α-Doyen: Ein Verfahren zur Wortführerschaft-Übertragung als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten

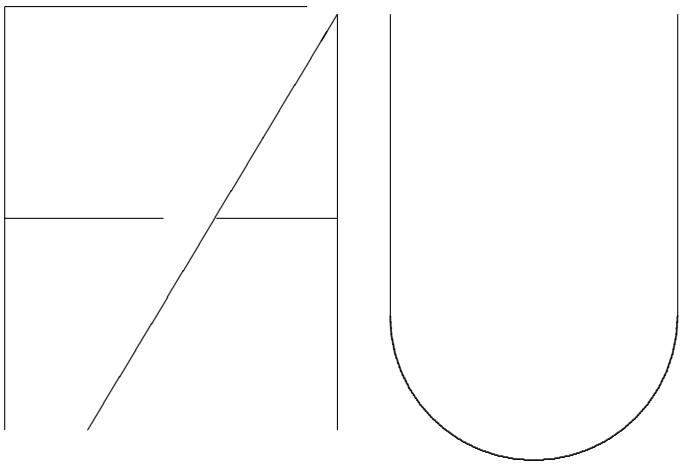
Masterarbeit

Christian Hunsen

Lehrstuhl für Informatik 6
(Datenmanagement)

Department Informatik
Technische Fakultät

Friedrich Alexander-
Universität
Erlangen-Nürnberg



α-Doyen: Ein Verfahren zur Wortführerschaft-Übertragung als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten

Masterarbeit im Fach Informatik

vorgelegt von

Christian Hunsen

geb. 01.09.1985 in Salzgitter-Bad

angefertigt am

Department Informatik
Lehrstuhl für Informatik 6 (Datenmanagement)
Friedrich-Alexander-Universität Erlangen-Nürnberg

Betreuer: Univ.-Prof. Dr.-Ing. habil. Richard Lenz
Dipl.-Inf. Christoph P. Neumann

Beginn der Arbeit: 01.02.2012
Abgabe der Arbeit: 31.07.2012

Erklärung zur Selbständigkeit

Ich versichere, dass ich die Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe und dass diese Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Der Universität Erlangen-Nürnberg, vertreten durch den Lehrstuhl für Informatik 6 (Datenmanagement), wird für Zwecke der Forschung und Lehre ein einfaches, kostenloses, zeitlich und örtlich unbeschränktes Nutzungsrecht an den Arbeitsergebnissen der Masterarbeit einschließlich etwaiger Schutzrechte und Urheberrechte eingeräumt.

Erlangen, den 31.07.2012

_____

(Christian Hunsen)

Kurzfassung

α-Doyen: Ein Verfahren zur Wortführerschaft-Übertragung als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten

Das Forschungsprojekt α-Flow stellt eine elektronische Prozessunterstützung für heterogene interinstitutionelle Szenarien im Gesundheitswesen, an denen viele unterschiedliche Akteure beteiligt sind, zur Verfügung. Dabei greift es auf ein dokumentenorientiertes Interaktionsparadigma zurück, das auf aktiven Dokumenten basiert. Die im Behandlungsverlauf eines Patienten anfallenden Informationen werden in Dokumenten abgelegt, die neben den Nutzdaten auch für den Workflow relevante Metadaten enthalten. Gebündelt werden die Dokumente in einer elektronischen Fallakte, dem α-Doc, das den verteilten Workflow steuert und Änderungen an den Dokumenten unter allen Prozessteilnehmern synchronisiert. Alle Teilnehmer können damit jederzeit auf die aktuellsten Informationen zugreifen.

Die Prozessteilnehmer sind alle gleichberechtigt. Um die in der Praxis vorliegende Situation besser abzubilden und bestimmte Positionen im Prozess festzulegen, erweitert α-Doyen das Domänenmodell um Prozessrollenbezeichner. Ein teilnehmender Akteur wird durch diese näher beschrieben. Insbesondere der Prozessinitiator und der Wortführer werden dabei berücksichtigt. Im Rahmen dieser Arbeit wurde ein Modell entwickelt, dass eine Weitergabe dieser Rollen durch Token ermöglicht, sodass sich das Prozessmodell im zeitlichen Verlauf des Prozesses anpassen kann.

Außerdem hilft die Erweiterung des Kommunikationsschemas um Empfangsbestätigungen von getätigten Änderungen dabei, die Verlässlichkeit bei der Synchronisierung von Informationen zu erhöhen. Rückmeldungen über Nachrichtenflüsse geben Auskunft darüber, ob und wann Änderungen beim Gegenüber angekommen sind und von diesem registriert wurden.

Abstract

α-Doyen: A method for propagating leadership as a module of a process support based on active documents

The research project α-Flow provides an electronic way to support processes in heterogenious inter-institutional scenarios in healthcare, where many parties are involved, using a document-centered approach based on active documents. The information that evolves during the course of treatment is stored in documents, which contain relevant metadata for the workflow in addition to the medical contents. The documents are bundled in an electronic case file, the α-Doc, which controls the distributed workflow and synchronizes changes of the documents with all process participants. All participants can use it at any time to access the latest available information.

All participants in the process have equal rights. To better reflect the real world situation and to define certain positions within the process, α-Doyen extends the domain model by process role labels. A participant is described in more detail by the latter. In particular, the process initiator and the spokesman are taken into account. This thesis has elaborated a model that enables the propagation of these roles through tokens, so that the process model can adapt itself during the progression of the process.

In addition the extension of the communication scheme by acknowledgements for changes made helps to increase the reliability during the synchronisation of information. Feedback about message flows provides information about whether and when changes have been received by other participants.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

APA	Adornment Prototype Artifact
ARA	Acknowledgement Repository Artifact
ASCII	American Standard Code for Information Interchange
AVC	Adaptive Version Clock
BI-RADS	Breast Imaging - Reporting and Data System
BPMN	Business Process Model and Notation
CDA	Clinical Document Architecture
CRA	Collaboration Resource Artifact
dDPM	verteiltes dokumentenorientiertes Prozessmanagementsystem
DOC	Dateiformat des Textverarbeitungsprogramms Microsoft Word
DOCX	Dateiformat des Textverarbeitungsprogramms Microsoft Word im „Office Open XML“-Format
DSN	Delivery Status Notification
EAV	Entity-Attribute-Value
HL7	Health Level 7
HTML	Hypertext Markup Language
IP	Internet Protocol
JPEG	Joint Photographic Experts Group
MDA	Mail Delivery Agent

MDN	Message Disposition Notification
MIME	Multipurpose Internet Mail Extensions
MTA	Mail Transfer Agent
OMG	Object Management Group
PDF	Portable Document Format
PGP	Pretty Good Privacy
POP3	Post Office Protocol Version 3
PSA	Process Structure Artifact
RBAC	Role-based Access Control
RFNM	Request For Next Message
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol
TNM	TNM Classification of Malignant Tumours
TPA	Token Prototype Artifact
UML	Unified Modelling Language
WfM	Workflow-Management
WfMC	Workflow Management Coalition
WfMS	Workflow-Management-System
XML	Extended Markup Language
XMPP	Extensible Messaging and Presence Protocol

1 Einleitung

Das deutsche Gesundheitswesen befindet sich im Wandel. Es ist eine Entwicklung zu erkennen, die die strikte Trennung von ambulanter, stationärer und rehabilitativer Versorgung auflöst und eine integrierte Versorgung über Institutions- und Disziplinsgrenzen hinweg anstrebt (cf. [HE10] und [Sch08a]). Der Patient wird im ambulanten Sektor von niedergelassenen Ärzten behandelt. Der im weiteren Behandlunsverlauf beteiligte stationäre Sektor umfasst vor allem Kliniken, Labore und Apotheken (cf. [NL12]). Das Zusammenspiel der an der Behandlung des Patienten beteiligten Akteure bedarf dabei vor allem an den Schnittstellen zwischen den Leistungserbringern besonderer Koordination, um einen effizienten Behandlungsverlauf zu gewährleisten. Dies ist insbesondere von Bedeutung, da das Gesundheitswesen immer komplexer wird und eine Spezialisierung auf Teilbereiche mit einer immer größeren Arbeitsteilung in der Behandlung eines Patienten einhergeht (cf. [Sch07]).

Die Kritik an der unzureichenden sektorübergreifenden Koordination und Information (cf. [NS05] und [Sch08b]) wird durch elektrische Fallakten aufgegriffen und es wird versucht, die Schwachstellen der Zusammenarbeit zu beheben. Elektronische Fallakten liefern eine strukturierte Sicht auf Dokumente, die zu einem Behandlungsfall eines Patienten erfasst sind. Die behandelnden Ärzte haben dadurch stets einen Überblick über den bisherigen Behandlungsverlauf und die aktuellsten Ergebnisse. Durch die Fallakte können sie sich auf elektronischem Weg schnell untereinander austauschen und jedem Arzt stehen jederzeit die aktuellsten Informationen zur Verfügung.

1.1 Motivation und Hintergründe

Das Projekt ProMed verfolgt das Ziel, adaptiv-evolutionäre Informationssysteme im Gesundheitssystem zu unterstützen. Ein wichtiges Teilprojekt von ProMed ist α-Flow, das auf der Basis von aktiven Dokumenten Prozessunterstützung in heterogenen interinstitutionellen Umgebungen geben soll. Innerhalb von Organisationseinheiten im Gesundheitssytem hat sich bereits häufig eine Verfahrensweise zum Informationsaus-

tausch entwickelt. Aber gerade die Kommunikation über Institutsgrenzen hinweg erfolgt oftmals noch unkoordiniert auf dem Papierweg.

α-Flow verfolgt das Ziel, die Zusammenarbeit von Akteuren institutsübergreifend zu unterstützen, ohne die bereits bestehenden Systeme zu integrieren. Um eine schnellere Rückkopplung, kurzfristige Befundübermittlungen und den gemeinsamen Zugriff auf alle Befunde und Untersuchungen zu ermöglichen, werden in α-Flow die anfallenden Dokumente in einer elektronischen Fallakte gesammelt und um aktive Eigenschaften erweitert. Die Fallakte lässt eine direkte Interaktion mit den Dokumenten zu und umfasst neben den medizinischen Dokumenten alle Informationen, die zum Mitwirken am Prozess benötigt werden.

α-Flow erleichtert den Informationsfluss, indem es automatisch die neuesten Behandlungsergebnisse an alle Teilnehmer propagiert. Ein papiergebundener, zeitintensiver Transport der Dokumente entfällt. Häufig auftretende Fehlbehandlungen aufgrund von nicht ausreichend zur Verfügung stehender Informationen können damit vermieden werden (cf. [NS05]).

Die Koordination der Ad-hoc-Zusammenarbeit der beteiligten Leistungserbringer entlang des Behandlungspfads eines Patienten mit dem Ziel, diesen bestmöglich zu heilen, kann durch einen „Koordinationsarzt“ erfolgen (cf. [Sch08b, Seite 152f.]). Er stimmt die Behandlung mit den anderen Ärzten (z.B. Fachärzten) ab, indem er Prozesse anstößt und Arbeitsanweisungen gibt. Im deutschen Gesundheitswesen nimmt diese Rolle häufig der Hausarzt ein, indem er als erster Ansprechpartner für den Patienten dient und den weiteren Behandlungsprozess abstimmt (cf. [SSD06] und [Köh06, Seite 37ff.]). Durch seinen Überblick über den Behandlungsverlauf kann er Zusammenhänge besser erkennen und geeignete Maßnahmen ergreifen. Zudem steht er allen Prozessbeteiligten als Ansprechpartner zur Verfügung.

Innerhalb der Patientenbehandlung können verschiedene Prozessrollen identifiziert werden. Zum einen ist der Prozessinitiator zu nennen, der die Fallakte erzeugt und die initialen Parameter wie den zu behandelnden Patienten festlegt (cf. auch [Böh09, Seite 72]). Seine Rolle wird beim Prozessstart festgelegt und bleibt ihm im Verlauf der Behandlung des Patienten zugeordnet.

Ein weitere Prozessrolle ist die des Wortführers, auch Doyen genannt. Das Doyen-Token zeichnet einen beteiligten Arzt auf fachlicher Ebene als Ansprechpartner aus. Zu Beginn des Prozesses sind die Initiatorrolle und die Doyenrolle demselben Akteur zugeordnet. Es wird sich aber zeigen, dass die Wortführerrolle weitergegeben werden

kann, während der Prozess fortschreitet. Das heißt, der für die Behandlung zu einem Zeitpunkt verantwortliche Arzt kann wechseln.

Eine weitere Eigenschaft, die ein am Workflow beteiligter Arzt haben kann, ist der Patientenkontakt. Es ist demnach denkbar, dass es Ärzte gibt, die keinen Patientenkontakt haben: beispielsweise Pathologen, die nur eine Gewebeprobe untersuchen. Diese Sonderrolle soll individuell gekennzeichnet und des Weiteren auch veränderbar sein. Eine Kennzeichnung könnte z.B. dann sinnvoll sein, wenn es darum geht, die Aktualisierungsnachrichten zu begrenzen oder eine wichtige Information zeitnah an den behandelten Patienten weiterzugeben.

Insbesondere bei für den weiteren Behandlungsablauf kritischen Informationen oder Befunden, kann es sinnvoll sein, darüber in Kenntnis gesetzt zu werden, wann der empfangende Akteur sein α-Doc synchronisiert und damit die Aktualisierungen erhält. Ein solcher Hinweis kann in Form von Empfangsbestätigungen erfolgen. Eine übersichtliche Darstellung der Akteure in Bezug auf den Erhalt einer Änderung an einem Prozessartefakt kann also entscheidend in der weiteren Behandlung sein. Wenn ein ändernder Akteur wahrnimmt, dass ein bestimmter Prozessteilnehmer seine Änderung noch nicht erhalten hat, dann kann er beispielsweise auf einem anderen Kommunikationsweg bei diesem nachfragen und ihn ggf. über die neuen dringenden Informationen in Kenntnis setzen.

1.2 Zielsetzung der Arbeit

Das Ziel dieser Arbeit ist es, das Systemmodell des α-Flow-Projekts dahingehend zu erweitern, dass einzelne Prozessteilnehmer durch Prozessrollenbezeichner, sogenannte Token, ausgezeichnet werden können. Es sollen der Prozessinitiator, der aktuelle Wortführer des Behandlungsprozesses und der Patientenkontakt durch Token indiziert werden. Dabei muss es auch möglich sein, das einzelne Token von einem Teilnehmer zum nächsten übertragen werden können. Die zu entwickelnde Speicherstruktur soll insbesondere so strukturiert werden, dass sie leicht um weitere Token erweitert werden kann, sodass diese in der Zukunft zur Laufzeit adaptiv angepasst werden kann. Dazu wurde in erster Linie das Konzept von α-Adaptive betrachtet (cf. [Sch11] und [NSWL11]) und für die Umsetzung des Token-Modells angepasst und erweitert. α-Adaptive erweitert das Modell von α-Flow um vom Benutzer festgelegte adaptiv-evolutionäre Statusattribute, die zur Laufzeit angepasst werden können.

Für eine kausale Ordnung auf den Token wird ein Zeitstempel-Verfahren verwendet, das auf logischen Zeitstempeln basiert. Das Übertragungsprotokoll für die Weitergabe der

Token muss die Konsistenz der Informationen gewährleisten. Es muss in der Lage sein, aufgrund von lokalen Informationen Aussagen über etwaige Änderungskonflikte treffen und diese beheben zu können. Im Rahmen einer prototypischen Implementierung soll das entwickelte Konzept umgesetzt werden und durch ein geeignetes Anzeige-Bedien-Konzept auch die geplante Adaptivität unterstützen.

Neben dieser Problemstellung soll diese Arbeit auch die Verlässlichkeit während der Verbreitung von Änderungsinformationen zwischen den am Behandlungsprozess beteiligten Akteuren durch Empfangsbestätigungen verbessern. Derjenige, der eine Änderung propagiert, will darüber informiert werden, welcher der anderen Teilnehmer seine Änderung bereits erhalten hat. Durch diese Information kann ein Aktor beispielsweise bei einer ausbleibenden Empfangsbestätigung für die von ihm durchgeführte Änderung beim anderen Akteur nachfragen. In der Oberfläche soll der Status der eingetroffenen Empfangsbestätigungen für eine α-Card-Änderung übersichtlich dargestellt werden können.

2 Methodik

Dieses Kapitel stellt die Vorgehensweise zur Analyse und Lösung der erläuterten Problemstellung von α-Doyen und der Einführung von Empfangsbestätigungen vor. Dabei wird vornehmlich das Vorgehen beschrieben, das sich in der Struktur der einzelnen Kapitel dieser Arbeit wiederspiegelt. Es werden die einzelnen Bearbeitungsschritte erläutert, um dem Leser einen Überblick über die Herangehensweise an die Problemstellung der Arbeit zu geben.

Zu Beginn der Arbeit werden wissenschaftliche Grundlagen, die für das Verständnis von α-Flow relevant sind und als Grundlage für α-Doyen dienen, vorgestellt (cf. Kapitel 3). Dazu werden zunächst die grundlegenden Konzepte von α-Flow wie Workflows und das Workflow-Management näher betrachtet. Ausgehend von der α-Flow inhärenten dokumentenorientierten, fallbasierten Vorgehensweise wird dann der Bogen zum α-Flow-Domänenmodell und dessen Einbettung in die Systemarchitektur geschlagen. Das Modell und die Architektur werden eingehend untersucht, weil sie beim späteren Entwurf des Token-Modells erweitert werden. Weiterhin wird in diesem Kapitel ein Anwendungsszenario vorgestellt, dass die Verwendung von α-Flow anhand einer Brustkrebsbehandlung anschaulich beschreibt und die Basis für weitere Überlegungen liefert.

An die Betrachtungen des wissenschaftlichen Kontexts schließen sich in Kapitel 4 einige verwandte Arbeiten an, die ebenfalls das Token-Konzept verwenden, und solche, die im Rahmen von Empfangsbestätigungen relevant sind. Hier wurden verschiedene Techniken untersucht, um sich einen Überblick über bereits bestehende Konzepte zu verschaffen. Dazu gehört die Business Process Model and Notation, die Token zur Visualisierung des Prozessablaufs verwendet. In verteilten Systemen werden Token neben anderen Verfahren zur Synchronisierung und zur Auszeichnung von Sonderrollen benutzt. Im Rahmen von Empfangsbestätigungen bildet die E-Mail-Anwendung im Internet die Ausgangsbasis für verschiedene Ansätze von Empfangsbestätigungen.

Im Anschluss wird das Token-Modell detailliert ausgearbeitet und die prototypische Implementierung erläutert (cf. Kapitel 5). Dazu wird zuerst der bereits vorgestellte Anwendungsfall auf das Token-Modell übertragen, um eine Motivation für den Einsatz von Prozessrollenbeschreibungen zu schaffen. Ausgehend von einer fachlichen Anfor-

derungsanalyse werden verschiedene Lösungsansätze diskutiert und der am meisten geeignete ausgewählt. Die Nachrichten bei der Token-Weitergabe werden so gestaltet, dass sie die beschriebenen Änderungskonflikte bei Token-Änderungen vermeiden. In einem Feinentwurf wird das technische Fachkonzept mit seinen Schnittstellen beschrieben und die Anzeige- und Bedienkomponente im α-Editor entworfen.

In Kapitel 6 wird zunächst auf Basis des Anwendernutzens die Motivation für das Fachkonzept der Empfangsbestätigungen erläutert und auf Anforderungen an das verwendete Nachrichtenformat und die eingesetzte Speicherstruktur eingegangen. Nachfolgend wird die technische Umsetzung erläutert, die eine Konfiguration des α-Docs zum Versand von Empfangsbestätigungen zulässt.

Um das Nutzenpotenzial und den Mehraufwand von α-Flow zu messen, wird in Kapitel 7 ausgehend vom vorgestellten Anwendungsfall eine Evaluation durchgeführt, die zum einen die in α-Flow verwendeten Nachrichten untersucht und zum anderen ein Mengengerüst über die Anzahl der versendeten Nachrichten liefert. Die Evaluation gibt Aufschluss darüber, welche Nachrichtenanzahl und -größe in einem beispielhaft angenommenen Szenario bei verschiedenen Durchführungs- und Konfigurationsvarianten auftritt.

Abschließend werden in Kapitel 8 offene Punkte und Erweiterungsmöglichkeiten von α-Flow diskutiert, die sich im Laufe dieser Arbeit ergeben haben. Sie können Anhaltspunkte für weitere Untersuchungen am α-Flow-Projekt liefern.

