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World (Net.ObjectDays 2004)*. 2004, pp. 216–238.

[281] Oliver Imbusch, Falk Langhammer, and Guido von Walter. 'Ercatons and organic programming: say good-bye to planned economy'. In: *Proc of the 20th Int'l Conf on Object-oriented Programming, Systems, Languages, and Applications (OOPSLA'05)*. Oct. 2005, pp. 41–52.

[282] S. Abiteboul, O. Benjellourn, I. Manolescu, T. Milo, and R. Weber. 'Active XML: Peer-to-peer data and web services integration'. In: *Proc of the 28th Int'l Conf on Very Large Data Bases (VLDB'02)*. Aug. 2002, pp. 1087–1090.

[283] S. Abiteboul, O. Benjelloun, and T. Milo. 'The Active XML project: an overview'. In: *The VLDB Journal* 17.5 (2008), pp. 1019–1040.

[284] W. S. Brainerd and L. H. Landweber. *Theory of computation*. John Wiley & Sons, 1974. ISBN: 0471095850.

[285] Kyle Jones. *VM User's Manual - Virtual Folders*. `http://www.wonderworks.com/vm/user-manual/vm_15.html`. 1991.

[286] Dave Caolo. *Using smart folders in Apple's Mail*. `http://www.tuaw.com/2005/06/14/using-smart-folders-in-apples-mail/`. June 2005.

[287] Apple Inc. *Folder Actions Reference*. `https://developer.apple.com/library/mac/documentation/applescript/Conceptual/ApplescriptLangGuide/reference/ASLR_folder_actions.html`. Mar. 2008.

[288] Cory Bohon. *AppleScript: Exploring the power of Folder Actions, part I*. `http://www.tuaw.com/2009/02/16/applescript-exploring-the-power-of-folder-actions-part-i/`. Feb. 2009.

[289] William R. Cook. 'AppleScript'. In: *Proc of the third ACM SIGPLAN Conf on History of programming languages*. 2007, pp. 1–12.

[290] WUGNET. *Office Binder*. `http://www.wugnet.com/tips/display.asp?ID=743`. Nov. 1998.

[291] Microsoft Corporation. *Active Documents*. `http://msdn.microsoft.com/en-us/library/aa269033(v=vs.60).aspx`. 1998.

[292] M. Williams and D. Bennett. *Visual C++ 6 Unleashed*. Sams, 2000.

[293] W. Jones. 'Personal information management'. In: *Annual Review of Information Science and Technology (ARIST)* 41.1 (2007), pp. 453–504.

[294] David Allen. *Getting Things Done: The art of stress-free productivity*. Penguin Group USA, 2001. ISBN: 0142000280.

[295] San-Yih Hwang und Ya-Fan Chen. 'Personal workflows: Modeling and management'. In: *Proc of the 4th Int'l Conf on Mobile Data Management (MDM'03)*. Jan. 2003, pp. 141–152.

[296] B. A. Tate and C. Hibbs. *Ruby on Rails: Up and Running*. O'Reilly, 2006. ISBN: 0596101325.

[297] R. Pawson and V. Wade. 'Agile development using naked objects'. In: *Proc on 4th Int'l Conf on eXtreme Programming and Agile Processes in Software Engineering (XP 2003)*. May 2003, pp. 1010–1010.

[298] Richard Pawson. 'Naked Objects'. PhD thesis. Trinity College, Dublin, Ireland, 2004.

[299] B. Xu, W. Lian, and Q. Gao. 'Migration of Enterprise JavaBeans with ProActive interposition objects'. In: *SIGPLAN Notices* 38.8 (2003), pp. 22–28.

[300] T. Illmann, F. Kargl, M. Weber, and T. Kruger. 'Migration of mobile agents in Java: Problems, classification and solutions'. In: Dec. 2000.

[301] Manuela Schinn. 'Entwurf und Realisierung eines Webportals zum Zugriff auf Patientenleitfäden mit Hilfe von Ercatons'. Studienarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Jan. 2010.