3 Projektspezifische Grundlagen

Dieses Kapitel beschreibt die Grundlagen, die dem α-Flow-Projekt zugrunde liegen. Zu Beginn werden die Konzepte des Workflow und des Geschäftsprozesses voneinander abgegrenzt. Darauffolgend wird die dokumentenorientierte fallbasierte Vorgehensweise betrachtet, weil sie Basis für das Grundverständnis von α-Flow ist. Daran schließt sich eine detaillierte Beschreibung des α-Flow-Domänenmodells an.

3.1 Grundlegende Konzepte von α-Flow

α-Flow wird in [NL12] als ein verteiltes dokumentenorientiertes Prozessmanagementsystem (dDPM) beschrieben, welches zum Ziel hat, institutionsübergreifende Behandlungsprozesse im Gesundheitswesen zu unterstützen. Es ermöglicht den teilnehmenden Akteuren, ad-hoc auf Basis von aktiven Dokumenten (cf. [LED+99])[1] zusammenzuarbeiten. Die Akteure können direkt mit einem aktiven Dokument interagieren und das Dokument selbst hat ebenfalls aktive Eigenschaften.

Für die Interaktion der Akteure ist es nicht erforderlich, die einzelnen lokalen Systeme zu integrieren. Stattdessen kann auf eine verteilte Fallakte, das aktive Dokument, zurückgegriffen werden. In α-Flow wird diese Fallakte α-Doc genannt. Die fallbasierte Behandlung und die Zusammenarbeit der verschiedenen Akteure auf Basis von Dokumenten bilden die Grundlage für das Projekt. Darum werden sie nun neben einer grundlegenden Untersuchung von Workflows und Geschäftsprozessen zuerst betrachtet, um im Anschluss das α-Flow-Domänenmodell besser verstehen zu können.

3.1.1 Geschäftsprozess und Workflow

Ein Geschäftsprozess und ein Workflow beschreiben in ihrem Kern den gleichen Sachverhalt. Sie werden häufig synonym verwendet, haben aber einen anderen Schwerpunkt.

1 Ein aktives Dokument kann selbstständig auf Änderungen reagieren und kontextsensitiv Aktionen auslösen. Durch seine Fähigkeit, auch prozessrelevante Informationen zu speichern, unterstützt es kooperative Prozesse ([LED+99]).

Diese Unterschiede sollen hier herausgearbeitet und die beiden Begriffe gegenübergestellt werden.

Geschäftsprozess

Ein Prozess ist eine Ablaufbeschreibung von Aufgaben, Ausführungen, Arbeitsschritten usw., wobei zwischen den einzelnen Abschnitten Abhängigkeiten bestehen können (cf. [HS04, Seite 21]). Er besitzt sowohl einen definierten Anfang als auch ein festgelegtes Ende.

Ein Geschäftsprozess wiederum ist ein Prozess in einem Unternehmen, der dessen Unternehmensziele verfolgt. Er beschreibt eine Folge von Aktivitäten, die in einem Unternehmen ausgeführt werden müssen, um ein bestimmtes Ziel zu erreichen, z. B. die Herstellung eines Produktes oder das Erbringen einer Dienstleistung. Beim Ablauf eines Geschäftsprozesses ist die Reihenfolge der Ausführung der einzelnen Aktivitäten genau festgelegt und die jeweilige Ausführung dieser an Bedingungen geknüpft. Solche Bedingungen können z. B. eine vorliegende Eingabe oder ein auslösendes Ereignis sein. Eine Aktivität hingegen ist eine logische Arbeitseinheit, die beispielsweise von einer Person oder Maschine ausgeführt wird (cf. [AH02, Seite 3ff.], [Gad01, Seite 29f.]).

Workflow

Die Workflow Management Coalition (WfMC) ist ein Zusammenschluss von Forschern, Anwendern und Software-Unternehmen und befasst sich seit 1993 mit der Erstellung von Standards im Umfeld des Workflow-Managements. Um ein gemeinsames Verständnis in dieser Domäne zu erlangen, bietet die WfMC eine weithin anerkannte Definition für einen Workflow:

> „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.“ ([Coa99, Seite 8])

Sie versteht den Workflow als einen teilweise oder gänzlich automatisierten Geschäftsprozess, in dem Dokumente, Informationen oder Aufgaben zwischen den Teilnehmern weitergereicht werden, um eine entsprechende Menge von vorgegebenen prozeduralen Regeln auszuführen.

Gadatsch fügt dieser Definition eine formale Beschreibung des Workflows hinzu (cf. [Gad01]). Diese formale Beschreibung definiert die zeitliche, fachliche und ressourcenbezogene Spezifikation, die zur automatischen Ablaufsteuerung nötig ist ([Gad01, Seite 31]).

Gegenüberstellung

Sowohl Geschäftsprozesse als auch Workflows bilden Arbeitsabläufe ab. Aufgrund des gemeinsamen Fokus werden die beiden Begriffe in der Literatur häufig synonym verwendet ([AH02]). Die Abbildung 3.1 stellt allerdings die erkannten Eigenheiten der beiden Begriffe nochmal heraus.

	Geschäftsprozess	Workflow
Ziel	Gestaltung von Arbeitsabläufen im Sinne gegebener Ziele	Beschreibung der technischen Ausführung von Arbeitsabläufen
Gestaltungsebene	Konzeptionelle Ebene mit Verbindung zur Geschäftsstrategie	Operative Ebene mit Verbindung zur unterstützenden Technologie
Detailierungsgrad	Von einem Menschen ausführbare Aktivität (Arbeitsschritte)	Konkretisierung der Arbeitsschritte hin zu einzelnen Arbeitsanweisungen

Bild 3.1: Geschäftsprozess vs. Workflow (cf. [Gad01, Seite 35])

Prinzipiell beschreibt der Geschäftsprozess, „was“ zu tun ist, wohingegen der Workflow sich mit dem „wie“ beschäftigt. Der Geschäftsprozess ist eher konzeptioneller Art, der Workflow operativer. Ein Workflow ist deutlich detaillierter als ein Geschäftsprozess. Reicht es für einen Geschäftsprozess aus, dass ein Beteiligter seine Arbeitsschritte versteht, geht die Definition eines Workflow soweit, dass konkrete Arbeitsanweisungen gegeben werden und ggf. eine computer-basierte Ausführung stattfinden kann.

Die Unterscheidung ist sehr nuanciert, sodass die beiden Begriffe *Prozess* und *Workflow* trotz der ausgeführten Unterschiede im Detailierungsgrad häufig synonym verwendet werden. Im Rahmen von α-Flow werden insbesondere detaillierte Arbeitsanweisungen festgelegt und Ergebnisse dazu dokumentiert. Daher wird hier der *Workflow*-Begriff verwendet.

Workflow-Management

Die zentrale Idee hinter Workflow-Management (WfM) ist die teilweise oder ganzheitliche Automatisierung von Geschäftsprozessen bzw. Workflows durch ein Informationssystem.

In dessen Folge werden Dokumente, Informationen und Arbeitsanweisungen gemäß der Prozessdefinition von einem Prozessteilnehmer zum nächsten weitergegeben ([Coa99, Seite 8]).

Nach Georgakopoulos et al. umfasst das WfM primär drei Aufgaben (cf. [GHS95]): Zum einen die Modellierung des Geschäftsprozesses und das Festhalten in einer Workflow Spezifikation. Zum anderen beinhaltet es die Optimierung der Definition durch eine Reorganisation der Arbeitsschritte. Und schließlich strebt es die Implementierung des Workflow und seine Automatisierung durch ein Informationssystem an.

Ein Workflow-Management-System (WfMS) besteht aus verschiedenen Komponenten und unterstützt die Aufgaben des WfM ([Coa99, Seite 9]). Es hilft bei der Definition von Workflows, verwaltet diese und überwacht die Ausführung von Workflows durch eine Workflow Engine, die die Workflow Definition interpretieren kann. Eine sog. Workflow-Instanz, also beispielsweise genau ein Geschäftsvorfall, wird vom WfMS so abgearbeitet, dass genau der Ablauf erfolgt, der in der Definition festgelegt ist. Das WfMS kann den Prozessteilnehmern dann Arbeitsanweisungen zustellen und die Teilnehmer mit dem benötigten Informationen versorgen. Außerdem kann es z.B. andere Programme oder Prozesse anstoßen, falls diese benötigt werden.

Zusammenfassung

α-Flow erfüllt in seiner Auffassung prinzipiell den Workflow-Ansatz. Es beschreibt in seiner Ausprägung, wie etwas zu tun ist und gibt auf operativer Ebene detaillierte Anweisungen für die handelnden Akteure. Die Interaktion wird durch Austausch von Dokumenten erzielt.

Aus den in diesem Abschnitt dargestellten Definitionen ergibt sich der aktivitätenorientierte Workflow-Ansatz, bei dem die Aktivitäten und Akteure im Mittelpunkt stehen. Er lässt sich beispielsweise durch Petri-Netze mit Zuständen und Übergängen oder Business Process Model and Notation (BPMN) mit Akteuren und Aktivitäten darstellen. Dieser Ansatz ist kennzeichnet Aufgaben durch Vor- und Nachbedingungen und Ausnahmen aus.

Im Gegensatz dazu erreicht der dokumentenorientierte Ansatz eine Koordination der Teilnehmer durch Statusänderungen der hinterlegten Artefakte (cf. [NL12]). Der Status eines Artefakts kann beispielsweise von „Entwurf“ zu „Korrigiert“ und dann zu „Final“ wechseln.

Um Workflows praktisch anzuwenden, müssen sie detailliert beschrieben werden. Eine automatisierte Ausführung wird durch entsprechende Service-Implementierungen erreicht.

Da sich medizinische Prozesse von Geschäftsprozessen aber vor Allem durch ihre Wissensintensität unterscheiden, ist es gerade im medizinischen Umfeld wichtig, dass weniger auf einer starren Ausführung beharrt wird, sondern vielmehr im Behandlungsverlauf vernünftig unterstützt wird.

Da Workflow-Systeme häufig server-zentriert sind, werfen dezentralisierte Ansätze noch häufig Probleme auf. Weiterhin sind Ad-hoc-Workflows, bei denen zu Beginn noch nicht alle teilnehmenden Akteure, Zustände und Übergänge bekannt sind, schwer im Voraus zu modellieren.

3.1.2 Dokumentenorientierte fallbasierte Vorgehensweise

Die Einschränkungen der bisher genannten Workflow-Ansätze greift α-Flow auf und wendet gleichzeitig den im Bereich des Gesundheitswesens vorherrschenden dokumentenorientierten Ansatz ab. In Gesundheitsbereich basiert die Arbeit der Ärzte auf Fallakten, in denen sie alle zu einem Patienten relevanten Informationen und Unterlagen ablegen. Eine Zusammenarbeit wird erreicht, indem diese Fallakte geteilt wird, meist in der Form, dass sie dem Patienten mitgegeben wird und er sie dem nächsten Arzt vorlegt. Arbeitsanweisungen und Diagnosen werden auf Überweisungsträgern und in Arztbriefen festgehalten. Jedes in der Fallakte enthaltene Dokument entspricht daher dem Ergebnis einer von einem beteiligten Arzt durchgeführten Aktivität.

Diese fallbasierte Vorgehensweise verfügt über bestimmte Eigenschaften (cf. [NL12]): Zu Beginn des Behandlungsprozesses ist noch nicht bekannt, welche Akteure später daran beteiligt sein werden. Ebensowenig kann vorhergesagt werden, wie der nächste Behandlungsschritt aussehen wird (Ad-hoc-Workflow). Weiterhin muss sichergestellt werden, dass die Prozessbeteiligten über die aktuellsten Informationen verfügen. Insgesamt ist der fallbezogene Workflow also nicht vollständig standardisiert und bei dessen Abarbeitung herrschen höhere Freiheitsgrade vor als bei einem aktivitätenorientierten Workflow.

Aalst et al. stellen dazu unterschiedliche Anforderungen gegenüber (cf. [AWG05]). Das von ihnen eingeführte sog. *Case Handling Paradigm* basiert darauf, dass die Kontrolle über den Workflow bei einer verteilten Fallakte liegt. Diese unterstützt die Beteiligten, Entscheidungen zu treffen. Vordefinierte Prozessschritte hingegen fehlen an dieser Stelle.

Die Aktivitäten im Workflow sind in diesem Zusammenhang dann die ärztlichen Untersuchungen und therapeutischen Behandlungsschritte. Der Workflow selbst entspricht der Fallakte.

3.2 α-Flow Domänenmodell

Die Ausführungen dieses Kapitels stammen hauptsächlich aus den Veröffentlichungen [NSWL11], [NL12] sowie aus den Abschlussarbeiten [Wah11], [Had11] und [Sch11]. [Wah11] beschäftigt sich eingehend mit der Synchronisation von Informationen zwischen den einzelnen Akteuren. [Had11] entwirft ein Versionsverwaltungssystem, um einzelne Prozessartefakte individuell und dauerhaft zu archivieren. Und [Sch11] beschreibt das evolutionsfähige Metadatenmodell α-Adaptive. Ausführlichere Informationen zur Komponente α-Properties finden sich in [TN11] und [Tod10]. Da das α-Flow-Domänenmodell die Grundlage für die weitere Arbeit bildet, wird dieses im Folgenden zusammenfassend dargestellt, wobei der Schwerpunkt auf die im weiteren Verlauf dieser Arbeit relevanten Bereiche gelegt wird.

3.2.1 α-Episode und α-Doc

Durch α-Flow wird die traditionelle papierbasierte Vorgehensweise elektronisch nachgebildet: Das α-Doc entspricht einer gemeinschaftlichen verteilten Fallakte, in die alle zu einer Behandlungsepisode gehörenden Dokumente abgelegt werden. Mit Hilfe des α-Doc können diese Dokumente unter den Teilnehmern verteilt werden. Ein α-Doc umfasst dabei einen Behandlungsfall. Hierbei entspricht damit ein α-Doc genau einer sog. α-Episode. Es besteht also eine 1-1-Beziehung zwischen α-Doc und α-Episode. Das α-Doc soll dabei den Gegenstand der Fallakte symbolisieren, wohingegen der Term α-Episode auf den Workflow-Charakter der Behandlungsepisode mit Aktivitäten und Behandlungssschritten abzielt. Ein α-Doc ist wiederum aus mehreren α-Cards (1-N-Beziehung) aufgebaut. Eine neue α-Card wird erstellt, wenn ein neuer Behandlungsschritt eingeleitet werden soll. Die Behandlung schreitet mit dem Erstellen und Füllen von α-Cards voran ([NSWL11]).

3.2.2 α-Cards

Die in der Fallakte verwalteten Informationen werden durch α-Cards modelliert. Sie stellen in sich geschlossene Informationseinheiten dar, die als Basis für die Validierung dienen und die Sichtbarkeit für andere Akteure bestimmen. Eine neu erstellte α-Card ist gleichbedeutend mit einer Arbeitsanweisung an einen Akteur. Sie ist eine Aufgabe, die Handeln erfordert. Das Ergebnis dieses Handelns kann dann als Anhang (sog. Payload) der α-Card gespeichert werden. Beispielhaft will ein Arzt seinen Patienten an einen anderen Arzt für eine weitergehende Untersuchung überweisen. Er fügt dazu eine neue

α-Card mit dem Auftrag ein, in deren Payload der andere Arzt, an den der Auftrag gerichtet ist, seine Ergebnisse abspeichern kann.

α-Cards bestehen prinzipiell aus zwei Teilen: dem Deskriptor und dem Payload. Dabei besteht der Deskriptor wiederum aus einem eindeutigen Identifizierer der α-Card und den sog. α-Adornments (cf. Abbildung 3.2). α-Adornments enthalten Informationen, die zur Steuerung des Prozesses eingesetzt werden können. Sie geben den Status einer einzelnen α-Card wieder und können bei einer Änderung auch aktiv Ereignisse auslösen. Die Grundlagen des α-Adornment-Modells stammen aus [NL09] (für weitergehende Ausführungen wird auf Abschnitt 3.2.3 verwiesen).

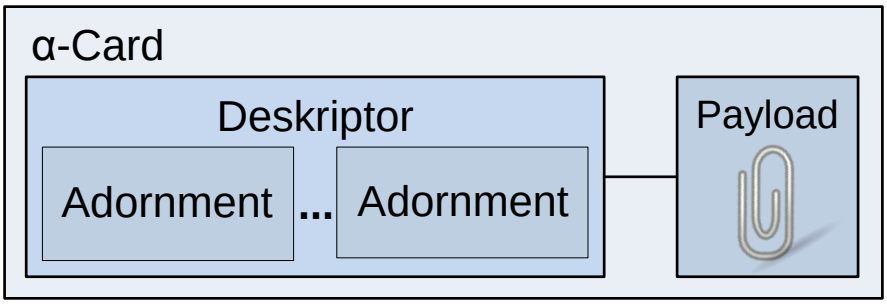


Bild 3.2: Aufbau einer α-Card

Als Basis-Adornments werden dort der Autor (*contributor*), der Betrachtungsgegenstand der α-Card (*object under consideration*), die Sichtbarkeit (*visibility*), die Gültigkeit (*validity*), der syntaktische Typ des Payloads, der fundamentale semantische Typ des Payloads sowie der domänenspezifische semantische Typ des Payloads genannt.

Ein Beispiel für einen Autoren wäre der behandelnde Arzt. Der Betrachtungsgegenstand der α-Card ist beispielsweise der zu behandelnde Patient bzw. der Behandlungsname der Therapie. Das Sichtbarkeitsadornment wird getrennt von der Gültigkeit behandelt, weil dies bei einer dokumentenorientierten Betrachtungsweise eher dem üblichen Vorgehen entspricht. So ist es durchaus üblich, vorläufige Entwürfe von Dokumenten der Allgemeinheit zugänglich zu machen, ohne dass garantiert wird, dass diese abgeschlossen bzw. fehlerfrei sind. Der Einfachheit halber werden für die Sichtbarkeit die beiden Werte „öffentlich“ und „privat“ und für die Gültigkeit „gültig“ und „nicht gültig“ angenommen. Der syntaktische Typ des Payloads spezifiziert auf technischer Ebene das Format des angehängten Payloads. Der domänenspezifische semantische Typ des Payloads hingegen gibt eine Auskunft über den Inhalt des Payloads (e. g. Diagnose, Therapiemaßnahme, Rezept).

Der fundamentale semantische Typ des Payloads unterteilt die α-Cards in Content- und Coordination-α-Cards[1]. Damit können die verwalteten Informationen also prinzipiell in zwei Arten kategorisiert werden:

- Informationen, die der Koordination dienen und prozessrelevante Daten enthalten (**Coordination**) und
- Informationen, die medizinischen Inhalt tragen (**Content**).

Das Payload der Coordination-α-Cards beinhaltet anders als bei Content-α-Cards keine medizinische Nutzinformationen, sondern stattdessen Prozessinformationen. Einen Überblick über ein α-Doc und seine α-Cards liefert Abbildung 3.3.

Coordination-α-Cards

Die Coordination-α-Cards beinhalten prozessrelevante Meta-Informationen. Sie verwalten das Prozessschema sowie die teilnehmenden Akteure. Prinzipiell können drei solcher α-Cards in einem α-Doc identifiziert werden: Das Process Structure Artifact (PSA), das Collaboration Resource Artifact (CRA) und das Adornment Prototype Artifact (APA). Diese Informationseinheiten existieren prinzipiell nur einmal pro α-Doc. Sie unterscheiden sich zudem von den Content-α-Cards dadurch, dass sie von allen Prozessteilnehmern verändert werden können.

Das PSA beinhaltet in seinem Payload das dynamische Prozessschema mit den Beziehungen zwischen den einzelnen Teilnehmern. Dazu wird eine Liste aller α-Cards geführt mit Informationen zu deren wechselseitigen Beziehungen.

Das Payload des CRA enthält Informationen über alle Prozessteilnehmer, deren Institutszugehörigkeit und ihre Rollen. Außerdem werden hier die Kommunikationsendpunkte der Teilnehmer gespeichert, also die Adresse, unter der ein Teilnehmer erreicht werden kann. Des Weiteren enthält das CRA das *Object under Consideration*, also die Patienteninformationen.

Das APA speichert in seinem Payload eine Vorlage für alle domänenspezifischen α-Adornments und deren Attribute; α-Adornments und die BEdeutung des APA als Vorlage werden näher im anschließenden Abschnitt 3.2.3 betrachtet.

1 Es wird darauf hingewiesen, dass in dieser Arbeit einige englische Begriffe nicht übersetzt werden, wenn eine Übersetzung dem Verständnis nicht förderlich wäre und der Begriff als feststehende Wendung verwendet wird.