[302] Roland Berger. 'Telematik im Gesundheitswesen – Perspektiven der Telemedizin in Deutschland'. In: *Studie im Auftrag des Bundesministeriums für Bildung, Wissenschaft, Forschung und Technologie in Zusammenarbeit mit dem Bundesministerium für Gesundheit, München* (1997).

[303] P. T. Eugster, P. A. Felber, R. Guerraoui, and A. M. Kermarrec. 'The many faces of publish/subscribe'. In: *ACM Computing Surveys (CSUR)* 35.2 (2003), pp. 114–131.

[304] Annika Hinze, Kai Sachs, and Alejandro P. Buchmann. 'Event-based applications and enabling technologies'. In: *Proc of the 3rd ACM Int'l Conf on Distributed Event-Based Systems (DEBS'09)*. July 2009.

[305] Christoph P. Neumann and Richard Lenz. 'The alpha-Flow Use-Case of Breast Cancer Treatment – Modeling Inter-Institutional Healthcare Workflows by Active Documents'. In: *Proc of the 8th Int'l Workshop on Agent-based Computing for Enterprise Collaboration (ACEC) at the 19th Int'l Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2010)*. Larissa, GR, June 2010.

[306] Hans-Ulrich Prokosch. 'Prozessoptimierung durch moderne Krankenhaus-Informations- und Workflowsysteme'. In: *eHealth: Innovations- und Wachstumsmotor für Europa*. Ed. by Jörg Eberspächer, Arnold Picot, and Günter Braun. Springer, 2006, pp. 221–238.

[307] T. Kirsche, R. Lenz, T. Ruf, and H. Wedekind. 'Cooperative problem solving using database conversations'. In: *Proc of the 10th IEEE Int'l Conf on Data Engineering (ICDE'94)*. Feb. 1994, pp. 134–143.

[308] Thomas Kirsche. 'Datenbankkonversationen'. PhD thesis. Friedrich-Alexander-Universität Erlangen-Nürnberg, 1995.

[309] Christoph P. Neumann, Peter K. Schwab, Andreas M. Wahl, and Richard Lenz. 'alpha-Adaptive: Evolutionary Workflow Metadata in Distributed Document-Oriented Process Management'. In: *Proc of the 4th Int'l Workshop on Process-oriented Information Systems in Healthcare (ProHealth'11) in conjunction with the 9th Int'l Conf on Business Process Management (BPM'11)*. Clermont-Ferrand, FR, Aug. 2011.

[310] N. Freed and N. Borenstein. *(RFC 2046:) Multipurpose Internet Mail Extensions (MIME) part two: Media types*. Tech. rep. Internet Engineering Task Force (IETF), Nov. 1996.

[311] Aneliya Todorova and Christoph P. Neumann. 'alpha-Props: A Rule-Based Approach to 'Active Properties' for Document-Oriented Process Support in Inter-Institutional Environments'. In: *Lecture Notes in Informatics (LNI) Seminars 10 / Informatiktage 2011*. Ed. by Ludger Porada. Gesellschaft für Informatik e.V. (GI). Mar. 2011.

[312] Stefan Hanisch. 'Konzeption und Implementierung einer Infrastruktur für aktive Dokumente'. Diplomarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Oct. 2010.

[313] Lakshmi Shankar and Simon Burns. *Demystifying class loading problems, Part 1: An introduction to class loading and debugging tools*. `http://www.ibm.com/developerworks/java/library/j-dclp1/`. Nov. 2005.

[314] T. Friese, M. Smith, and B. Freisleben. 'Hot service deployment in an ad hoc grid environment'. In: *Proceedings of the 2nd international conference on Service oriented computing*. ACM. 2004, pp. 75–83.

[315] John Mazz. *Classloaders Keeping Jar Files Open*. `http://management-platform.blogspot.de/2009/01/classloaders-keeping-jar-files-open.html`. Jan. 2009.

[316] Florian Wagner. 'alpha-Forms: Selbst-editierbare Formulare als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten'. Diplomarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Dec. 2011.

[317] Ian Hickson. *HTML5 - A vocabulary and associated APIs for HTML and XHTML*. Tech. rep. `http://www.w3.org/TR/html5/`. World Wide Web Consortium (W3C), May 2011.

[318] John M. Boyer. *XForms Version 1.1*. Tech. rep. http://www.w3.org/TR/2009/REC-xforms-20091020/. World Wide Web Consortium (W3C), Oct. 2009.

[319] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison-Wesley, 1994. ISBN: 0201633612.

[320] Patrick Reischl. 'alpha-Templates: Import und Export von 'Process Templates' als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten'. Bachelorarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Oct. 2011.

[321] Anelyia Todorova. 'Design and Implementation of a Lightweight, Autonomous, Rule-Based System Which Realizes 'Active Properties' in the Context of Active Documents'. Diplomarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, July 2010.

[322] Klaus R. Dittrich, Angelika M. Kotz, and Jutta A. Mülle. 'An event/trigger mechanism to enforce complex consistency constraints in design databases'. In: *SIGMOD Record* 15.3 (1986), pp. 22–36.

[323] Dennis McCarthy and Umeshwar Dayal. 'The architecture of an active database management system'. In: *Proc of the 1989 ACM SIGMOD Int'l Conf on Management of Data (SIGMOD'89)*. ACM, 1989, pp. 215–224.

[324] U. Dayal, A. Buchmann, and D. McCarthy. 'Rules are objects too: a knowledge model for an active, object-oriented database system'. In: *Advances in Object-Oriented Database Systems* (1988), pp. 129–143.

[325] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The description logic handbook: theory, implementation, and applications*. 2nd ed. Cambridge University Press, May 2010. ISBN: 0521150116.

[326] Peter Jackson. *Introduction to Expert Systems*. Addison-Wesley, 1998. ISBN: 0201876868.

[327] H. Chen, T. Finin, and A. Joshi. 'An ontology for context-aware pervasive computing environments'. In: *The Knowledge Engineering Review* 18.03 (2003), pp. 197–207.

[328] F. Hayes-Roth. 'Rule-based systems'. In: *Communications of the ACM* 28.9 (1985), pp. 921–932.

[329] JBoss Drools team. *Drools Expert User Guide, Version 5.2.0.Final*. http://docs.jboss.org/drools/release/5.2.0.Final/drools-expert-docs/html_single/. June 2011.

[330] Peter Schwab. 'alpha-Adaptive: Ein adaptives Attributmodell als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten'. Diplomarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, June 2011.

[331] P. M. Nadkarni. 'Data Extraction and Ad Hoc Query of an Entity-Attribute-Value Database'. In: *Journal of the American Medical Informatics Association (JAMIA)* 5.6 (1998), p. 511.

[332] P. H. Winston and R. H. Brown. *Artificial intelligence: an MIT perspective*. MIT Press, 1979. ISBN: 0262230968.

[333] Christian Hunsen. 'alpha-Doyen: Ein Verfahren zur Wortführerschaft-Übertragung als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten'. Masterarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, July 2012.

[334] Andreas M. Wahl. 'alpha-OffSync: Verteilten Datensynchronisation in Form von IMAP-basiertem Mail-Transfer als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten'. Bachelorarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Oct. 2011.

[335] Christoph P. Neumann, Andreas M. Wahl, and Richard Lenz. 'Adaptive Version Clocks and the OffSync Protocol'. In: *Proc of the 10th IEEE Int'l Symposium on Parallel and Distributed Processing with Applications (ISPA-12)*. Madrid, Spain, July 2012.

[336] Richard Lenz. 'Adaptive Datenreplikation in verteilten Systemen'. PhD thesis. Stuttgart: Friedrich-Alexander-Universität Erlangen-Nürnberg, 1997.