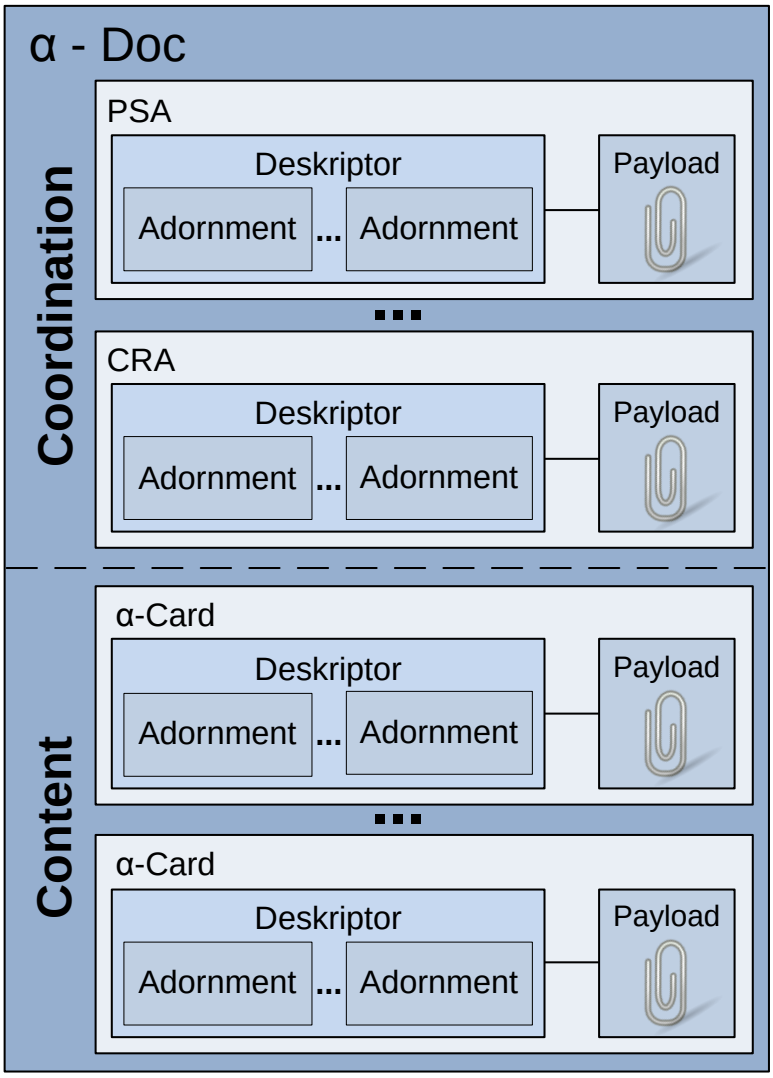


Bild 3.3: Aufbau eines α-Doc

Content-α-Cards

Content-α-Cards haben in ihrem Payload die medizinischen Nutzinformationen in Form von Dokumenten-Artefakten gespeichert. Diese können in einem beliebigen binären Format vorliegen, beispielsweise als ein Word-Dokument, eine Portable Document Format (PDF)-Datei oder eine Health Level 7 (HL7) Clinical Document Architecture (CDA). Inhaltlich kann es beispielsweise ein ärztlicher Befund, eine Diagnose oder die Beschreibung einer therapeutische Maßnahme sein. Diese Nutzinformationen entstehen, wenn ein handelnder Akteur die Ergebnisse seines Handelns dokumentiert und diese als Payload an die α-Card anhängt.

3.2.3 α-Adornment-Modell

Die Grundlagen zum adaptiven Adornment-Modell stammen aus [Sch11] und [NSWL11]. Dort wird ausgeführt, wie neben den bereits genannten α-Adornments (cf. Abschnitt 3.2.2) das Modell adaptiv angepasst werden kann. Ziel des Modells ist es, erweiterbar zu sein, um Attribute, die zur Entwurfszeit nicht bekannt sind, zur Laufzeit hinzufügen zu können, bestehende Attribute anzupassen und ggf. auch Attribute zu löschen.

Das α-Adornment-Modell basiert auf dem bei evolutionären Systemen bekannten Prinzips des verzögerten Entwurfs (cf. [Pat02]). Werden semantische Entscheidungen beispielsweise in einem Datenbankschema manifestiert, können sie nur unter großem Aufwand und Anpassung des Schemas verändert werden. Wenn semantische Entscheidungen bis zur Laufzeit verzögert werden, dann ist die Verwirklichung eines Systems mit dauerhaftem Anpassungsvermögen gegeben (cf. [Len09]). Die für den verzögerten Entwurf verwendeten Konzepte der Prototypbasierten Programmierung und des Entity-Attribute-Value (EAV)-Modells ermöglichen eine kontinuierliche Anpassbarkeit des Adornment-Modells und die Erweiterbarkeit durch benutzerdefinierte Attribute zur Laufzeit (cf. [NSWL11]).

Auf das α-Adornment-Modell und die ihm zugrunde liegenden Konzepte wird an dieser Stelle detailliert eingegangen, weil sie für die vorliegende Arbeit eine zentrale Rolle spielen. Die bei α-Adaptive verwendeten Ansätze werden im Rahmen der Entwicklung des Token-Modells wieder aufgegriffen.

Prototypbasierte Programmierung

Das Prinzip der Prototypbasierten Programmierung besteht darin, dass nicht abstrakte Klassen zur Bereitstellung von gemeinsamen Attributen und Verhalten von konkreten Objekten verwendet werden, sondern stattdessen Prototyp-Objekte benutzt werden ([Bor86]). Ein neues Objekt wird durch sog. Klonen des bestehenden Prototyp-Objekts erstellt. Dabei wird das Konzept der Vererbung beibehalten, indem jeweils ein Verweis auf die Klon-Basis mitgeführt wird. Wenn nun der Prototyp verändert wird, werden auch alle Ableitungen automatisch aktualisiert. Der große Vorteil gegenüber der klassenbasierten Vorgehensweise besteht darin, dass sowohl das Schema des Prototypen und der Klone als auch deren Werte zur Laufzeit verändert werden können.

Bei der klassenbasierten Vorgehensweise werden Objekte durch Instanzieren einer Klasse erzeugt. Soll ein Objekt mit anderen Eigenschaften und Verhalten erzeugt werden, muss dazu eine neue Klasse erstellt werden. Damit muss die semantische Entscheidung für ein

Objekt während der Systemkonzeption erfolgen. Die Prototypbasierte Programmierung hingegen erlaubt es, Änderungen am Design zur Laufzeit durchzuführen.

EAV-Modell

Das Entity-Attribute-Value-Modell ist ebenfalls ein Verfahren des verzögerten Entwurfs. Es dient insbesondere der Persistierung von heterogenen und variablen Daten und kommt häufig in der Medizin beim Speichern von patientenbezogenen Daten zum Einsatz ([NMC+99]). Es hat den Vorteil, dass zur Entwurfszeit nicht bekannt sein muss, wie viele Attribute ein Objekt zur Laufzeit haben wird bzw. haben soll.

Beim herkömmlichen Datenbankdesign wird jedem Attribut eines Entity-Typs eine Spalte in einer Tabelle zugeordnet. Zur Entwurfszeit wird damit die Semantik der Daten im Schema fixiert. Änderungen am Datenmodell gestalten sich im Folgenden schwierig, da immer das Schema geändert werden muss.

Im Gegensatz zum konventionellen Ansatz liefert das EAV-Modell ein generisches Datenschema. Es basiert auf einer Tabelle mit drei Spalten:

1. Die erste Spalte identifiziert das Entity.
2. Die zweite Spalte legt den Namen des Attributs fest oder identifiziert dieses.
3. Die dritte Spalte beinhaltet den eigentlichen Attributwert.

Den Primärschlüssel der Tabelle bilden die erste und zweite Spalte, er besteht also aus der Entity-ID und der Attribut-ID. Für jedes Attribut-Wert-Paar wird eine neue Zeile in dieser Tabelle hinzugefügt. Neue Attribute bedeuten in diesem Modell also keine Erweiterung der bestehenden Tabellen um eine weitere Spalte, sondern das Hinzufügen einer weiteren Zeile in die Tabelle.

Prinzipiell hat das EAV-Modell den Vorteil der Evolutionsfähigkeit. Es können also beliebig viele Attribute zu einem Entity-Typen hinzugefügt werden, ohne das Schema der Datenbank zu verändern. Des Weiteren kommen in der Tabelle keine *null*-Werte vor, da nur benötigte Attributwerte tatsächlich gespeichert werden.

Neben diesen Vorteilen, gibt es aber auch einige Nachteile. So werden durch die Entkopplung vom logischen und physischen Datenbankschema Datenbankanfragen deutlich komplizierter. Es ist ein hoher Programmieraufwand zu leisten, um dem Benutzer die physische Schicht zu verbergen. Außerdem geht ein Teil der Semantik auf der Datenbankebene verloren. Beispielsweise können keine Integritätsbedingungen mehr festgelegt werden und die Überprüfung von Plichtfeldern, die zwingend mit einem Wert belegt werden müssen, ist auch nicht möglich. Sind diese Eigenschaften aber gefordert, müssen sie in

eigens für diesen Funktionsumfang erstellten Tabellen abgelegt und ausprogrammiert werden.

In Abbildung 3.4 sieht man ein Beispiel in Anlehnung an [Len10] das konventionelle Datenbankdesign und das EAV-Modell gegenübergestellt. Konventionell wird für eine Untersuchung ein Eintrag in einer umfangreichen Tabelle angelegt, der die verschiedensten Ergebnisse enthalten kann. Dadurch, dass die einzelnen Attribute vordefiniert sind, kann es an dieser Stelle zu vielen Null-Werten kommen, wenn bestimmte Werte gar nicht untersucht wurden. Genauso kann es andersherum vorkommen, dass ein bestimmtes Untersuchungsergebnis nicht vorgesehen ist.

Dem gegenüber steht der generische Entwurf in Form eines EAV-Modells. Kommt es hier zu einer Untersuchung, wird für jeden Wert ein neuer Eintrag in der Tabelle „Ergebnis“ erstellt. Es existieren am Ende also nur genauso viele Werte, wie auch Untersuchungen durchgeführt wurden. Des Weiteren können neue Ergebnistypen (z. B. neue Untersuchungsmethoden, neue relevante Werte etc.) hinzukommen. Für die Attributwerte können prinzipiell verschiedene Wertebereiche vorkommen. Häufig wird deswegen für jeden dieser Wertebereiche eine eigene EAV-Tabelle angelegt.

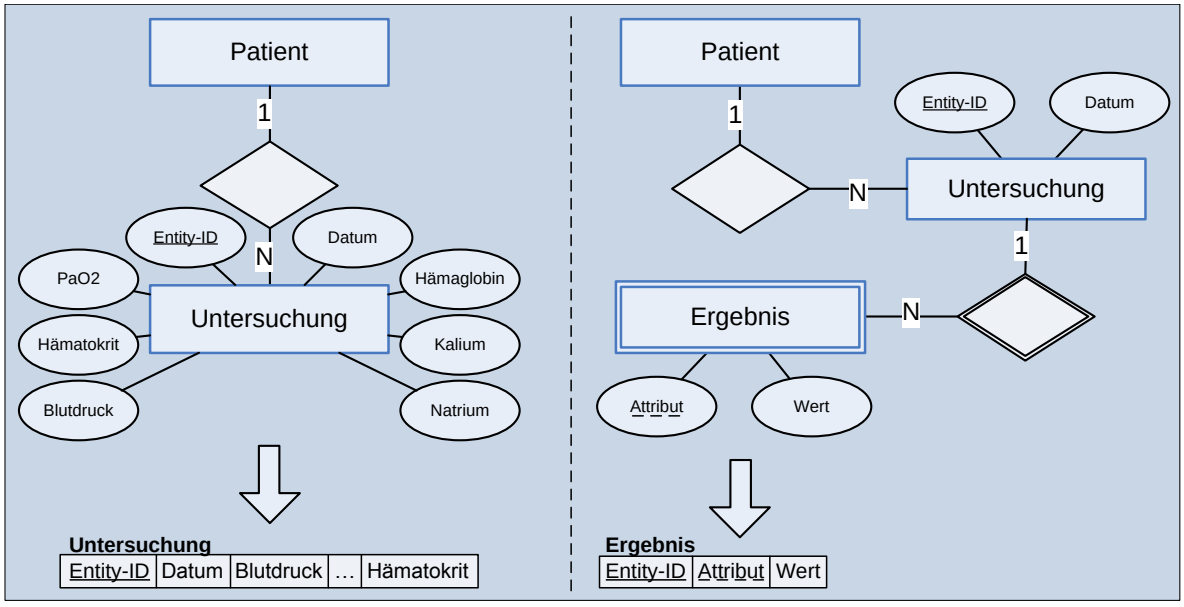


Bild 3.4: Konventionelles Datenbankdesign vs. Entity-Attribute-Value-Modell (in Anlehnung an [Len10] und [Sch11])

Das adaptive Adornment-Modell basiert, wie bereits erläutert, auf dem EAV-Modell. Wie in Abbildung 3.5 zu sehen ist, enthält der α-Card-Deskriptor die α-Adornments. Das elementare EAV-Schema wird dabei um einige Elemente erweitert.

Das Attribut *Benutzer-zentrischer Datentyp* wird eingeführt, weil α-Flow neben den im klassischen EAV-Modell vorgesehenen String-Werten auch andere Datentypen verwendet. Dazu zählen numerische Attribute (z. B. Version), zeitbezogene Attribute (z. B. Stichtag), längere Textfelder ähnlich einer Haftnotiz und Aufzählungen. Die Verwendung dieser Benutzer-zentrischen Datentypen erhöht insgesamt die Benutzerfreundlichkeit, da sie eben nicht wie beispielsweise bei Extended Markup Language (XML) System-bezogen sind sondern sich auf die Benutzersicht von Datentypen beziehen. Dazu gehört eine einfache Vorstellung von Datentypen mit verborgener detaillierter Definition derselben.

Das Attribut *Konsensebene* unterstützt die Prozessteilnehmer in der Konsensfindung über der Adornment-Menge, indem es die Adornments in Ebenen einteilen lässt. Im Fall von α-Flow sind dies eine generische Basisebene, die alle Adornments zur Gewährleistung der Systemfunktionalität enthält (*generic*). Darüber liegt die Ebene, die alle episodenspezifischen Adornments enthält (*episode-specific*). Dann folgen diejenigen Adornments, über die ein Konsens auf Institutionsebene besteht (*institution-specific*). Desweiteren gibt es deine domänenspezifische Adornment-Ebene (*domain-specific*). Insgesamt gilt: je höher der Konsens, desto höher die Ebene.

Das *Instanz-Attribut* wird benötigt, um das Modell um Prototyp-orientierte Semantik zu erweitern (Vorlagenfunktion eines α-Card-Deskriptors). Mit diesem Attribut kann eine Teilmenge von allen verfügbaren Adornments maskiert werden. Das setzen dieser Maskierung für ein einzelnes Adornment bedeutet, dass es bei diesem individuellen α-Card-Deskriptor verwendet wird.

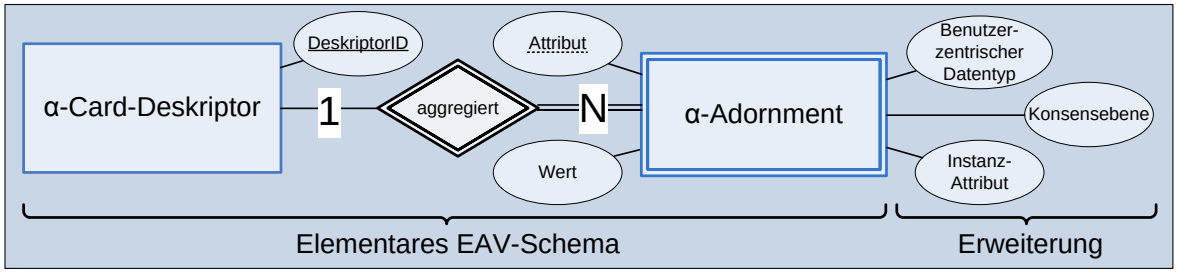


Bild 3.5: Adornment-Modell (in Anlehnung an [Sch11])

Adornment Prototype Artifact

Bisher kann jeder einzelne α-Card-Deskriptor einzeln an die Bedürfnisse der Prozessteilnehmer angepasst werden. Allerdings ist es der Wunsch der Teilnehmer innerhalb einer Behandlungsepisode (wenn nicht sogar innerhalb von Institutionen oder Domänen) einen

Konsens über die verwendeten Adornments anzustreben. Demzufolge wird ein Prototyp benötigt, der als Vorlage für alle weiteren Deskriptoren dient.

Das APA ist genau dieser Prototyp, der auf den erläuterten Konzepten des prototypbasierten Programmierens basiert. Im APA wird der Konsens der beteiligten Akteure über die Adornments festgehalten. Wird eine neue α-Card innerhalb eines α-Docs erstellt, soll diese immer auf dem gleichen Attributmodell basieren wie es im APA definiert ist. Dazu wird beim Erstellen eines neuen α-Card-Deskriptors das APA geklont, um die Standard-Adornments zu erhalten. Als Standardwert wird den Adornments der im APA vorliegende Wert zugewiesen. Sie können aber im weiteren Verlauf mit eigenen Werten belegt werden.

Ein α-Card-Deskriptor wird gewöhnlich nicht alle verfügbaren Standard-Adornments verwenden. Um eine Trennung zwischen den tatsächlich verwendeten Adornments und den vorkonfigurierten zu gewährleisten, wird in α-Flow zwischen dem Adornment-Schema und den Adornment-Instanzen unterschieden. Das Schema umfasst dabei alle im APA definierten Adornments. Wird ein Adornment zur Verwendung im α-Card-Deskriptor ausgewählt, wird das bereits erwähnte Instanz-Attribut gesetzt. Ist es hingegen nicht gesetzt, wird das Adornment in diesem Deskriptor nicht verwendet.

Wenn das Instanz-Attribut im APA gesetzt ist, wird das Adornment automatisch in die davon geklonten Deskriptoren übernommen. Dies kann aber wiederum durch individuelles Aufheben des Attributs rückgängig gemacht werden.

Klonen des APA

Wie in [Sch11] ausgeführt, muss das Klonen des APA eine tiefe Kopie erzeugen. Die von Java gelieferte *clone()*-Methode liefert allerdings nur flache Kopien. Dabei wird zwar das Originalobjekt kopiert, aber nicht die Objekte, auf die es verweist. Das hat zur Folge, dass Klon und Referenz auf dieselben Objekte verweisen. Die benötigten unabhängigen α-Card-Deskriptoren werden deswegen unter Verwendung von Serialisierung und anschließender Deserialisierung erzeugt. Dieses Vorgehen erzeugt echte tiefe Kopien vom APA.

Das APA kann dann vom Akteur durch selbst definierte Adornments erweitert werden. Für die neu definierten α-Adornments können dann sowohl der Datentyp als auch ein Gültigkeitsbereich der Werte festgelegt werden. Dieses Vorgehen ermöglicht es den Ärzten, das Modell zur Laufzeit an ihre Bedürfnisse anzupassen.

3.2.4 Off-line-Synchronisierung

Durch die Komponente α-OffSync kann ein α-Doc auf den durchgeführten Änderungen lokal eine Ordnung herstellen. Sie stellt Offline-Konzepte für die Synchronisation bereit, da in α-Flow nicht davon ausgegangen werden kann, dass alle Knoten gleichzeitig online sind. Die Komponente erhöht auf der einen Seite die Komplexität des Systems, bietet aber auf der anderen Seite den Vorteil, unter Verwendung von logischen Zeitstempeln verteilt durchgeführte Änderungen zu synchronisieren, wobei aufgrund von lokalen Informationen eine global einheitliche Ordnung hergestellt wird. Ausgehend von den in [Wah11] und [NWL12] beschriebenen logischen Zeitstempeln wird erläutert, wie diese dazu verwendet werden, eine kausale Ordnung herzustellen. Des Weiteren wird die Erkennung von Änderungsanomalien und ihr Einfluss auf die Versionshistorie betrachtet (cf. [Had11]).

3.2.4.1 Aufbau der logischen Zeitstempel

Der Synchronisierungsmechanismus basiert auf in einer Liste angeordneten logischen Zeitstempeln, die an Vektoruhren und Versionsvektoren angelehnt sind. In α-OffSync wird diese Zeitstempel-Struktur Adaptive Version Clock (AVC) genannt (cf. [NWL12]).