[337] C. J. Fidge. 'Timestamps in message-passing systems that preserve the partial ordering'. In: *Australian Computer Science Communications* 10.1 (Feb. 1988), pp. 56–66.

[338] R. Schwarz and F. Mattern. 'Detecting causal relationships in distributed computations: In search of the holy grail'. In: *Distributed computing* 7.3 (1994), pp. 149–174.

[339] D. S. Parker Jr, G. J. Popek, G. Rudisin, A. Stoughton, B. J. Walker, E. Walton, J. M. Chow, D. Edwards, S. Kiser, and C. Kline. 'Detection of mutual inconsistency in distributed systems'. In: *IEEE TSE* 9.3 (1983), pp. 240–247.

[340] S. Ceri, M.A.W. Houtsma, A.M. Keller, and P. Samarati. 'The Case for Independent Updates'. In: *Proc of the 2nd IEEE Workshop on the Management of Replicated Data*. Nov. 1992, pp. 17–19.

[341] R. A. Golding. 'Weak-consistency group communication and membership'. PhD thesis. Santa Cruz: University of California, USA, 1992.

[342] Leslie Lamport. 'Time, clocks, and the ordering of events in a distributed system'. In: *Communications of the ACM* 21.7 (1978), pp. 558–565.

[343] Scott Hady. 'alpha-VVS: An integrated Version Control System as a Component of Process Support based on Active Documents'. Diplomarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Nov. 2011.

[344] Christoph P. Neumann, Scott A. Hady, and Richard Lenz. 'Hydra Version Control System'. In: *Proc of the 10th IEEE Int'l Symposium on Parallel and Distributed Processing with Applications (ISPA-12)*. Madrid, Spain, July 2012.

[345] Jon Loeliger. *Version Control with Git*. O'Reilly, 2009.

[346] Patrick Mukherjee. 'A Fully Decentralized, Peer-to-Peer Version Control System'. PhD thesis. Technische Universität Darmstadt, 2005.

[347] Marc J. Rochkind. 'The source code control system'. In: *IEEE Transactions on Software Engineering* 1 (4 1975), pp. 364–470.

[348] Walter F. Tichy. 'RCS – A System for Version Control'. In: *Software Practice and Experience* 15.7 (July 1985), pp. 637–654.

[349] Hong Gao and Jie Tan. *Multiple Payments at One Click*. `http://www.google.com/patents?id=B0H6AAAAEBAJ`. US Patent No.: 2011/0022516 A1. July 2009.

[350] Konstantin Tsysin. 'alpha-PrintPut: Ein Windows-Druckertreiber zum Einbringen von Dokumenten aus beliebigen Drittanwendungen als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten'. Bachelorarbeit. Friedrich-Alexander-Universität Erlangen-Nürnberg, Dec. 2011.

[351] Martin Fowler. *Inversion of Control Containers and the Dependency Injection Pattern*. `http://martinfowler.com/articles/injection.html`. Jan. 2004.

[352] Robert C. Martin. *The Dependency Inversion Principle*. `http://objectmentor.com/resources/articles/dip.pdf`. May 1996.

[353] Rod Johnson and Bob Lee. *(JSR 330:) Dependency Injection for Java*. Tech. rep. Java Community Process (JCP), Oct. 2009.

[354] Michele Lanza and Radu Marinescu. *Object-oriented metrics in practice: using software metrics to characterize, evaluate, and improve the design of object-oriented systems*. Springer, 2006. ISBN: 3540244298.

[355] A. H. Watson, T. J. McCabe, and D. R. Wallace. 'Structured testing: A testing methodology using the cyclomatic complexity metric'. In: *NIST special Publication* 500.235 (1996), pp. 1–114.

[356] C. Marinescu, R. Marinescu, P. F. Mihancea, and R. Wettel. 'iPlasma: An integrated platform for quality assessment of object-oriented design'. In: *Proc of the 21st IEEE Int'l Conf on Software Maintenance (ICSM'05), Industrial and Tool Volume*. Sept. 2005.

[357] B. W. Boehm. 'Software engineering economics'. In: *IEEE Transactions on Software Engineering (TSE)* 1 (1984), pp. 4–21.

[358] David A. Wheeler. *More than a gigabuck: Estimating GNU/Linux's size*. `http://www.dwheeler.com/sloc/redhat71-v1/redhat71sloc.1.03.html`. June 2001.

[359] R. Valerdi. 'Pioneers of parametrics'. In: *Proc on the 28th Conf of the Int'l Society of Parametric Analysts*. May 2007.

[360] Daniel D. Galorath and Michael W. Evans. *Software Sizing, Estimation, and Risk Management: When Performance Is Measured Performance Improves*. CRC Press, 2006. ISBN: 0849335930.

[361] R. A. Ghosh et al. *Economic impact of open source software on innovation and the competitiveness of the Information and Communication Technologies (ICT) sector in the EU*. `http://ec.europa.eu/enterprise/sectors/ict/files/2006-11-20-flossimpact_en.pdf`. A study commissioned by the European Commission's Directorate General for Enterprise and Industry. Nov. 2006.

[362] E. Raymond. 'The cathedral and the bazaar'. In: *Knowledge, Technology & Policy* 12.3 (1999), pp. 23–49.

[363] K. Haaland, I. Stamelos, R. Ghosh, and R. Glott. 'On the Approximation of the Substitution Costs for Free/Libre Open Source Software'. In: *Proc of the 4th Balkan Conference in Informatics (BCI'09)*. Sept. 2009, pp. 223–227.

[364] Ron Hitchens. *Java NIO*. O'Reilly, 2002. ISBN: 0596002882.

[365] S. Rönnau, J. Scheffczyk, and U. M. Borghoff. 'Towards XML version control of office documents'. In: *Proc of the ACM Symp on Document Engineering (DocEng 2005)*. Nov. 2005, pp. 10–19.

[366] Luuk Peters. 'Change Detection in XML Trees: a Survey'. In: *Proc of the 3rd Twente Student Conference on IT*. June 2005.

[367] Anish Karmarkar and Ümit Yalcinalp. *Describing Media Content of Binary Data in XML*. Tech. rep. `http://www.w3.org/TR/xml-media-types/`. World Wide Web Consortium (W3C), May 2005.

[368] Stefania Castellani and Francois Pacull. 'XFolders: A flexible workflow system based on electronic circulation folders'. In: *Proc of 2nd Int'l Workshop on Web Based Collaboration (WBC) in conjunction with the 13th Int'l Conf on Database and Expert Systems Applications (DEXA'02)*. Sept. 2002, pp. 307–312.

[369] Davide Rossi. 'Orchestrating document-based workflows with X-Folders'. In: *Proc of the ACM Symposium on Applied Computing (SAC'04)*. Mar. 2004, pp. 503–507.

[370] Davide Rossi. 'X-Folders: documents on the move'. In: *Concurrency and Computation: Practice and Experience* 18.4 (2005), pp. 409–425.

[371] Jean-marc Andreoli, Damián Arregui, François Pacull, Michel Rivière, Jean yves Vion-dury, and Jutta Willamowski. 'CLF/Mekano: a framework for building virtual-enterprise applications'. In: *Proc of 3rd Int'l Conf on Enterprise Distributed Object Computing (EDOC'99)*. Sept. 1999.

[372] Stephan Wilczek. 'Aktive elektronische Dokumente in Telekooperationsumgebungen: Konzept und Einsatzmöglichkeiten am Beispiel elektronischer Patientenakten'. PhD thesis. Universität Hohenheim, Germany, 2007.

[373] C. Crabtree, P. B. Howard, and P. El-Mallakh. 'The Care and Outcomes Management Plan and Kardex'. In: *Journal of Healthcare Information Management (JHIM)* 23 (1 2009).

[374] S. Josefsson. *(RFC 4648:) The Base16, Base32, and Base64 Data Encodings*. Tech. rep. Internet Engineering Task Force (IETF), Oct. 2006.