Eine AVC wird durch eine Datenstruktur V mit n Einträgen dargestellt, wobei n die Anzahl der Teilnehmer ist. In dieser Struktur V wird in jedem Eintrag wie bei einem assioziativen Feld einem Schlüssel ID_{a_i} ein Wert c_i zugeordnet. Mit ID_{a_i} wird der i-te Akteur a_i aus der Menge der Akteure A eindeutig identifiziert. c_i stellt den diskreten Zähler dar, der die Anzahl der von a_i durchgeführten Änderungen wiedergibt. Bei Zugriff auf $V[ID_{a_i}]$ erhält man als Ergebnis $c_i, \forall(i = 1,...,n)$.

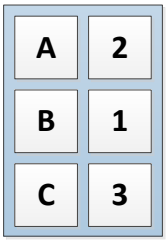


Bild 3.6: Beispiel eines Zeitstempels

Die Abbildung 3.6 zeigt beispielhaft eine AVC, bei der drei Akteure für Modifikationen verantwortlich waren. Es ist zu sehen, dass Akteur A zwei Änderungen, Akteur B eine Änderung und Akteur C drei Änderungen durchgeführt hat.

Eine AVC umfasst dabei nur die Akteure, die bereits Änderungen durchgeführt haben. An dem Prozess, der den Zeitstempel aus Abbildung 3.6 enthält, können noch weitere

Akteure beteiligt sein. Diese haben allerdings noch keine Änderungen durchgeführt. Die AVC ist so aufgebaut, dass dynamisch weitere Einträge für neue Akteure hinzugefügt werden können.

3.2.4.2 Zeitliche Ordnung auf Zeitstempeln

Im Rahmen des Vergleichs zweier Versionen ist es wichtig, auf der Grundlage von zwei AVCs, die als Zeitstempel fungieren, eine kausale Ordnung herstellen zu können. Dabei kann es sein, dass die beiden Zeitstempel identisch sind, es keine Abhängigkeit gibt oder die beiden Zeitstempel voneinander abhängen. Um den Vergleich durchzuführen, müssen die einzelnen Einträge jeweils miteinander verglichen werden. Als Ergebnis lässt sich dann über die Beziehung zweier Zeitstempel V_1 und V_2 sagen:

1. $V_1 \equiv V_2$: Die Zeitstempel sind identisch.
2. $V_1 > V_2$: V_1 besitzt eine kausale Abhängigkeit zu V_2.
3. $V_1 < V_2$: V_2 besitzt eine kausale Abhängigkeit zu V_1.
4. $V_1 \mid\mid V_2$: V_1 und V_2 stehen in keinem Zusammenhang, sind also nebenläufig entstanden.

Um zwei Zeitstempel miteinander zu vergleichen, müssen die beiden Zeitstempel elementweise miteinander verglichen werden. Die nachfolgenden Regeln bieten die Basis für den Vergleich:

1. $V_1 \equiv V_2$ genau dann, wenn $\forall i$ (i=1...n): $V_1[ID_{a_i}] = V_2[ID_{a_i}]$
2. $V_1 > V_2$ genau dann, wenn $\forall i$ (i=1...n): $V_1[ID_{a_i}] \geq V_2[ID_{a_i}]$ und $\neg(V_1 \equiv V_2)$
3. $V_1 < V_2$ genau dann, wenn $\forall i$ (i=1...n): $V_1[ID_{a_i}] \leq V_2[ID_{a_i}]$ und $\neg(V_1 \equiv V_2)$
4. $V_1 || V_2$ genau dann, wenn $\neg(V_1 < V_2)$ und $\neg(V_2 < V_1)$

Die in Abbildung 3.7 dargestellten Beispiele von Zeitstempelpaaren mit ihrer jeweiligen kausalen Beziehung zueinander sollen die Anwendung dieser Regeln verdeutlichen.

Die beiden Zeitstempel in Abbildung 3.7a) sind identisch, da die Wertpaare aller Einträge identisch sind. Der rechte Zeitstempel in Abbildung 3.7b) hat eine kausale Abhängigkeit vom linken Zeitstempel, da er zwar in den Einträgen für B und C mit diesem übereinstimmt, aber im Eintrag für A einen kleineren Wert aufweist. Die Zeitstempel in Abbildung 3.7c) haben keinen Zusammenhang, sind also durch nebenläufige Änderungen entstanden. Weder der linke, noch der rechte Zeitstempel kann den jeweils

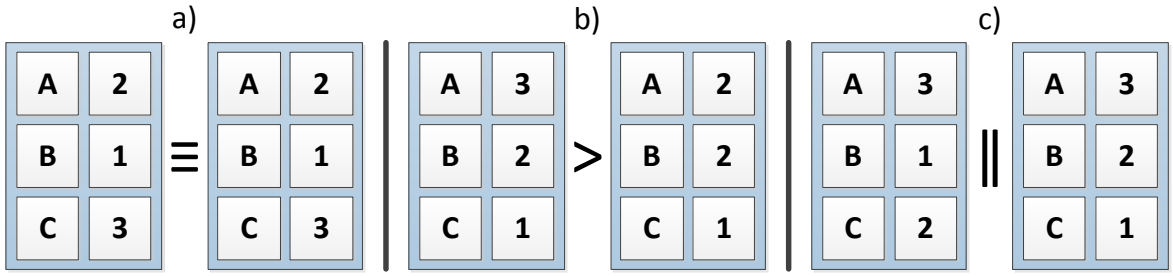


Bild 3.7: Beispiele für die kausale Beziehung zwischen Zeitstempeln

anderen dominieren, denn der linke Wert für B ist kleiner, der Wert für C dafür aber größer. Nebenläufige Zeitstempel bedeuten einen Änderungskonflikt, der durch parallele Modifikationen entstanden ist.

3.2.4.3 Erkennung von Änderungsanomalien

Insbesondere zwei Änderungsanomalien sind möglich und müssen gesondert betrachtet werden (cf. [Had11]). Abbildung 3.8 zeigt das Eintreffen von Nachrichten in einer veränderten Reihenfolge als die Sendereihenfolge (cf. Abbildung 3.8a) und das Auftreten von nebenläufigen Änderungen (cf. Abbildung 3.8b). Die Abbildung zeigt zwei Prozessteilnehmer, die Änderungen an der gleichen α-Card durchführen. Im linken Teil der Abbildung kommen zwei durch Teilnehmer A durchgeführte Änderungen in umgekehrter Reihenfolge bei Teilnehmer B an. Teilnehmer B erhält also zuerst die zweite Änderung und dann erst die erste. Im rechten Teil der Abbildung führen beide Teilnehmer gleichzeitig eine Änderung an der α-Card durch.

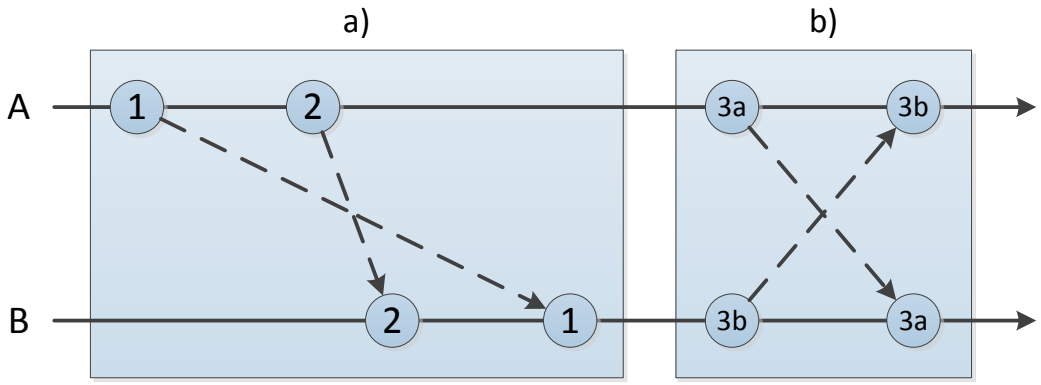


Bild 3.8: Synchronisierungsanomalien

Die angesprochenen Änderungsanomalien können durch Versionsvektoren, die in den ausgetauschten Nachrichten enthalten sind, erkannt werden. Dazu müssen die ankommenden Versionen mit den aktuellsten lokal vorliegenden Versionen verglichen werden. Abbildung 3.9 zeigt am Beispiel von Abbildung 3.8, wie die Versionsvektoren dazu benutzt werden können, die beschriebenen Anomalien zu erkennen.

Sowohl im Fall der divergierenden Sende- und Empfangsreihenfolge, als auch im Fall der nebenläufigen Änderungen kann mit Hilfe der in der gesendeten Nachricht enthaltenen Versionsvektoren erkannt werden, dass eine Änderungsanomalie aufgetreten ist.

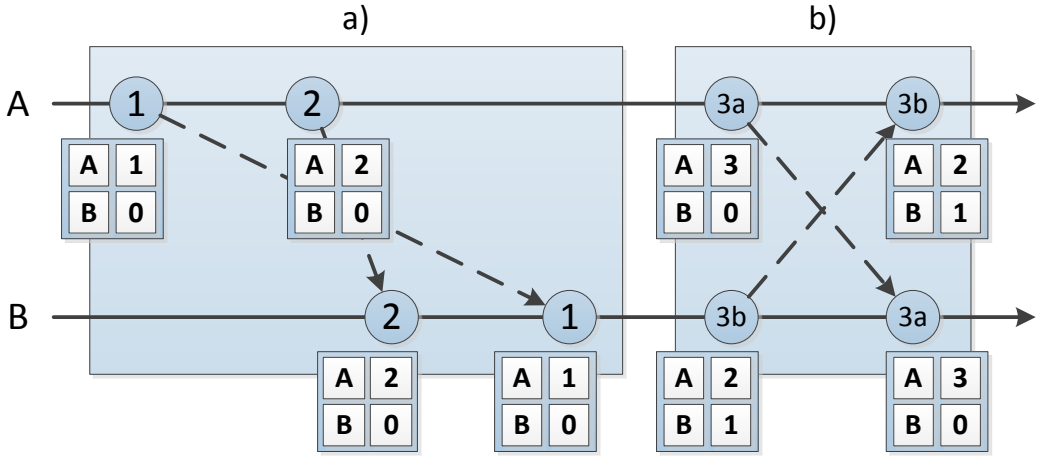


Bild 3.9: Erkennung von Synchronisierungsanomalien mit Versionsvektoren

3.2.4.4 Umgang mit erkannten Änderungsanomalien

Um die Nachvollziehbarkeit der durchgeführten Änderungen sicherzustellen, wird die Entstehungsgeschichte der Prozessartefakte dokumentiert. Neu entstandene Versionen werden daher in die Historie der Prozessartefakte eingefügt. Werden Änderungsanomalien erkannt, muss die Historie entsprechend erweitert werden.

Damit fehlende Nachrichten später in die Historie eingefügt werden können, werden beim Eintreffen neuer Nachrichten entsprechend der Distanz der neuen Versionen Lücken gelassen, die vorerst mit Platzhalterversionen gefüllt sind. Sobald die fehlenden Nachrichten eintreffen, wird in der Historie an die entsprechende Stelle navigiert und die Änderungen an der richtigen Position eingefügt. Im Fall der Abbildung 3.9a wird beim Eintreffen der Nachricht der zweiten Änderung von Teilnehmer A anhand der AVC erkannt, dass es vor dieser Änderung bereits zu einer weiteren Version gekommen sein muss. Daher wird beim Einfügen in die Versionshistorie ein Platzhalter für das

nachträgliche Einfügen der ersten Version erstellt und erst nach diesem die erhaltene Version eingefügt.

Der Umgang mit nebenläufigen Änderungen gestaltet sich in einem Offline-Umfeld schwieriger, denn es muss eine automatische Aussage über das gewünschte Resultat der Änderungen getroffen werden ohne dabei mit den anderen Teilnehmern Rücksprache halten zu können. Da im Fall von α-Flow eine minimale Benutzerinteraktion in diesem Fall erwünscht ist, wird der Nutzinhalt bei der Erkennung eines nebenläufigen Änderungskonflikts auf die letzte gültige Version gesetzt und gleichzeitig eine neue Version erzeugt. Die in der Zwischenzeit getätigten Änderungen werden zur Nachvollziehbarkeit und zu Dokumentationszwecken in einem Konfliktzweig gespeichert. Abbildung 3.10 zeigt, wie aufgrund von lokalen Informationen der Konflikt erkannt wird und eine neue einheitliche Version mit dem Maximum aus den konfliktbehafteten Versionen gebildet wird.

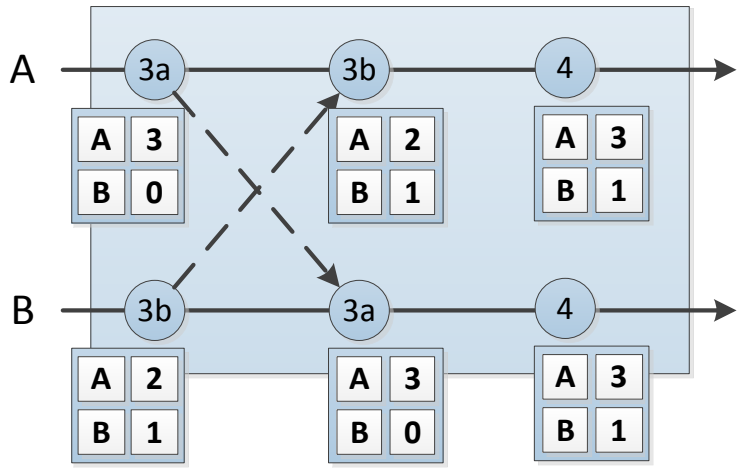


Bild 3.10: Lokale Behandlung von nebenläufigen Änderungskonflikten

Der Hauptnutzen der Akteure in diesem Vorgehen liegt darin, dass automatisch die global zuletzt konfliktfreie Version ermittelt und diese dann als neue gültige Version gesetzt wird. Desweiteren werden die konfliktbehafteten Versionen invalidiert und in einem gesonderten Zweig in der Versionsverwaltung abgelegt, um die Datenprovenienz zu sichern. Im Beispiel aus Abbildung 3.10 entspricht damit der Nutzinhalt von Version *4* dem Nutzinhalt von Version *2*.

Abbildung 3.11 zeigt noch einmal die Änderungshistorie. Der dort abgebildete Systempfad enthält alle durch Änderungen hervorgerufene Versionen. Der gültige Pfad hingegen überspringt die Versionen *3a* und *3b*, da diese konfliktbehaftet sind. Die Änderungen,

die aus den nebenläufigen Änderungskonflikten hervorgehen, werden in der Historie in sog. Konfliktzweigen abgespeichert.

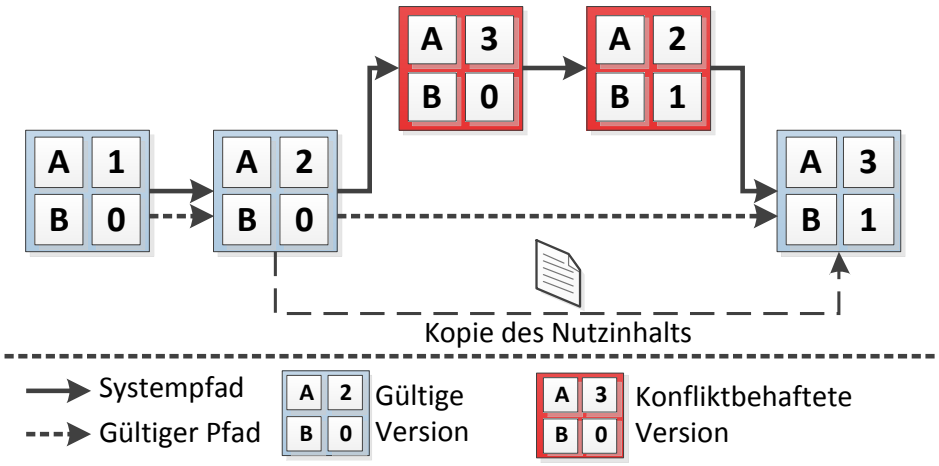


Bild 3.11: Änderungshistorie bei nebenläufigen Änderungen

3.3 α-Flow-System-Architektur

Wie in [Wah11] sowie in [Had11] beschrieben, sind die einzelnen von α-Flow realisierten Funktionalitäten in Subsysteme gekapselt, welche wiederum konzeptionell in Schichten angeordnet werden können. Dabei übernimmt jedes Subsystem eine ganz bestimmte Aufgabe. Das in Abschnitt 3.2 dargestellte Domänenmodell findet sich in dem Modul α-Model wieder. Es bildet die Basis für alle weiteren Module, die es wiederum verwenden.

Die anderen Software-Module lassen sich wie in Abbildung 3.12 dargestellt grob in drei Schichten anordnen:

1. Die Präsentationsschicht umfasst alle Komponenten, die zur Darstellung des α-Doc für den Benutzer und für die Benutzerinteraktion mit diesem nötig sind.
2. Die Logik-Schicht bzw. der Systemkern, die die gesamte Koordinationslogik von α-Flow enthalten und anwenden und das Domänenmodell verwalten.
3. Die Infrastrukturschicht, welche für die Persistenz aller Objekte und deren Versionierung sorgt, sowie die Funktionalitäten für die Kommunikation mit den anderen Akteuren bereit stellt.

Die Präsentationsschicht besteht aus den Komponenten α-Startup, α-Injector und der Komponente α-Editor. α-Startup koordiniert den Startvorgang beim Öffnen eines α-Doc,

indem es alle weiteren benötigten Komponenten initialisiert. Die α-Injector-Komponente stellt die Möglichkeit zur Erzeugung einer neuen α-Card in einem α-Doc zur Verfügung. Bei der Erzeugung eines neuen Dokuments fügt der α-Injector auch die erste leere α-Card hinzu.

Wenn alle Komponenten erfolgreich gestartet wurden, wird der α-Editor initiiert. Durch seine Benutzeroberfläche wird der aktuelle Status des α-Docs dargestellt. Er ermöglicht es dem Benutzer, α-Cards neu anzulegen und bestehende α-Cards mit ihren Adornments anzupassen. Weiterhin kann zu α-Cards Payload hinzugefügt werden. Benutzereingaben im α-Editor werden an α-Properties weitergegeben, wo sie wiederum an die entsprechenden Komponenten bzw. die regelbasierte Bibliothek weitergeleitet werden. Damit stellt die Komponente α-Properties eine zentrale Rolle dar, indem sie für einen großen Teil der Geschäftslogik in Form einer regelbasierten Bibliothek verantwortlich ist. Beim Programmablauf koordiniert diese Komponente die Interaktion der übrigen Komponenten. Jede Änderung an den Adornments wird an α-Adaptive weitergegeben.

In der Infrastrukturschicht gibt es die Module α-VerVarStore, α-Overnet und α-OffSync. α-VerVarStore persistiert und verwaltet die unterschiedlichen Versionen der α-Cards mit der Hilfe eines Versionsverwaltungssystems. Dieses Modul kann eine logische Ordnung auf den Versionen einer α-Card herstellen und nebenläufige Änderungskonflikte auf der Grundlage dieser Versionen lösen. Es stellt weiterhin Schnittstellen zur Verfügung, mit denen man auf das lokale Dateisystem zugreifen kann.

Die Kommunikationsschnittstelle zu anderen α-Docs ist in α-Overnet gekapselt. Dieses Modul bildet ein Interface gegenüber den anderen Komponenten, das von der darunterliegenden Schicht abstrahiert. Eine konkrete Implementierung der Kommunikationsschnittstelle von α-Overnet findet sich in α-OffSync, das offlinefähigen Nachrichtenaustausch bereitstellt. Durch diese Kapselung ließe es sich problemlos durch andere Verfahren austauschen.