[375] Qusay H. Mamoud. *Getting started with JavaSpaces technology: Beyond conventional distributed programming paradigms*. `http://java.sun.com/developer/technicalArticles/tools/JavaSpaces/`. July 2005.

[376] Michael Kay et al. *XSL Transformations (XSLT) Version 2.0*. Tech. rep. `http://www.w3.org/TR/2007/REC-xslt20-20070123/`. World Wide Web Consortium (W3C), Jan. 2007.

[377] M. Martin, S. Schick, T. Bürkle, S. Petsch, U. Altmann, M. Beckmann, and H.-U. Prokosch. 'Kann man ein Tumordokumentationssystem einfach austauschen? Erfahrungen aus einem Umstellungsprojekt'. In: *Tagungsband der 56. GMDS-Jahrestagung und 6. DGEpi-Jahrestagung*. Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS). Mainz, DE, Sept. 2011.

[378] S. Kirn, C. Anhalt, and C. Heine. 'Mobiles Computing in der Medizin'. In: *Tagungsband des 4. Workshop der GMDS-Projektgruppe Mobiles Computing in der Medizin (MoCoMed 2004)*. Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie (GMDS). Suttgart, DE, Apr. 2004.

Glossary

.NET
: Microsoft .NET Framework

ADEPT
: A workflow engine by the Database Group of the Department of Computer Science, Ulm University, Germany.

ADEPT$_{flex}$
: Originally, a defined set of change operations being supported by the ADEPT workflow engine as an extension. The term is often used synonymously to the extended workflow engine itself.

HYGEIAnet
: The integrated regional health information network of Crete.

IBM
: A U.S.-American company for technology and consulting. The company title is an abbreviation for "International Business Machines Corporation".

MSDN
: A centralized repository of official developer-related documentation as well as a forum and a blog for Microsoft employees. MSDN is an abbreviation for "Microsoft Developer Network".

OHF
: The Open Healthcare Framework, a collection of projects hosted by the Eclipse Foundation.

OpenPGP
: The OpenPGP specification is a Request for Comment (RFC) by the Internet Engineering Task Force (IETF). It is an open standard specification for email encryption based on the original concepts from Pretty Good Privacy (PGP).

RESTful
: A system or interface that is designed based on the principles of REST by Roy T. Fielding.

SAP

A German company for business applications. The company title is originally an abbreviation for the German sentence “Systeme, Anwendungen, Produkte in der Datenverarbeitung”, which translates to “systems, applications, products for data processing”.

SOAP

A medical term that stems form Lawrence Weed for his concept of problem-oriented medical records. In this context, SOAP is an abbreviation for Subjective, Objective, Assessment, and Plan. (Notably, the acronym SOAP can also appear in a technical context. In this case, it would stand for Simple Object Access Protocol, which is a protocol to provide or access web services.)

TNM

Classification of Malignant Tumors: Tumor, Lymph Nodes, Metastasis.

WS-*

The variety of specifications associated with web services.

openEHR

The openEHR foundation and its open standard specification for electronic health records.

List of Acronyms

ABD	"Arztbriefdienst"
ACM	Adaptive Case Management
AIIM	Association for Information and Image Management
AJAX	Asynchronous JavaScript and XML
AMDD	"Arzneimitteldokumentationsdienst"
AOX	"akteonline extendable"
APA	Adornment Prototype Artifact
ASTM	American Society for Testing and Materials
ATNA	Audit Trail and Node Authentication
AVC	Adaptive Vector Clock
AWMF	"Arbeitsgemeinschaft der Wissenschaftlichen Medizinischen Fachgesellschaften"
AXML	Active XML
B2B	Business-to-Business
BFS	Be File System
BI-RADS	Breast Imaging – Reporting and Data System
BPEL	Business Process Execution Language
BPMI	Business Process Management Initiative
BPMN	Business Process Model and Notation
BPM	Business Process Modelling
BeOS	Be Operating System
CAD	Computer-Aided Design
CCD	Continuity of Care Document
CCOW	Clinical Context Object Workgroup
CCR	Continuity of Care Records
CDA	Clinical Document Architecture
CDSS	Clinical Decision Support System
CMS	Content Management System
COCOMO	COnstructive COst MOdel
COM	Component Object Model

CPU	Central Processing Unit
CRA	Collaboration Resource Artifact
CSS	Cascading Style Sheets
CVS	Concurrent Versions System
DBMS	Database Management System
DCOM	Distributed Component Object Model
DC	Digital Card
DEC	Digital Equipment Corporation
DEUS	Distributed Electronic Patient File Update System
DFD	Data-Flow Diagram
DICOM	Digital Imaging and Communications in Medicine
DMA	Document Management Alliance
DMPS	Distributed Medical Process Support
DOM	Document Object Model
DSD	Deferred System Design
DSL	Domain Specific Language
DSSP	Dataspace Support Platform
DSSSL	Document Style Semantics and Specification Language
DSS	Decision Support System
EAI	Enterprise Application Integration
EAV	Entity-Attribute-Value
EBM	Evidence-Based Medicine
ECA	Event-Condition-Action
ECF	Electronic Circulation Folder
EHR	Electronical Health Record
EJB	Enterprise JavaBean
EMR	Electronic Medical Record
EPAD	“Elektronischer Patientenaktendienst”
EPC	Event-driven Process Chain
EPK	“Ereignisgesteuerte Prozesskette”
GMDS	“Deutsche Gesellschaft für Medizinische Informatik, Biometrie und Epidemiologie”
GSM	Guard-Stage-Milestone
GTD	Getting Things Done
GUI	Graphical User Interface

HCIS	Healthcare Information System
HIS	Hospital Information System
HL7	Health Level 7
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
ICD	International Statistical Classification of Diseases and Related Health Problems
IDL	Interface Description Language
IEEE	Institute of Electrical and Electronics Engineers
IFD	Information Flow Diagram
IHE	Integrating the Healthcare Enterprise
IMAP	Internet Message Access Protocol
INRIA	Institut National de Recherche en Informatique et en Automatique
IOM	Institute of Medicine
IPR	Interleaved Parallel Routing
JAR	Java Archive
JCR	Java Content Repository
JEE	Java Platform, Enterprise Edition
JVM	Java Virtual Machine
LIMS	Laboratory Information Management System
LOINC	Logical Observation Identifiers Names and Codes
MPI	Master Patient Index
MRI	Magnetic Resonance Imaging
NFDD	"Notfalldatendienst"
NFS	Network File System
NIST	National Institute of Standards and Technology
OC	Object under Consideration
OLE	Object Linking and Embedding
OMG	Object Management Group
OSGi	Open Services Gateway Initiative
OWL-DL	OWL Description Language
OXDBS	Ontological XML Database System
PARC	Palo Alto Research Center
PCC	Patient Care Coordination
PDA	Personal Digital Assistant

PDD	"Patientendatendienst"
PDF	Portable Document Format
PDS	Personal Data Store
PHR	Personal Health Record
PIDS	Person Identification Service
PIM	Personal Information Management
PIM tool	Personal Information Manager
PIX	Patient Identifier Cross-referencing
PKI	Public Key Infrastructure
POMR	Problem-Oriented Medical Record
PRICE-S	Parametric Review of Information for Costing and Evaluation – Software
PSA	Process Structure Artifact
PhD	Doctor of Philosophy
RAD	Rapid Application Development
RCS	Revision Control System
REST	Representational State Transfer
RHIN	Regional Healthcare Information Network
RIM	Reference Information Model
RIS	Radiology Information System
RLM	Release Management Workflow
SBCE	Set-Based Concurrent Engineering
SCCS	Source Code Control System
SCIPHOX	Standardized Communication of Information Systems in Physician Offices and Hospitals using XML
SCM	Supply Chain Management
SECI	Socialization, Externalization, Combination, Internalization
SEER-SEM	Software Evaluation and Estimation of Resources – Software Estimating Model
SGML	Standard Generalized Markup Language
SKOS	Simple Knowledge Organization System
SMTP	Simple Mail Transfer Protocol
SNOMED	Systematized Nomenclature of Medicine
SOAP	Simple Object Access Protocol
SOMR	Source-Oriented Medical Record