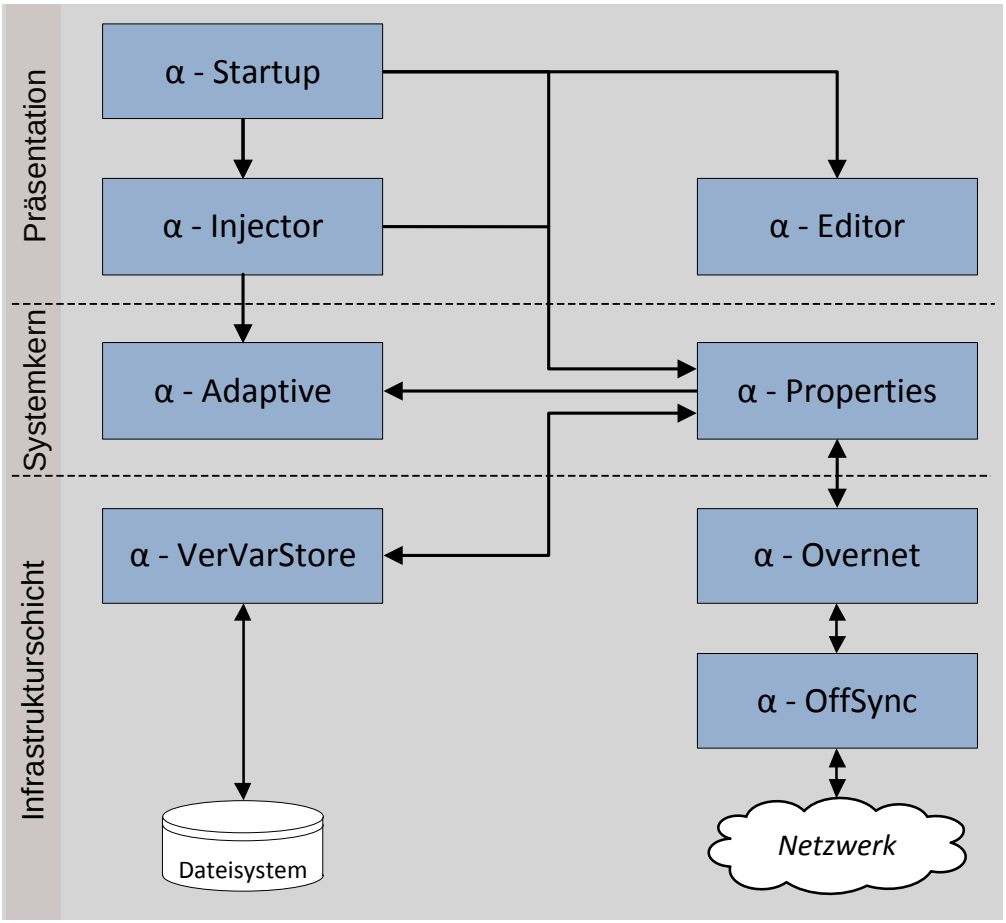


Bild 3.12: Systemarchitektur von α-Flow

3.4 Anwendungsszenario

Dieser Abschnitt soll das Anwendungsszenario von α-Flow beschreiben und greift dazu auf die Ausführungen von [NL10] und [NL12] zurück. Beispielhaft wird der interdisziplinäre Behandlungsprozess bei einer Brustkrebsbehandlung betrachtet. Dieser Anwendungsfall bietet sich an, da er viele zusammenarbeitende Parteien in sich vereint und damit für die Nutzung einer verteilten aktiven Fallakte prädestiniert ist. Zuerst wird der Verlauf der initialen Behandlung vorgestellt, der der Überprüfung gilt, ob der gefundene Knoten in der Brust der Patientin bösartig ist. Im Folgenden soll dann der weitere Vorgang der primären Behandlung dargestellt werden, der abläuft, wenn es sich tatsächlich um Brustkrebs handelt.

Initiale Brustkrebsbehandlung - Klassifikation

Das Ziel der in Abbildung 3.13 gezeigten Behandlungsepisode ist die Überprüfung, ob es sich bei einem Knoten in der Brust um einen bösartigen Tumor handelt.

Die Patientin besucht zuerst ihren niedergelassenen Gynäkologen[1] Gyn^A. Nach einer Anamnese und auf Basis einer Sonographie verweist dieser sie im Verdachtsfall an einen Radiologen Rad^A, um eine Mammographie durchführen zu lassen. Nach der Durchführung der Mammographie sendet der Radiologe seine Ergebnisse zurück an Gyn^A, der diese auswertet und auf Basis von Kennzahlen[2] die Patientin ggf. für eine Biopsie an ein Krankenhaus verweist. Im Krankenhaus entnimmt der Gynäkologe[3] Gyn_B^C die Gewebeprobe und sendet sie an einen Pathologen zur weiteren Untersuchung. Die Histologie des Pathologen schafft nun eindeutige Klarheit. Das Ergebnis der Biopsie wird der Patientin dann wieder von Gyn^A mitgeteilt. Die initiale Behandlung ist an dieser Stellen beendet. Im Falle eines bösartigen Tumors beginnt jetzt die primäre Brustkrebstherapie in einem speziellen Brustkrebsversorgungszentrum.

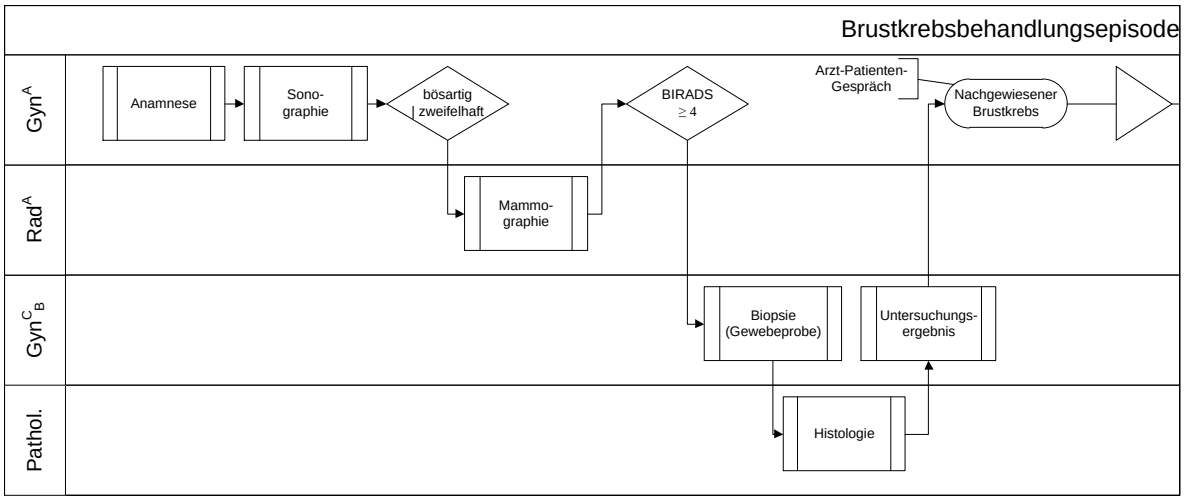


Bild 3.13: Initiale Brustkrebsbehandlungsepisode: Aktivitätenorientierte Sicht ([NL10])

Der beschriebene Prozess kann neben der aktivitätenbasierten Vorgehensweise auch dokumentenorientiert aufgefasst werden. Dazu werden die Aktivitäten nicht selbst dargestellt, sondern vollständig durch ihre Ergebnisse repräsentiert (cf. Abbildung 3.14).

1 „A“ steht für ambulant.
2 Hauptsächlich wird hier der Breast Imaging - Reporting and Data System (BI-RADS)-Wert verwendet.
3 „C“ steht für klinisch, „B“ für Biopsie.

Prinzipiell wird eine Aktivität also erst nach dessen Durchführung dargestellt, doch analog zum Vorgehen im Gesundheitswesen kann auch ein Platzhalter für das spätere Ergebnis der Aktivität abgebildet werden. Dies wäre beispielsweise eine Überweisung oder, um den Prozess einzuleiten, das schriftliche Festhalten eines Termins für eine Untersuchung.

Das erste Dokument im Prozess ist die Anamnese-Dokumentation von Gyn^A. Weiterhin verfasst er auch einen Sonographie-Bericht. Im Verdachtsfall auf Brustkrebs schickt er die Patientin mit einer Überweisung[1] RV_M^I zu einen Radiologen Rad^A. Dieser führt die Mammographie durch und dokumentiert seine Ergebnisse in einem Report. Falls der BI-RADS-Wert erhöht ist erfolgt von Gyn^A eine weitere Überweisung RV_B^I, der die Patientin in ein Krankenhaus zur Biopsie schickt. Die entnommene Gewebeprobe wird mit einer weiteren Überweisung[2] RV_H^I an einen Pathologen zur Histologie geschickt. Am Ende werden die Ergebnisse der Biopsie-Operation und die Ergebnisse des Pathologen zurück an Gyn^A übermittelt.

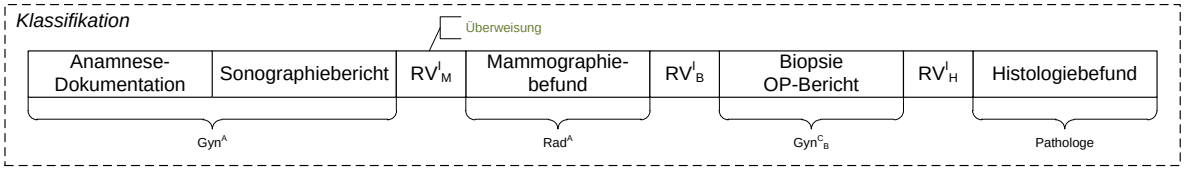


Bild 3.14: Initiale Brustkrebsbehandlungsepisode: Dokumentenorientierte Sicht ([NL10])

Primäre Behandlung

Die Primärbehandlung der Patientin schließt sich an die initiale Behandlung an, falls die Diagnose einen bösartigen Tumor ergab, und erfolgt in einem speziellen Brustkrebsversorgungszentrum. Der Ablauf lässt sich in Abbildung 3.15 verfolgen.

Der Prozess beginnt damit, dass die Patientin in ein Krankenhaus überwiesen wurde (RV_{BC}^H)[3]. Nach der einführenden Anamnese-Aufnahme durch Gyn^C erfolgen drei Untersuchungen. Diese werden unter dem Namen „Staging“ zusammengefasst. Sie umfassen eine weitere Sonographie durch einen Internisten, eine Röntgenaufnahme durch einen Radiologen und eine Szintigraphie durch einen Nuklearmediziner. Alle drei Untersuchungen sind durch eine entsprechende Überweisung (RV_{AS}^H, RV_{PX}^H, RV_{BS}^H)[4] und die

1 „RV“ steht für Überweisung (engl. *referral voucher*), „I“ für Instruktion und „M“ für Mammographie.
2 „H“ steht für Histologie.
3 „H“ steht für Krankenhaus, „BC“ für Brustkrebs.
4 „AS“ steht für Abdomen Sonographie, „PX“ für Röntgen und „BS“ für Szintigraphie.

jeweilige Ergebnisdokumentation festgehalten. Über die Reihenfolge und Vollständigkeit des Staging wird vor Ort entschieden. Die Sonographie und das Röntgen werden prinzipiell immer durchgeführt, die Szintigraphie muss nicht unbedingt stattfinden und kann ggf. auch nach der Operation erfolgen. Vor der Operation wird durch Gyn^C eine Klassifikation gemäß TNM Classification of Malignant Tumours (TNM)[1] durchgeführt, die die Operationsmethode festlegt. Die Operation wird von Gyn^C durchgeführt.

Im Anschluss an die Operation wird das entnommene Gewebe an einen Pathologen übermittelt (RV_H^I). Der Pathologe erstellt daraufhin einen Bericht und eine erneute TNM-Klassifikation.

Alle erstellten Dokumente bilden die Entscheidungsgrundlage für eine im Krebszentrum abgehaltene sog. Tumor-Konferenz. Der Leiter dieser Konferenz ist ein Onkologe, weitere Teilnehmer sind ein Radiologe, ein Pathologe, sowie Gyn^C. Die Konferenz legt in ihrem Verlauf die anschließende Behandlungsabfolge für die Patientin fest. Die weiteren Behandlungsschritte umfassen dabei häufig eine Chemotherapie, eine Bestrahlungstherapie und auch eine Hormontherapie, wobei diese drei Verfahren beliebig kombiniert werden können. Die Ergebnisse der Tumor-Konferenz werden festgehalten und die Primärbehandlung endet mit einen Entlassungsbrief von Gyn^C an Gyn^A. Gyn^A führt auf Basis der in der Primärtherapie erstellten Dokumente die anschließende Behandlung durch.

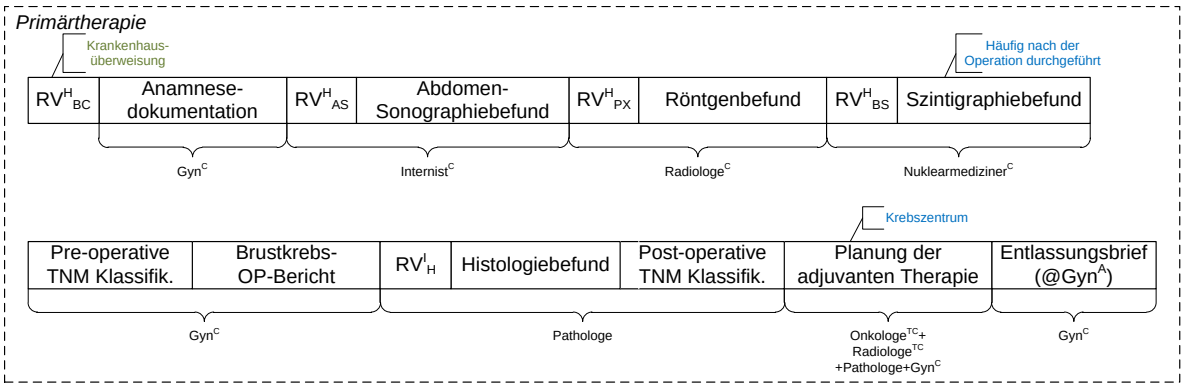


Bild 3.15: Primäre Brustkrebsbehandlungsepisode: Dokumentenorientierte Sicht ([NL10])

1 TNM ist eine Klassifikation, die der Einteilung von bösartigen Tumoren dient und mit deren Hilfe eine Vorhersage über die weitere Behandlung getroffen werden kann [SGW09].

3.5 Zusammenfassung

Zusammenfassend lässt sich α-Flow als ein dokumentenorientierter Ansatz zur Koordinierung interdisziplinärer, institutionsübergreifender Prozesse beschreiben. Das auf aktiven Dokumenten basierende System bildet eine papierbasierte Vorgehensweise digital ab und kann damit den flexiblen, nicht vorhersagbaren Verlauf der Behandlung wiedergeben.

In diesem Kapitel wurde nach einer Einführung in die grundlegenden Konzepte von α-Flow, den Geschäftsprozessen bzw. Workflows und der dokumentenorientierten fallbasierten Vorgehensweise, das Domänenmodell von α-Flow vorgestellt. Dieses besteht hauptsächlich aus dem α-Doc, das die verteilte Fallakte darstellt, und den α-Cards. α-Cards enthalten entweder prozessrelevante Koordinationsinformationen (Coordination-Cards) oder medizinische Nutzinformationen.

Im Rahmen des Domänenmodells wurde ein gesonderter Blick auf das Konzept von α-Adaptive gelegt. Es ermöglicht eine Adaption des Datenmodells zur Laufzeit, indem es sowohl die Prototypbasierte Programmierung als auch das EAV-Modell anwendet. Das Modell findet besondere Erwähnung, weil das im weiteren Verlauf der Arbeit beschriebene Token-Konzept darauf aufbauen wird.

Das durch α-OffSync implementierte Verfahren zur Offline-Synchronisierung basiert auf logischen Zeitstempeln, die es ermöglichen, mit lokal vorliegenden Daten eine global einheitliche Ordnung auf durchgeführten Aktivitäten herzustellen. Auftretende Änderungsanomalien können damit erkannt und behoben werden. Insbesondere bei der Token-Änderung und -Weitergabe kann es zu ähnlichen Schwierigkeiten kommen. Der hier vorgestellte logische Zeitstempel kann, wie später gezeigt wird, auch dort verwendet werden.

Weiterhin wurde die System-Architektur von α-Flow vorgestellt. Dazu gehört die Aufteilung einzelner Module mit bestimmten Funktionalitäten und ihre Anordnung in Schichten. Diese Aufteilung hat insbesondere den Vorteil, dass es zu weniger Abhängigkeiten der Module untereinander kommt und die Software damit leichter zu warten ist.

Abschließend wurde ein Anwendungsszenario skizziert, das die Anwendung von α-Flow verdeutlichen soll. Es handelt sich hierbei um eine interdisziplinäre Brustkrebsbehandlung. Es dient in erster Linie dafür, ein Verständnis für die Nutzung von α-Flow zu schaffen. Des Weiteren bildet es auch die Grundlage für den in Kapitel 5 um Prozessrollen erweiterten Anwendungsfall. Und auch im Rahmen der Evaluation wird auf dieses

Anwendungsszenario zurückgegriffen, weil es beispielhaft das Vorgehen bei einer realen Brustkrebsbehandlung wiedergibt (cf. Kapitel 7).

4 Verwandte Arbeiten

In diesem Kapitel werden verschiedene Ansätze betrachtet, die die Grundlage für Überlegungen in dieser Arbeit liefern und sich z.T. mit ähnlichen Problemen wie diese Arbeit beschäftigen. Im ersten Abschnitt dieses Kapitels werden Ansätze betrachtet, die das Token-Konzept verwenden (cf. Abschnitt 4.1 und 4.2). Daran anschließend werden verwandte Arbeiten zum Thema Empfangsbestätigungen vorgestellt (cf. Abschnitt 4.3).

Zunächst wird die Business Process Model and Notation(BPMN) besprochen. Sie ist ein Standard im Umfeld des Workflow-Managements, um Geschäftsprozesse grafisch darzustellen. An dieser Stelle soll allerdings nur ein kurzer Überblick über die Notation gegeben werden, da im Rahmen dieser Arbeit kein Bezug zur Prozessmodellierung besteht, sondern lediglich einige Element und Konzepte von Bedeutung sind. Die BPMN verwendet das Token-Konzept, um den Kontrollfluss während des Ablaufs einer Workflow-Instanz zu verdeutlichen. Zuerst wird dieses allgemein betrachtet, um dann im Folgenden auf einige Teilaspekte wie das Konstrukt der Pools und Lanes näher einzugehen.

Im Anschluss daran werden Konzepte aus dem Gebiet der Verteilten Systeme untersucht. Insbesondere Verfahren zum gegenseitigen Ausschluss und die Koordinatorwahl sind im Rahmen dieser Arbeit relevant, weil sie auch mit Token arbeiten.

Änderungen an α-Cards werden durch α-Offsync per E-Mail versandt. Daher werden zunächst ausgehend vom Simple Mail Transfer Protocol (SMTP) Erweiterungen zu diesem Protokoll näher betrachtet, die es um verschiedene Empfangsbestätigungsmodelle erweitern. Dies sind zum einen Delivery Status Notification (DSN) und Message Disposition Notification (MDN). Ein weiteres Protokoll, das durch Erweiterungen ebenfalls Acknowledgements unterstützt, ist das Extensible Messaging and Presence Protocol. Seine Empfangsbestätigungen werden im Anschluss erläutert.

4.1 Business Process Model and Notation

Die BPMN ist ein von der Object Management Group (OMG) entwickelter Standard, der eine Möglichkeit zur grafischen Notation von Geschäftsprozessen bietet. Ziel des Standards ist es, zu gewährleisten, dass alle am Prozess beteiligten Personen die Definition des

Prozesses verstehen – vom Geschäftsanalysten über den technischen Entwickler bis hin zum Prozessmanager ([Obj11, Seite 1]). Damit schlägt es eine Brücke vom Prozessentwurf bzw. seiner Abbildung hin zur Prozessimplementierung.

Die OMG definiert in [Obj11] graphische Symbole, deren Bedeutung und wie sie kombiniert werden dürfen, um Prozesse zu modellieren.

4.1.1 Überblick über die Notation

Mit BPMN können verschiedene Diagrammtypen entworfen werden. Die Version 2.0 erlaubt folgende Typen: das Sequenzdiagramm, das Kollaborationsdiagramm, das Choreographiediagramm und das Konversationsdiagramm.

Die größte Bedeutung für die Beschreibung von Prozessabläufen hat das Sequenzdiagramm. In ihm wird modelliert, in welcher Reihenfolge Aktivitäten ausgeführt werden und unter welchen Bedingungen. Die hier verwendeten Kernelemente werden im weiteren Verlauf dieses Kapitels näher beschrieben. Um die Zusammenarbeit von mehreren Prozessen abzubilden und um den Nachrichtenfluss zwischen zwei oder mehr Prozessen abzubilden, wird das Kollaborationsdiagramm verwendet. Choreographiediagramme geben wie Kollaborationsdiagramme die ausgetauschten Nachrichten zwischen Prozessen wieder. Allerdings bewegen sie sich auf einem anderen Abstraktionsniveau. Sie zeigen detaillierter, in welcher zeitlichen Abfolge und in welcher logischen Reihenfolge Nachrichten versandt werden. Konversationsdiagramme stellen ein Übersichtsdiagramm dar. Mittels eines solchen Diagramms werden Gruppen von Nachrichten als sog. *Communications* zusammenfassend abgebildet. Diese Form von Diagramm ist insbesondere für die Kommunikation mit externen Partnern geeignet, weil sie einen guten Überblick über den Prozess gibt.