SSM	Soft Systems Methodology
SVN	Subversion
TID	Tuple Identifier
TOMR	Time-Oriented Medical Record
TSD	trip status document
UAC	User Account Control
UML	Unified Modeling Language
UUID	Universally Unique IDentifier
U.S.	United States
VCS	Version Control System
VODD	"Verordnungsdatendienst"
VRM	Vendor Relationship Management
VSDD	"Versichertenstammdatendienst"
W3C	World Wide Web Consortium
WPDL	Workflow Process Definition Language
WSDL	Web Services Description Language
WS-BPEL	Web Services Business Process Execution Language
WWW	World-Wide Web
WebDAV	Web-based Distributed Authoring and Versioning
WfMC	Workflow Management Coalition
WfMS	Workflow Management System
XACML	eXtensible Access Control Markup Language
XDI	XRI Data Interchange
XDS	Cross Enterprise Document Sharing
XML	Extended Markup Language
XMPP	Extensible Messaging and Presence Protocol
XPDL	XML Process Definition Language
XPHR	Exchange of Personal Health Record Content
XPath	XML Path Language
XQuery	XML Query Language
XRI	Extensible Resource Identifier
XSLT	Extensible Stylesheet Language Transformations
YAWL	Yet Another Workflow Language
dDPM	distributed Document-oriented Process Management
dVCS	distributed Version Control System

eGK	“Elektronische Gesundheitskarte”
ebXML	Electronic Business using eXtensible Markup Language

List of Symbols

α-Adaptive	A subsystem component of the *α-Flow* engine: provides run-time adaptiveness for the *α-Adornment* model
α-Adornment	A process relevant attribute that belongs to an *α-Card*; the general term "adornment" is borrowed from the Unified Modeling Language: an adornment adds to the meaning and/or semantics of the element to which it pertains and has a textual or graphical representation
α-Card	A fragment document of the distributed case file
α-Doc	A distributed case file in form of an active ($\cong$ "α") document
α-Doyen	A subsystem component of the *α-Flow* engine: provides process role labels like process coordinator
α-Editor	A subsystem component of the *α-Flow* engine: an embedded viewer and editor
α-Episode	Essentially a case, considered from an inter-institutional workflow perspective; one *α-Episode* is represented by one *α-Doc*
α-Flow	The overall approach name; the active part of an *α-Doc* is called the *α-Flow engine*
α-Forms	A subsystem component of the *α-Flow* engine: a combined form composer and form editor
α-Injector	A subsystem component of the *α-Flow* engine: provides drag-and-drop functionality (drag-and-drop contribution) and handles the initial creation of an *α-Doc* case file (the "alph-o-matic injection" that transforms a passive document into an active document)

α-*Kernel*	A subsystem component of the α-*Flow* engine: basically a rule engine; for example, it monitors α-*Adornment* state changes
α-*OffSync*	A subsystem component of the α-*Flow* engine and an implementation of the α-*OverNet*: uses SMTP&IMAP for data transfer as well as GnuPG for encryption
α-*OverNet*	A subsystem component of the α-*Flow* engine: an overlay network
α-*PrintPut*	A MS Windows printer-driver to ease the contribution of content documents into an α-*Doc* case file
α-*Startup*	A subsystem component of the α-*Flow* engine: provides the command-line interface and initializes the other subsystems
α-*Templates*	A subsystem component of the α-*Flow* engine: provides import and export of process templates
α-*VVS*	A subsystem component of the α-*Flow* engine: an embedded version control system

List of Figures

List of Tables