Die bisher beschriebenen Prozessdiagramme bestehen hauptsächlich aus verschiedenen Symbolen, den sog. Kernelementen.[1] Ihre Anordnung legt fest, was in welcher Reihenfolge unter bestimmten Bedingungen ausgeführt wird. Zu ihnen zählen insbesondere die Flussobjekte wie Aktivitäten, Ereignisse und Gateways. Durch ihre Ausprägung und ihre Anordnung steuern sie das Verhalten eines Geschäftsprozesses.

1 Für eine detailliertere Beschreibung der Kernelemente der BPMN wird auf das Anhangskapitel A verwiesen.

4.1.2 Token-Konzept

Der Sequenzfluss in einem Prozess kann durch das theoretische Konstrukt Token, auch Marken genannt, visualisiert werden (cf. [Obj11, Seite 27] und [FRH10, Seite 23f.]). Der Begriff Token wird verwendet, um sich den in einer Prozessinstanz verwendeten Prozesspfad zu verdeutlichen. Der Sequenzfluss kann dadurch ähnlich der Ablaufsteuerung bei Petri-Netzen erklärt werden. Dort werden Transitionen ausgelöst, wenn genügend Token an den Eingangsstellen vorliegen. Auch in der Unified Modelling Language (UML) wird das Token-Konzept verwendet. Es bildet dort das Verständnismodell hinter Aktivitätsdiagrammen, um deren Ablauf besser nachvollziehen zu können (cf. [RQZ07, Seite 259ff.]).

Beim Starten eines Prozesses wird ein Token am Startereignis erzeugt und an die erste Aktivität geschickt. Ist diese abgearbeitet, wird das Token weitergereicht. Kommt das Token an ein Gateway, wird wiederum dafür gesorgt, dass das Token den richtigen Weg nimmt. Es kann dabei auch vorkommen, dass ein Token dabei kopiert wird und mehrere Wege bestreitet (paralleles Gateway). Dies führt zu einer parallelen Arbeitsweise in den unabhängigen Zweigen. An Gateways, die parallele Prozesspfade wieder zusammenführen, werden die Tokens dann wieder zu einem einzelnen Token verschmolzen und nur dieses wird weitergegeben. Prinzipiell werden also stets die Aktivitäten ausgeführt, an denen ein Token anliegt, und dieses dann weitergereicht. Erreicht das Token das Endereignis, wird es dort vernichtet. Wenn sich kein Token mehr im Umlauf befindet, ist der Prozess beendet. Das Token-Konzept dient hierbei lediglich als Veranschaulichung des Prozessablaufs. In [Obj11, Seite 57] wird darauf hingewiesen, dass eine konkrete BPMN-Implementierung zwar die Token-Semantik abzubilden hat, aber ein Token nicht explizit im Programmcode verwendet werden muss.

4.1.3 Pools und Lanes

Neben der Beschreibung, welche Vorgänge in einem Prozess ablaufen, liefert BPMN auch die Möglichkeit, zu beschreiben, wer einzelne Aufgaben zu erledigen hat. Dazu greift BPMN auf das Konzept der Lanes zurück (cf. [FRH10, Seite 44ff.]). Diese Zuständigkeiten von Personen für bestimmte Aufgaben sind bereits in Abbildung 3.13 zu erkennen. Auch wenn es sich nicht um ein reines BPMN-Diagramm handelt, kann man deutlich die Verteilung der einzelnen Aktionen auf die Ärzte erkennen. Sichtbar ist dies durch die sog. Lanes, die von links nach rechts ähnlich einer „Schwimmbahn“ verlaufen.

Zusammengefasst werden die Lanes zu einem Pool, der die Schwimmbecken-Analogie schließt – ein Schwimmbecken umfasst mehrere Schwimmbahnen (cf. Abbildung 4.1). Im Fall der Abbildung 3.13 ist der Pool die *Breast Cancer Classification Episode* und die einzelnen Ärzte sind durch ihre jeweiligen Lanes dargestellt.

Bild 4.1: Kernelemente der Business Process Model and Notation

Der Pool umfasst den gesamten Prozess von Anfang bis Ende. Pools werden als Container für einen Prozess benutzt und werden insbesondere verwendet, wenn die Zusammenarbeit zwischen mehreren Prozessen dargestellt werden soll (cf. [MW08, Seite 157ff.] und [FRH10, Seite 95ff.]). Dabei können Pools für den Betrachter auch als Blackbox dargestellt werden und nur die Nachrichtenflüsse zwischen den Prozessen werden modeliert, sodass für den Betrachter ein übersichtliches Bild entsteht, das interne Abläufe verbirgt. In diesem Fall spricht man vom bereits genannten Kollaborationsdiagramm.

4.1.4 Rollen im Prozessumfeld

Freund beschreibt im Zusammenhang der Prozessmodellierung mehrere Rollen, die am Projekt Beteiligte einnehmen können (cf. [FRH10, Seite 12ff.]). Zu diesen Rollen zählen u. a. der Process Owner und der Process Manager. Der Prozess Owner (Prozessverantwortliche) hat die strategische Verantwortung für den Prozess. Er interessiert sich für die Optimierung der Prozessleistung und seine Position in der Führungsebene gibt ihm das Recht, Budgetfragen zu klären. Der Prozess Manager (Prozessmanager) hingegen ist operativ für den Prozess verantwortlich und tritt nach außen hin als Auftraggeber auf. Er ist dem Process Owner hierarchisch untergeordnet und hat diesem zu berichten. Insgesamt ist hier aber zu sagen, dass diese Rollen im Sinne von Betrachtern des Prozessmodells und weniger als Prozessteilnehmer verstanden werden. Sie spiegeln zwar im weiteren Sinne die Positionen der in dieser Arbeit zu betrachtenden Wortführer- und Prozessinitiator-Rollen wider, doch legen sie einen anderen Fokus, da es sich nicht um Prozessteilnehmer handelt.

4.2 Token-Konzepte in verteilten Systemen

In diesem Kapitel werden verschiedene Konzepte untersucht, die sich bei verteilten Systemen für den gegenseitigen Ausschluss beim Betreten eines kritischen Abschnitts und die Koordinator-Wahl entwickelt haben. Diese Verfahren sind daher relevant, da sie sich zum einen mit der Problematik der Einmaligkeit (nur ein Akteur darf den kritischen Abschnitt gleichzeitig betreten) und der Wahl eines Anführers beschäftigen, der im Prozess eine Sonderrolle einnimmt. Dabei werden vornehmlich die Verfahren betrachtet, die das Token-Konzept zur Lösung des Problems benutzen.

Ausgehend von einer allgemeinen Definition von verteilten Systemen, die die auftretenden Charakteristika beschreibt, werden zunächst Ansätze für den gegenseitigen Ausschluss und dann Verfahren zur Koordinator-Wahl betrachtet.

4.2.1 Charakteristika verteilter Systeme

Unter einem verteilten System versteht man eine Ansammlung unabhängiger Computer, die von außen wie ein einzelnes System erscheinen ([Tan07, Seite 19]). Für den Benutzer erweckt das System den Eindruck, als wäre es ein einzelner Computer.

Die Computersysteme selbst sind durch ein Netzwerk verbunden und kommunizieren, indem sie Nachrichten verschicken. Insbesondere weisen solche Systeme einige Charakteristika auf, die zu Problemen führen können und gesondert behandelt werden müssen. Zu diesen Charakteristika zählen ([Gar02, Seite 3f.]):

- Das Fehlen einer gemeinsamen Uhr. In einem verteilten System können physische Uhren nicht ohne weiteres synchronisiert werden. Aufgrund von Kommunikationsverzögerungen wird daher häufig auf andere Konzepte zurückgegriffen.
- Das Fehlen gemeinsamen Speichers. Der globale Zustand des Systems muss auf andere Art und Weise ermittelt werden.
- Neue Fehlerarten. In einem verteilten System ist es schwierig, zwischen einem langsamen Prozess und einem abgestürzten Prozess zu unterscheiden. Das Ausbleiben einer Antwort könnte beispielsweise auf einen langsamen Kommunikationspartner hindeuten oder der Kommunikationspartner ist ausgefallen.

Aufgrund dieser Komplexität schreibt Leslie Lamport auch von einem System, bei dem der Ausfall eines Computers, dessen Existenz nicht einmal bekannt war, bewirkt, dass das System selbst unbrauchbar wird ([Lam87]):

> *A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable.*

Bei der Zusammenarbeit müssen sich die Prozesse in irgendeiner Form synchronisieren, z. B. beim Zugriff auf eine gemeinsame Resource. Wenn mehrere Prozesse gleichzeitig auf eine Ressource zugreifen, kann diese Ressource in einen inkonsistenten Zustand gebracht oder beschädigt werden. Im lokalen Fall bietet sich hier die Verwendung von Semaphoren oder Monitoren an. Im verteilten Fall hingegen müssen aufgrund der bereits genannten Charakteristika von verteilten Systemen kompliziertere Ansätze gewählt werden. Lösungen zu diesem Problem bieten die in Abschnitt 4.2.2 beschriebenen Algorithmen, die Prozessen einen gegenseitig ausschließenden Zugriff gewähren sollen.

Außerdem wird in einem verteilten System oftmals ein Koordinatorprozess benötigt. Dabei soll die Wahl auf einen Prozess fallen, wobei es prinzipiell keine Rolle spielt, welcher Prozess der Gewinner ist. Am Ende ist es nur wichtig, dass die Wahl eindeutig ist und jeder teilnehmende Prozess über das Ergebnis der Wahl informiert wird. Bei den vorgestellten Algorithmen wird von einer aufsteigenden Prozessnummerierung ausgegangen, wobei die Prozesse ansonsten identisch sind. Ziel ist es jeweils, den Prozess mit der höchsten Prozessnummer zu finden. Ausgewählte Wahlalgorithmen werden in Abschnitt 4.2.3 vorgestellt.

4.2.2 Gegenseitiger Ausschluss

Bei den Algorithmen zum gegenseitigen Ausschluss unterscheidet man prinzipiell tokenbasierte und berechtigungsbasierte Verfahren.

Die tokenbasierten Verfahren folgen der Annahme, dass man nur in den kritischen Abschnitt eintreten kann, wenn man ein entsprechendes Token besitzt. Im verteilten System existiert genau ein solches Token. Damit ist sichergestellt, dass nur ein Prozess gleichzeitig in den kritischen Abschnitt eintreten kann. Vorteile dieses Verfahrens sind, dass es zu keinen Verklemmungen kommen kann und dass jeder Prozess irgendwann das Token erhält (kein Verhungern). Dem gegenüber haben diese Verfahren aber den Nachteil, dass das Token verloren gehen kann. Falls also der Prozess, der momentan das Token besitzt, ausfällt oder das Token während der Übertragung verloren geht, muss ein neues Token erzeugt werden.

Bei den Verfahren, die den berechtigungsbasierten Ansatz verfolgen, muss ein Prozess, der in den kritischen Abschnitt eintreten will, dazu die Erlaubnis der anderen Prozesse einholen.

Die folgenden Ausführungen basieren in erster Linie auf [Tan07], [Gar02] und [PF05], die die vorgestellten Ansätze detailliert beschreiben.

4.2.2.1 Zentralisierter Koordinator-Ansatz

Beim zentralisierten Ansatz wird das Eintreten in den kritischen Abschnitt durch einen zentralen Prozess, dem Koordinator, überwacht. Wenn ein Prozess auf die gemeinsam genutzte Ressource zugreifen will, dann stellt er zuerst eine Anfrage an den Koordinator. Dieser überprüft, ob sich momentan ein anderer Prozess im kritischen Abschnitt befindet. Falls sich kein anderer Prozess im kritischen Abschnitt befindet, schickt der Koordinator ein Token zurück, das das Eintreten erlaubt. Wenn ein anderer Prozess bereits im kritischen Abschnitt ist, wartet der Koordinator mit einer Antwort, bis der kritische Abschnitt wieder frei ist. Solange verwaltet er die Anfragen in einer Warteschlange, aus der er jeweils bei Freiwerden des kritischen Abschnitts die erste Anfrage entnimmt. Beendet ein Prozess die Ausführung des kritischen Abschnitts, dann sendet er das Token zurück an den Koordinator.[1]

Ein entscheidender Nachteil der Vorgehensweise mit einem zentralen Koordinator ist, dass der Koordinator einen *single point of failure* darstellt. Fällt dieser aus, können keine Prozesse mehr in den kritischen Abschnitt eintreten bzw. werden in ihrer Ausführung blockiert, weil sie auf eine Antwort des Koordinators warten. Des Weiteren kann es an der Stelle des Koordinators zu einem Flaschenhals kommen. Weiterhin kann es zu einem Deadlock kommen, wenn der Prozess, der sich momentan im kritischen Abschnitt befindet, ausfällt.

Der Vorteil hingegen besteht darin, dass dieser Ansatz einfach zu handhaben und implementieren ist. Für den Eintritt in den kritischen Abschnitt sind außerdem nur drei Nachrichten nötig.

4.2.2.2 Verteilter Ansatz

Der verteilte Ansatz soll die in Abschnitt 4.2.2.1 genannten Nachteile aufheben. Dieser Ansatz basiert ursprünglich auf einer Arbeit von Ricart und Agrawala (cf. [RA81] und [Tan07, Seite 285ff.]).

Es wird davon ausgegangen, dass alle Ereignisse im System einer vollständigen Ordnung unterliegen. D. h., für jedes Paar von Ereignissen kann eindeutig gesagt werden, welches

1 Eine ausführliche Beschreibung der Vorgehensweise wird in den Listings im Anhangskapitel B gegeben.

von beiden zuerst eingetreten ist. Um dies zu gewährleisten, werden Lamports logische Uhren benutzt[1].

Will in diesem Ansatz ein Prozess in den kritischen Abschnitt eintreten, dann sendet er eine Nachricht an alle Prozesse, auch an sich selbst. Diese Nachricht enthält den Namen der Ressource, auf die er zugreifen möchte, seine Prozessnummer und die momentane logische Zeit.

Wenn ein Prozess eine solche Request-Nachricht erhält, dann hängt seine Reaktion von seinem eigenen Zustand ab. Dabei können insgesamt drei Fälle unterschieden werden:

- Der Empfänger greift selber nicht auf die gewünschte Ressource zu und möchte dies auch nicht tun:
  → Er sendet eine OK-Nachricht zurück.
- Der Empfänger befindet sich selbst im kritischen Abschnitt:
  → Er antwortet nicht und fügt die Anfrage in eine Warteschlange ein.
- Der Empfänger möchte selbst auch auf die Ressource zugreifen, hat es aber noch nicht getan:
  → Er vergleicht den Zeitstempel seiner eigenen Anfrage mit der eingegangenen. Der niedrigere Zeitstempel erhält den Vorzug. Hat die neue Anfrage den niedrigeren Zeitstempel, sendet er eine OK-Nachricht, ansonsten stellt er die Anfrage in eine Warteschlange.

Abbildung 4.2 zeigt wie zwei Prozesse gleichzeitig auf eine Ressource zugreifen möchten (P_1 und P_2). Beide Prozesse senden einen Broadcast mit ihrem aktuellen Zeitstempel an alle Prozesse auch an sich selbst. Dabei hat Prozess P_1 den niedrigeren Zeitstempel und gewinnt. Wenn er die Abarbeitung des kritischen Abschnitts beendet hat, sendet er ebenfalls eine OK-Nachricht an P_2, damit dieser weitermachen kann.

Wenn ein Prozess von allen anderen eine OK-Nachricht erhalten hat, kann er in den kritischen Abschnitt eintreten. Sobald er diesen verlässt, schickt er allen Prozessen in seiner Warteschlange eine OK-Nachricht und löscht diese anschließend aus der Warteschlange.

Das Problem dieses Ansatzes liegt darin, dass der *single point of failure* aus dem vorherigen Abschnitt durch n Fehlerquellen ersetzt wurde. Denn nun wartet ein Prozess, der in den kritischen Abschnitt eintreten will, auf die Antwort von n Prozessen. Dabei kann er aber nicht zwischen dem willentlichen Ausbleiben einer Antwort und dem Absturz

1 Der Algorithmus von Lamport dient der Uhrensynchronisation und kann dazu benutzt werden, in einem verteilten System eine Ordnung auf Ereignissen herzustellen (cf. [Lam78]).

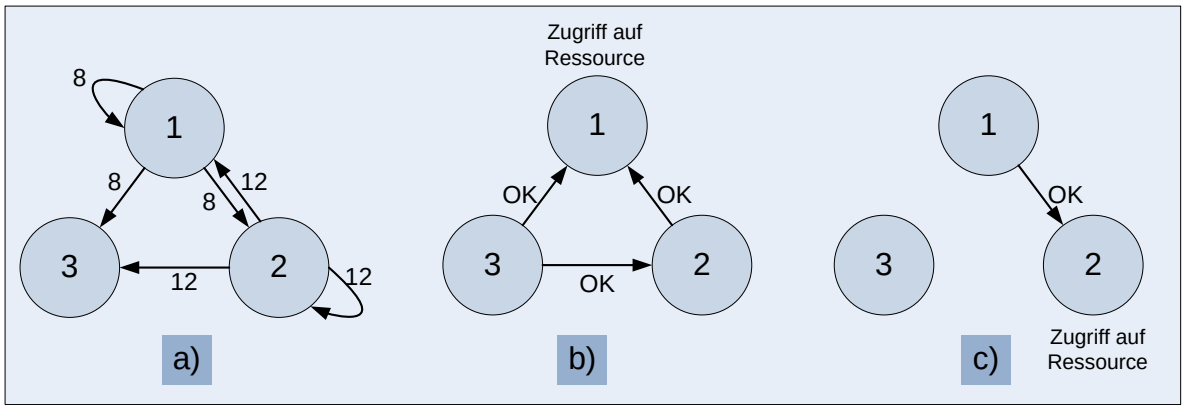


Bild 4.2: Ablauf des verteilten Ansatzes

eines Prozesses unterscheiden. Außerdem ist der Kommunikationsaufwand durch einen Broadcast an alle Teilnehmer höher als im zentralen Fall. Dieser Nachteil kann aber, wie im Folgenden zu sehen ist, durch die Benutzung von Quoren abgemildert werden.

4.2.2.3 Quorenbasierter Ansatz

Der quorenbasierte Algorithmus (cf. [Gar02, Seite 86ff.]) soll dabei helfen, die Anfälligkeit des Systems zu beseitigen, wenn derjenige Prozess ausfällt, der gerade das Token besitzt. Bei diesem Ansatz fragt ein Prozess nicht alle anderen Prozesse, ob er in den kritischen Abschnitt eintreten kann (cf. Abschnitt 4.2.2.2), sondern nur ausreichend viele. Dadurch, dass die Erlaubnis in den kritischen Abschnitt eintreten zu dürfen von einer Untermenge aller Prozesse erbeten wird, sinkt der Kommunikationsaufwand. Diese Untermenge wird im Folgenden *request set* genannt. Da sichergestellt ist, dass alle *request sets* jeweils eine nicht-leere Schnittmenge haben, kann immer nur ein Prozess in den kritischen Abschnitt eintreten. Insgesamt fragt also jeder ausreichend viele Prozesse, ob er eintreten darf. Eine einfache Möglichkeit dieser Vorgehensweise ist das Mehrheitsverfahren, das mindestens ein *request set* von $\lceil N + 1/2 \rceil$ Prozessen benötigt.

4.2.2.4 Token-Ring

Die Grundlage dieses Algorithmus ist die Anordnung der Prozesse auf einem logischen Bus. Auf diesem Bus hat jeder Prozess eine bestimmte Position. Er weiß also, wer sein Nachfolger ist.

In diesem Ring kreist das Zugriffstoken und wird immer von einem Prozess an den nächsten weitergegeben. Erhält ein Prozess das Token, dann prüft er, ob er in den kritischen Abschnitt eintreten will. Ist dies der Fall, tut er es und leitet nach dem

Austreten aus dem kritischen Abschnitt das Token an den nächsten Prozess weiter. Wenn der Prozess nicht in den kritischen Abschnitt eintreten will, dann gibt er das Token direkt weiter.

Ein Nachteil dieses Ansatzes ist es, dass das Token immer umherwandert, auch wenn kein Prozess in den kritischen Abschnitt eintreten will.

4.2.3 Wahlalgorithmen

Wie bereits erwähnt benötigen verteilte Systeme oftmals einen Koordinator-Prozess, der im System eine Sonderrolle einnimmt. Im folgenden Abschnitt sollen zwei solcher Algorithmen vorgestellt werden, die eine Anführerwahl vornehmen.

4.2.3.1 Bully-Algorithmus

Der Bully-Algorithmus stammt ursprünglich von Garcia-Molina ([GM82]). Das Ziel des Algorithmus ist es, aus einer Anzahl an n Prozessen P_1-P_n denjenigen Prozess zu finden, der die höchste Prozessnummer hat und aktiv ist([BBKS08, Seite 342ff.]). D. h. der Koordinator ist P_k *mit* $k = \{i \mid 1 \leq i \leq n, P_i \textit{ ist aktiv}\}$.

Der Algorithmus wird von einem beliebigen Prozess P initiiert. Dieser sendet eine Wahl-Nachricht an alle Prozesse, die eine höhere Prozessnummer haben als er selbst, und wartet ein Zeitintervall T ab. Erhält ein Prozess innerhalb von T keine Antwort, so siegt dieser Prozess und teilt seine Wahl allen anderen Prozessen mit. Wenn ein höherer Prozess antwortet, dann übernimmt dieser die Wahl und P scheidet aus der Wahl aus. Damit beginnt der Algorithmus mit dem höheren Prozess als Initiator von neuem.

Abbildung 4.3 zeigt beispielhaft den Ablauf des Algorithmus bei fünf Teilnehmern. Der Prozess P_5, der frühere Koordinator, ist ausgefallen und P_2 hält eine neue Wahl ab (cf. Abbildung 4.3 a). Dazu sendet er eine Wahlnachricht an die Prozesse P_3 bis P_5. Von den beiden Prozessen P_3 und P_4 erhält er eine Antwort (cf. Abbildung 4.3 b). Weil beide eine höhere Prozessnummer haben, starten diese wiederum selbst eine neue Wahl (cf. Abbildung 4.3 c). P_3 wiederum erhält eine Antwort von P_4 und scheidet damit aus (cf. Abbildung 4.3 d). P_4 erhält keine Antwort und ist damit zum neuen Koordinator gewählt. Das Ergebnis der Wahl teilt er im Anschluss allen anderen Prozessen mit (cf. Abbildung 4.3 e).

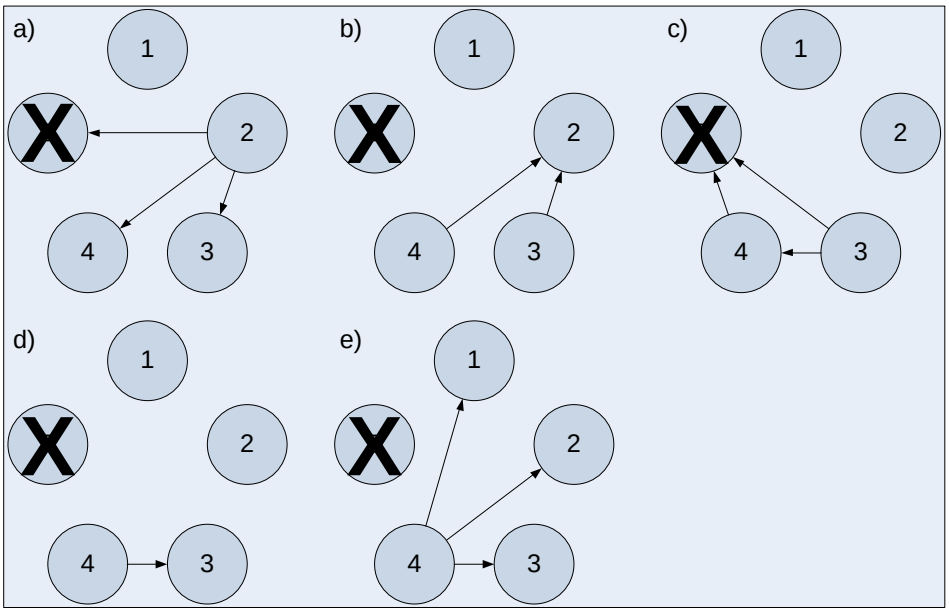


Bild 4.3: Ablauf des Bully-Algorithmus

4.2.3.2 Ring-Algorithmus

Bei diesem Algorithmus sind die Prozesse in einem logischen Ring angeordnet. Jeder Prozess kennt seinen Nachbarn, aber auch die folgenden Prozesse, damit er bei Ausfall seines Nachbarn Nachrichten an den Nachfolger richten kann. Die Wahl wird von einem Prozess gestartet, der eine Wahlnachricht an seinen Nachfolger im Ring sendet. Diese Wahlnachricht wird im Ring weitergeleitet, wobei jeweils vorher die eigene Prozessnummer an eine Liste angefügt wird. Im Laufe der Wahl erweitert sich die Liste also immer und enthält alle Prozessnummern der Prozesse, die bereits an der Wahl teilgenommen haben. Wenn die Wahlnachricht wieder am Start beim initiierenden Prozess ankommt, ist der Prozess mit der größten Prozessnummer zum Koordinator gewählt. Dies wird mit einer weiteren Nachricht, die das Ergebnis der Wahl enthält, bekannt gegeben.

In Abbildung 4.4 ist der vorherige Koordinator ausgefallen und Prozess P_1 leitet eine neue Wahl ein. Die umherwandernde Nachricht wird von jedem Prozess im Ring um die eigene Prozessnummer erweitert. Wenn die Nachricht wieder bei P_1 ankommt, steht fest, dass P_4 neuer Koordinator ist, da er unter den aktiven Prozessen die höchste Prozessnummer hat.

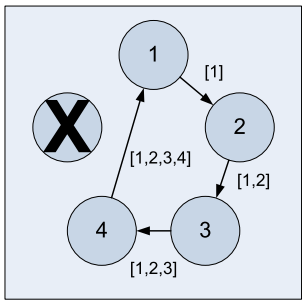


Bild 4.4: Ablauf des Ring-Algorithmus

4.2.4 Zusammenfassung

Es hat sich gezeigt, dass die in verteilten Systemen verwendeten Algorithmen durchaus das Token-Konzept verwenden und Token zur Bestimmung des gegenseitigen Ausschlusses verwenden. Auch die Generierung eines solchen Tokens wird durch die Verfahren zur Koordinator-Bestimmung verwirklicht.

Der Besitzer des Tokens zeichnet sich durch eine Sonderrolle aus, z.B. als ein Koordinator oder für die Synchronisation eines kritischen Abschnitts. Die bei verteilten Systemen verwendeten Algorithmen für den gegenseitigen Ausschluss stellen sicher, dass immer nur ein Prozess gleichzeitig auf eine gemeinsame Ressource zugreifen kann. Sehr einfach lässt sich erreichen, wenn ein zentraler Koordinator verwendet wird, der kontrolliert, wer am Zug ist.

Es gibt auch verteilte Ansätze, die auf diesen Flaschenhals verzichten, den der Koordinator in einem solchen System verkörpert, aber damit einhergehend neue Probleme mit sich bringen. So kommt es zu einem höheren Kommunikationsaufwand und der Algorithmus ist anfälliger gegenüber Prozess- und Kommunikationsausfällen.

Eine geeignete Abschwächung dieser Probleme bietet der quorenbasierte Ansatz, bei dem nicht alle Prozesse, sondern nur ausreichend viele Prozesse gefragt werden müssen.

Die herausragende Rolle des Koordinators muss in bestimmten Fällen neu bestimmt werden, z. B. wenn der Koordinatorprozess ausgefallen ist. Die zwei vorgestellten Wahlalgorithmen können dazu verwendet werden.

4.3 Empfangsbestätigungen

Prinzipiell sollen Empfangsbestätigungen einen Nachweis liefern, dass eine Nachricht beim Empfänger angekommen ist. Sie sollen die Zuverlässigkeit des Datenaustauschs per E-Mail erhöhen.

Eine einfache Form dieser Empfangsbestätigungen findet sich bei Kommunikationsverfahren der Datenschicht. Im ARPANET[1] beispielsweise wird jeder Empfang einer Nachricht durch ein Paket beantwortet, das sog. *Request For Next Message (RFNM).* Dieses Paket gab an, dass die Nachricht beim Empfänger eingetroffen ist. Und mit dem Erhalt des *RFNM*-Pakets wurde gleichzeitig das Übertragungsmedium für die nächste zu übertragende Nachricht freigegeben (cf. [MCC+72, Seite 472] und [HKO+70, Seite 553]).

Prinzipiell können Empfangsbestätigungen nach dem Ort ihrer Generierung unterschieden werden (cf. [BMM90]):

- **lokal**: Das Übertragungssystem nimmt die zu sendende Nachricht entgegen und generiert eine Empfangsbestätigung.
- **entfernt**: Die zu sendende Nachricht kommt beim Empfänger-Server an, der den Erhalt mit einer Empfangsbestätigung quittiert.
- **Benutzer-zu-Benutzer**: Die zu sendende Nachricht wird vom Empfänger wahrgenommen. Dieser versendet daraufhin eine Empfangsbestätigung.[2]

Es hat sich jedoch gezeigt, dass eine Auskunft, ob eine Nachricht beim Empfängerhost eingetroffen ist, weniger Bedeutung hat als die Information, ob der Empfänger darauf reagiert hat ([SRC84, Seite 282]). Es ist also vielmehr eine Ende-zu-Ende-Empfangsbestätigung wünschenswert. Auf die Problematik der verschiedenen Arten von Empfangsbestätigungen wurde an verschiedenen Stellen in der Forschung eingegangen.

Im Folgenden sollen die Ansätze im Rahmen von SMTP und Extensible Messaging and Presence Protocol (XMPP) näher betrachtet werden.

4.3.1 Empfangsbestätigungen in SMTP

Die Zustellung einer E-Mail ist vergleichbar mit der Zustellung einer gewöhnlichen Postkarte, betrachtet man Sicherheitsfunktionen und Beweiskraft (cf. [Tau11]). Bekannte E-Mail-Erweiterungen wie etwa PGP[3] fügen E-Mail-Sicherheitsfunktionen für die Au-

1 Das ARPANET war ein Forschungsprojekt in den USA in um 1970, das als Vorläufer des Internet gilt. In seinem Rahmen wurden Technologien wie die Paketvermittlung und eine dezentrale heterogene Netztopologie getestet (cf. [Abb94] und [FP78]).

2 In diesem Fall entspricht die Empfangsbestätigung auch einer Lesebestätigung des Empfängers, da er die Nachricht wahrgenommen hat.

3 Pretty Good Privacy (PGP) ist ein auf dem Public-Key-Verfahren basierendes Verschlüsselungsprogramm, das die Authentizität, Integrität und Vertraulichkeit von Daten sicherstellt.

thentizität, Integrität und Vertraulichkeit hinzu. Sie bilden somit quasi das Kuvert für den Brief.

Auf der Seite der Beweiskraft wird SMTP[1] um Empfangsbestätigungen (DSN) und Lesebestätigungen (MDN) erweitert (cf. [Moo03], [HV04], [NM08] und [HNM12]). Sie sollen sicherstellen, dass Nachrichten beim Empfänger eintreffen und Unsicherheiten und Fehler, die auf dem Übertragungsweg auftreten können, vermeiden.

Prinzipiell können verschiedene Optionen von Empfangsbestätigungen unterschieden werden je nachdem an welcher Stelle im Verlauf des E-Mail-Versands diese erstellt werden. Erstellt werden diese auch nur, wenn sie vom Sender der Nachricht explizit angefordert werden. Dies vermeidet insbesondere Nachrichtenschleifen bei Empfangsbestätigungen.

Delivery Status Notification

Der Sender einer Nachricht kann eine DSN vom Empfänger anfordern. Dabei kann er angeben, ob er eine Nachricht bei erfolgreichem Versand und auch bei Fehlern erhalten möchte. Diese Empfangsbestätigung enthält dann u.a. Informationen über den ursprünglichen Empfänger der Nachricht, die ursprüngliche Nachricht selbst oder einen Teil davon und die E-Mail-Adresse des Erstellers der DSN.

Damit DSNs generiert werden können, ist es wichtig, dass die empfangenden oder weiterleitenden SMTP-Server diese Erweiterung unterstützen (cf. [Mik09]). Eine DSN kann daher je nach Funktionalität und Konfiguration der einzelnen Server an verschiedenen Stellen im Sendeprozess generiert werden (cf. Abbildung 4.5):

- **DSN Ende-zu-Ende**: Wenn DSN an allen Mail Transfer Agent (MTA)s implementiert ist, wird eine Empfangsbestätigung verschickt, die bestätigt, dass die Nachricht beim Receiver-MTA eingegangen ist. Der Absender der Nachricht erhält dann diese Bestätigung per E-Mail, und weiß damit, dass die Nachricht beim Empfänger-SMTP-Server angekommen ist. Er erhält allerdings keine Information darüber, ob der Empfänger die Nachricht abgerufen hat.
- **DSN teilweise**: Wenn zwar am Relay-MTA aber nicht am Receiver-MTA DSN implementiert bzw. aktiviert ist, dann wird nur eine Empfangsbestätigung verschickt, die bestätigt, dass die Nachricht an den Receiver-MTA weitergeleitet wurde.

1 Ausführliche Informationen über E-Mail im Internet im Allgemeinen und SMTP im Speziellen liefert das Anhangskapitel C.

- **DSN nicht implementiert**: Wenn kein DSN implementiert wurde, dann weiß der Sender-MUA nur, dass seine Nachricht beim Relay-MTA für die Weiterleitung angenommen wurde.

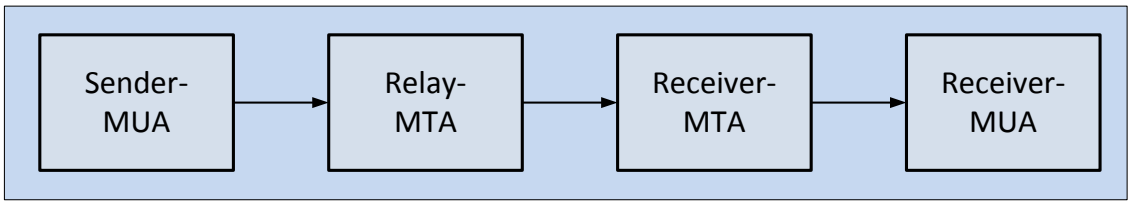


Bild 4.5: Unterschied Empfangsbestätigungen

Genauso erhält der Absender eine Nachricht, wenn die Zustellung der E-Mail aus irgendeinem Grund fehlgeschlagen ist, z.B. wenn die E-Mail-Adresse des Empfängers nicht existiert oder wenn die Disk-Quota des Empfängers ausgeschöpft ist.

Der Versand von Delivery Status Notification erfolgt auf dem Vertrauen, dass der Empfänger diese auch wirklich versendet. Außerdem müssen wie bereits erwähnt, bei einem Fehler während des Weiterleitens der Nachricht alle MTAs so konfiguriert sein, dass sie DSNs nicht verwerfen. Installierte Spam-Filter können ebenfalls dafür sorgen, dass ordnungsgemäß zugestellte DSNs nicht beim Empfänger im User Agent angezeigt werden.

Message Disposition Notification

Durch eine Mail Delivery Agent (MDA) kann der Sender einer Nachricht das Öffnen der Nachricht durch den Empfänger bestätigen lassen. Sie entspricht damit einer Lesebestätigung (cf. [Wie01, Seite 11f.]). Angefordert wird sie, indem der Sender ein bestimmtes Header-Feld in der ursprünglichen Nachricht mit seiner E-Mail-Adresse füllt.

Wenn der Receiver-MUA MDN implementiert hat, dann kann eine Lesebestätigung im Sendeprozess generiert werden, wenn die Nachricht heruntergeladen und angezeigt wird (cf. Abbildung 4.5). Allerdings kann der Empfänger die Generierung von MDAs auch unterbinden. Der Sender einer Nachricht ist also auf die Kooperation des Empfängers angewiesen, dass dieser das Versenden von MDNs erlaubt bzw. diese manuell generiert.

Die Benutzung von MDN hat den Vorteil, dass die zwischengeschalteten MTAs über dessen Existenz nicht informiert zu werden brauchen und man damit nicht von ihnen abhängig ist. Auf der anderen Seite gibt es im Rahmen der MDNs Sicherheitsbedenken, dass diese ohne die Zustimmung des Empfängers verschickt werden und Informationen über diesen beispielsweise an eine Mailing-Liste preisgeben (cf. [Kly98]).

4.3.2 Extensible Messaging and Presence Protocol

Das XMPP ist eine offene Technologie für Echtzeit-Kommunikation. Sie basiert auf der XML als Format, um Informationen nahezu unter Echtzeit-Bedingungen zwischen zwei oder mehr Teilnehmern auszutauschen (cf. [SA11a], [SA11b] und [SAST09]).

XMPP wird in erster Linie für Instant Messaging verwendet und bietet Funktionen zur Nachrichten- und Dateiübermittlung und für Konferenzen mit mehreren Benutzern. Dabei benutzt es eine dezentralisierte Client-Server-Architektur, die ähnlich der E-Mail-Architektur ist. Um am Netzwerk teilzunehmen, müssen sich Benutzer bei einem Server anmelden und können dann ihre Nachrichten in Form von kleinen Dateneinheiten (*stanzas*)[1] übertragen.

Der entscheidende Unterschied zur E-Mail-Architektur liegt jedoch darin, wie Nachrichten versendet werden. Wenn ein Benutzer in XMPP eine Nachricht an einen entfernten Benutzer verschicken will, dann verbindet er sich mit seinem Server und überträgt die Nachricht an diesen. Bis hierher ist der Ablauf analog zu E-Mail (cf. Abschnitt C.0.1. Im nächsten Schritt verbindet sich der XMPP-Server allerdings direkt mit dem Server des anderen Benutzers. Das entfallen von Zwischenschritten wie es beim E-Mail-Versand häufig üblich ist, hat Vorteile im Rahmen von *Address Spoofing*[2] und Formen von Spam (cf. [SAST09]).

Stream Management

Bei XMPP gibt es eine Reihe von Komponenten deren Ausfall die Zuverlässigkeit des Systems senken würde. Dazu gehören die eingesetzten Server und Clients aber auch die Verbindungen untereinander.

In der Standard-XMPP-Konfiguration werden XMPP-Nachrichten zwischen Client und Server und zwischen zwei Servern unbestätigt über Transmission Control Protocol (TCP)-Streams verschickt. Dieser Ansatz erhöht die Effizienz und ermöglicht einen schnellen Versand der Nachrichten. Aber gesendete XMPP-Nachrichten können bei diesem Vorgehen

1 XML-*stanzas* können als kleinste Dateneinheit verstanden werden. Sie sind ähnlich wie Pakete oder Nachrichten in anderen Kommunikationsprotokollen. Neben dem grundlegenden Typ *message* für einfache Informationsübertragung exitiert beispielsweise der Typ *presence*, der den Onlinestatus einzelner Benutzer übermittelt. Des Weiteren ist es möglich *stanzas* mit einem Payload zu erweitern, das dem entfernten Benutzer angezeigt werden soll (cf. [Seite 18ff.][SAST09]).

2 *Address Spoofing* bezeichnet den Vorgang von IP-Paketen mit gefälschten IP-Adressen mit dem Hintergedanken die Identität des Senders zu verbergen und sich als jemand anderes auszugeben (cf. [Tan03]).

auf Applikationsebene unbemerkt verloren gehen, wenn die XMPP-Verbindung abbricht oder ein teilnehmender Rechner abstürzt.

Das Problem von verlorenen Nachrichten, über deren Verbleib der Sender nicht informiert wird, wird durch die Erweiterung des Stream-Managements gelöst (cf. [KSAH+11]). Sie bietet erweiterte Funktionalität im Rahmen von XML-Streams zwischen zwei Entities. Dazu gehören auch *stanza*-Acknowledgements und die Möglichkeit, beendete Streams schnell wieder aufzunehmen.

Acknowledgements können für einzelne Nachrichten und auch für eine Reihen von *stanzas* angefordert werden. Die Abbildung 4.6 gibt einen Überblick über den Ablauf. Es werden eine Reihe von Nachrichten über den aufgebauten Stream geschickt. Die dritte Nachricht (roter Rahmen) enthält eine Aufforderung an den Empfänger, eine Empfangsbestätigung zu verschicken. Daraufhin verschickt der Empfänger umgehend die gewünschte Empfangsbestätigung. Diese enthält dabei immer einen Verweis auf das zuletzt verarbeitete *stanza*. Einem Server ist es im Rahmen der Erweiterung auch freigestellt, unaufgefordert Acknowledgements zu verschicken.

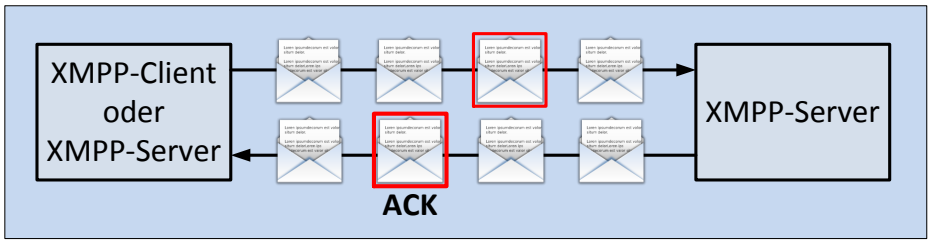


Bild 4.6: Acknowledgements im Stream Management bei XMPP

Kommt es zu einem unerwarteten Verbindungsabbruch eines XML-Streams, dann kann durch das Protokoll des Stream-Managements der beendete Stream zügig wieder aufgenommen werden. Dazu werden die Sequenznummern der zuletzt erhaltenen *stanzas* ausgetauscht. Damit ist für die Teilnehmer sichergestellt, welche *stanzas* wiederholt versendet werden müssen und welche nicht.

Message Delivery Receipts

Um eine Ende-zu-Ende-Zustellgarantie zu erhalten werden entsprechend andere Acknowledgements benötigt (cf. [SRC84]). In XMPP werden dabei prinzipiell Empfangsbestätigungen (*Delivery Reports)* und Lesebestätigungen (*Read Receipt*) unterschieden, doch wird diese Unterscheidung auf Protokoll- und Implementierungsebene nicht getroffen. Vielmehr wird durch [SAH11] XMPP um nur Empfangsbestätigungen erweitert, die den

Erhalt von Nachrichten auf Client-Seite bestätigen. Eine Lesebestätigung ist nicht Teil des Protokolls.

Ein Sender einer Nachricht kann den Empfänger auffordern, den Empfang dieser Nachricht durch Versand eines Acknowledgements zu bestätigen. Allerdings gewährt diese Erweiterung keine absolute Garantie darüber, dass Nachrichten auch angekommen bzw. auf dem Weg verloren gegangen sind. Dies hat mehrere Gründe: beispielsweise kann eine Acknowledgement-Nachricht beim Versand verloren gehen, der empfangende Client diese Erweiterung nicht unterstützen oder der Client bei der Generierung eines Acknowledgements abstürzen. Dennoch erhöht es insgesamt die Funktionalität, da man bei Erhalt eines Acknowledgements sicher sein kann, dass die ursprüngliche Nachricht angekommen ist.

4.4 Zusammenfassung

Der Begriff Token hat verschiedene Bedeutungen. Zum einen wird er dazu verwendet, einen Prozessfluss zu beschreiben (cf. Abschnitt 4.1.2). Das theoretische Konstrukt des Token traversiert durch den Prozess und interagiert dabei mit den Prozesselementen. Es stellt damit den Kontrollfluss während des Ablaufs einer Workflow-Instanz dar. An der Stelle, an der sich das Token aufhält, befindet sich momentan der Prozess.

Zum anderen wird das Token-Konzept auf der Kommunikationsebene gebraucht, um Teilnehmer mit einer Sonderrolle bzw. mit bestimmten Rechten auszustatten. Wie in Abschnitt 4.2.2.4 kann es dazu verwendet werden, den Zugriff auf einen kritischen Abschnitt, beispielsweise eine gemeinsam genutzte Ressource, zu synchronisieren. Einige Ansätze benutzen dazu ein Token, um diesen einen Prozess mit der Eigenschaft des Zugriffsrechts auszustatten. Nur der Besitzer des Tokens darf auf die Ressource zugreifen.

Insbesondere in Bezug auf den Token-Ring-Algorithmus (cf. Abschnitt 4.2.2.4) kann ein Token auch als „Sprechstab“ betrachtet werden, der es dem momentanen Besitzer erlaubt, ein Übertragungsmedium zu nutzen (cf. [BCJ+81] und [DSM83]) bzw. in einen kritischen Abschnitt einzutreten.

Falls ein Koordinator, der ebenfalls eine Sonderrolle einnimmt, ausfällt, muss dieser neu bestimmt werden. Die vorgestellten Wahlalgorithmen könnten beispielsweise auch sicherstellen, dass bei Verlust des Token nur genau ein neues Token generiert wird.

Das Ziel dieser Arbeit ist es, ebenfalls bestimmte Teilnehmer mit Hilfe von Token eine gesonderte Rolle im Prozess zuzuweisen. Allerdings soll diese Wahl auf einen Teilnehmer außerhalb vom α-Doc stattfinden, also auf einer fachlichen Ebene. Die nach

fachlichen Entscheidung stattfindende Weitergabe des Token soll dann aber sehr wohl im α-Doc abgebildet werden. Außerdem ist es nicht Bestandteil der Anwendung das Token selbstständig weiterzugeben. Dies muss ebenfalls von einem Benutzer angestoßen werden.

Im Rahmen der Betrachtung von Empfangsbestätigungen wurden zunächst die Erweiterungen von SMTP DSN und MDN vorgestellt. Diese beiden stellen Erweiterungen zum gewöhnlichen E-Mail-Versand dar, indem sie das Generieren von Empfangsbestätigungen erlauben. Damit wird die Aussagekraft beim Versand erhöht. Es kann allerdings nicht ausgeschlossen werden, dass eine Nachricht beim Ausbleiben einer Bestätigung trotzdem angekommen ist. Gründe für ein Ausbleiben können das Nicht-Unterstützen dieser Funktionalität eines Teilnehmers auf dem Sendeweg wie auch im Fall von MDN das Verweigern der Lesebestätigung durch den Empfänger sein. Der Sender ist in diesem Fall auf das Wohlwollen des Empfängers angewiesen.

Auch bei XMPP werden Empfangsbestätigungen angeboten. Auf der einen Seite gibt es dort durch das Stream Management Acknowledgements, die den Empfang von XML *stanzas* auf Serverseite bestätigen. Auf der anderen Seite werden *Message Delivery Receipts* als eine Bestätigung des Empfänger-Clients verschickt.

Die in dieser Arbeit eingeführten Empfangsbestätigungen unterscheiden sich von DSN, MDN und Stream Management Acknowledgements darin, dass sie prinzipiell von der Kooperation des Empfängers ausgehen. Das heißt, es besteht die positive Erwartung, dass der Empfänger einer Nachricht auch wirklich Empfangsbestätigungen generiert und versendet. Eine Zustellung der Empfangsbestätigung kann allerdings dennoch nicht garantiert werden, weil der Versand per E-Mail erfolgt und diese kein zuverlässiges Kommunikationsmedium darstellt. Das Prinzip der Empfangsbestätigungen lässt sich also analog auf α-Flow übertragen, wobei dieses ansonsten unabhängig vom Kommunikationskanal ist.

5 Prozessrollen und deren Weitergabe als Token

Der Token-Begriff wird zum einen dazu verwendet, einen Prozessfluss zu beschreiben (cf. Abschnitt 4.1.2). Ein Token wandert dabei durch den Prozess und gibt durch seine Position den aktuellen Stand der Instanz an.

Zum anderen wird der Begriff auf Kommunikationsebene gebraucht. Teilnehmer mit einer Sonderrolle oder besonderen Rechten werden mit einem Token ausgestattet. So wird der Eintritt in einen kritischen Abschnitt beispielsweise in einigen Algorithmen durch Token synchronisiert (cf. 4.2.2).

In Bezug auf diese Arbeit hat der Begriff allerdings eine etwas andere Bedeutung. Ein Token zeichnet im Falle des Doyen den besitzenden Akteur mit einer bestimmten Sonderrolle aus, die weitergegeben werden kann. Betrachtet man das Patientenkontakt-Token und das Prozessinitiator-Token ist es vielmehr eine Auszeichnung, die den Akteur näher beschreibt. Das Prozessinitiator-Token wird einmalig beim Prozessstart vergeben und kann dann im weiteren Verlauf nicht weitergegeben werden. Das Patientenkontakt-Token wird im eigentlichen Sinn ebenfalls nicht weitergegeben: Ein Prozessteilnehmer kann selbst bestimmen, ob er es besitzen möchte oder nicht. Im Allgemeinen ist ein Token also vielmehr eine Prozessrollenbezeichnung (*Process Role Label*), die teilweise weitergegeben werden können.[1]

Im Folgenden wird die Motivation für die Einführung einer solchen Auszeichnung dargelegt, um darauf aufbauend einen Anwendungsfall zu konkretisieren und dann Anforderungen daraus abzuleiten. Daran schließt die Beschreibung der fachlichen Umsetzung an, welche auch die Weitergabe einzelner Token einschließt.

1 Im weiteren Verlauf wird weiterhin der Begriff Token verwendet. Damit ist dann immer die Bezeichnung der Prozessrolle gemeint.

5.1 Motivation

Bei einem α-Doc handelt es sich wie bereits beschrieben um die Abbildung eines Behandlungsprozesses. In diesem können verschiedene Prozessrollen identifiziert werden. Die durch die Erstellung der Fallakte festgelegte Rolle des Prozessinitiators wird fix vergeben. Ein Wortführer fungiert als Ansprechpartner für die anderen Akteure und dirigiert die weitere Behandlung. Seine Rolle kann weitergegeben werden. Der Patientenkontakt ist eine Eigenschaft, die einem beteiligtem Arzt zugeschrieben werden kann. Diese Sonderrolle ist im Prozessverlauf ebenfalls veränderbar.

Die hier beschriebenen Prozessrollen stellt Abbildung 5.1[1] dar und versieht sie jeweils mit einem Symbol, das ihren Charakter wiederspiegeln soll und sowohl in der Implementierung als auch in Abbildung 5.2 verwendet wird. Für das Initiator-Token wurde ein Zauberstab-Symbol gewählt, da der Initiator der Behandlungsepisode das α-Doc erstellt, mit dem die weitere Behandlung koordiniert wird. Das Symbol des Sterns für den Doyen des Prozesses stellt den wegweisenden Charakter des Akteurs dar, der die Rolle des Wortführers trägt. Ein Akteur mit Patientenkontakt wird durch das Spritzen-Symbol charakterisiert.

Bild 5.1: Identifizierte Prozessrollen und ihre Ikonisierung

Nutzen eines Token-Modells für Prozessbeteiligte

Der Nutzen eines Token-Modells im Verlauf des Prozesses besteht darin, dass jederzeit einsehbar ist, wer momentan welche Rolle im Prozess einnimmt. Den teilnehmenden Ärzten ist es damit leicht möglich, den entsprechenden Ansprechpartner im Falle eines Problems oder für eine weitere Koordinierung zu finden. Genauso ist für jeden Beteiligten selbst immer ersichtlich, welche Rolle er im Prozess gerade einnimmt, und welches Aufgabenfeld er im Blick haben muss. So ist jederzeit sichergestellt, dass jeder Arzt für

1 Die verwendeten Symbole stehen unter einer freien Lizenz und können unter [Isr09] abgerufen werden.

den ihm zugewiesenen Teil der Behandlung erkennbar verantwortlich ist und es zu einem Fortgang der Behandlung kommt.

5.2 Anwendungsfall

Das in Abschnitt 3.4 beschriebene Anwendungsszenario einer Brustkrebsbehandlung bildet die Grundlage für die im Rahmen der Einführung von Token benutzten Anwendungsfälle. Die Benutzung der Token soll nun anhand dieses Szenarios beschrieben werden.

Der an der initialen Behandlung beteiligte Gynäkologe Gyn^A erstellt das α-Doc für die Patientin. Damit ist er der Prozessinitiator. Während der weiteren initialen Behandlung ist er zudem Doyen und besitzt auch das Token „Patientenkontakt“. Für alle weiteren zum Prozess eingeladenen Teilnehmer ist er der initiale Ansprechpartner. Er koordiniert die Behandlung und überweist die Patientin an die jeweiligen Spezialisten.

Sollte der Fall eintreten, dass sich der Tumor als bösartig herausstellt, wird die primäre Behandlung an die initiale Behandlung angeschlossen. Wie bereits erwähnt findet diese in einem speziellen Krankenhaus statt. Dort übernimmt der Gynäkologe Gyn^C die weitere Behandlung. Im α-Doc wird dies durch die Weitergabe des Doyen-Token von Gyn^A an Gyn^C abgebildet. Für alle Prozessbeteiligten ist damit ersichtlich, dass die Rolle des Wortführers und Prozessverantwortlichen gewechselt hat.

Die Rolle des Wortführers wechselt ein weiteres Mal, wenn die Tumor-Konferenz abgehalten wird. Dort übernimmt ein Onkologe die Verantwortung. Sobald eine Übereinkunft über den weiteren Behandlungsablauf gefällt wurde, begibt sich die Patientin wieder in die Behandlung von Gyn^A, der auch wieder das Doyen-Token erhält.

Genau diesen Ablauf verdeutlicht Abbildung 5.2. Zu Beginn hat Gyn^A bei der Dokumentenerzeugung alle Tokens. Die beiden anderen Ärzte werden im weiteren Verlauf der Behandlung zum α-Doc eingeladen und besitzen initial lediglich das Patientenkontakt-Token. Sobald die Primär-Therapie angestoßen wird, wandert das Doyen-Token zu Gyn^C, für die Tumor-Konferenz weiter zu Onk, um abschließend wieder zu Gyn^A zu gelangen.

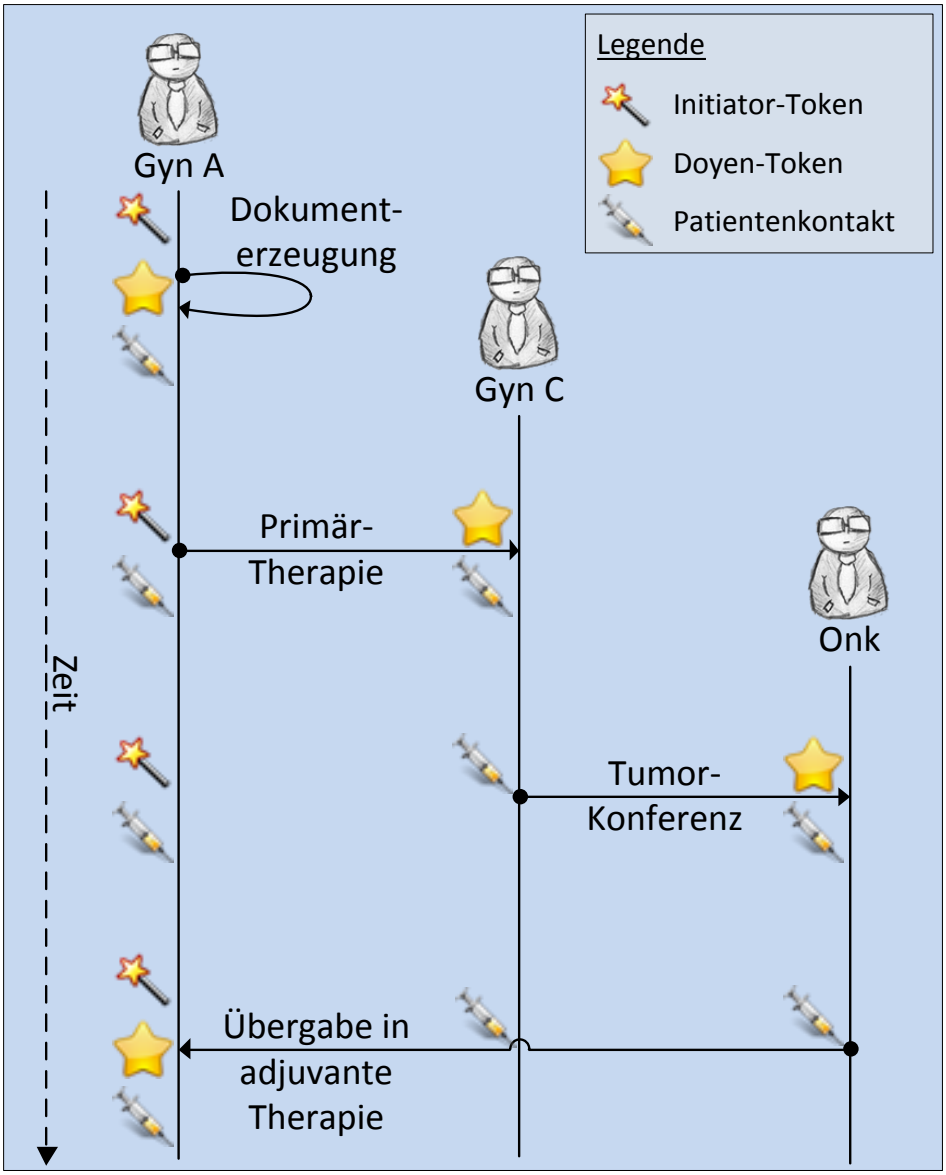


Bild 5.2: Anwendungsszenario Tokenweitergabe

5.3 Anforderungsanalyse

Dieses Kapitel befasst sich mit den funktionalen Anforderungen an die Token-Benutzung. Dazu muss in erster Linie das Domänenmodell angepasst werden. Für eine geeignete Anzeigekomponente wird der Editor angepasst, damit er das erweiterte Domänenmodell entsprechend darstellen kann.

5.3.1 Entwurf eines Token-Modells

Die Token gehören jeweils zu einem Prozessteilnehmer. Prinzipiell kann jeder Teilnehmer die drei Standard-Token besitzen. Als Standard-Token sind vorgesehen:

1. Initiator (α-Doc-Ersteller),
2. Doyen (Wortführer des Prozesses) und
3. Patientenkontakt (Indikator, ob Kontakt zum Patienten besteht).

Dabei kommt das Initiator- und das Doyen-Token jeweils nur einmal im α-Doc vor, weil es im Behandlungsprozess nur einen Prozessinitiator und nur einen Wortführer geben kann. Das Patientenkontakt-Token wird jedem Prozessteilnehmer zugewiesen, der Kontakt zum Patienten hat. Deswegen kann es prinzipiell genauso oft vorkommen, wie es Prozessteilnehmer gibt.

Auf Basis dieser drei Token muss das Token-Modell die im Folgenden näher beschriebenen funktionalen Anforderungen besitzen.

Erweiterbarkeit des Token-Modells

Es ist im Interesse des Prozessteilnehmers, im Behandlungsverlauf eigene Prozessrollen zu definieren. Daher soll es das Token-Modell prinzipiell zur Laufzeit ermöglichen, neue Token hinzuzufügen, Token zu bearbeiten und Token zu löschen. Die Basis des Datenmodells soll also ähnlich wie α-Adaptive adaptiv-evolutionär sein.

Unterscheidung zwischen Token mit Selbstbezug und mit Fremdbezug

Wie bereits erwähnt trifft der ursprüngliche Begriff des Token nur im Fall des Doyen-Token zu. Allgemeiner wurde bereits der Begriff der Prozessrollenbezeichnung eingeführt. Diese Unterscheidung zwischen Token, die quasi nur eine Selbstauskunft ähnlich einem Attribut über den Tokenbesitzer geben, und Token, die weitergegeben werden, muss sich im Modell wiederspiegeln.

Weitergabe bzw. Wertänderung eines Tokens

Im Verlauf der Behandlung können sie die Begebenheiten und Prozessrollen verändern. Daher muss es möglich sein, dass ein Token seinen Wert ändern bzw. ein Token an einen anderen Akteur weitergegeben werden kann. Beispielsweise kann ein Arzt während der Behandlung seinen Status des Patientenkontakts verlieren oder die Wortführerschaft wird auf einen anderen Arzt übertragen.

Bestätigung eines Token-Empfangs

Prinzipiell ist es wünschenswert, dass der Sender eines Token über den Empfang auf der Empfängerseite informiert wird. Diese Anforderung wird aber dadurch abgeschwächt, dass im Fall von α-Flow von einem zuverlässigen Kommunikationskanal ausgegangen wird, der folgende Eigenschaften aufweist ([Wah11, Seite 14ff. und Seite 80]):

- Der Kommunikationskanal muss im Falle von nicht erreichbaren Kommunikationspartnern in der Lage sein, Nachrichten für den späteren Abruf zwischenzuspeichern.
- Der Kommunikationskanal muss gewährleisten, dass eine abgesendete Nachricht auch beim Empfänger ankommt.

Identifikation des aktuellen Tokens

Der verwendete Kommunikationskanal garantiert, dass Nachrichten beim Empfänger ankommen. Allerdings sorgt er nicht dafür, dass diese in der richtigen Reihenfolge dort eintreffen. Um zu gewährleisten, dass veraltete Nachrichten nicht zu Inkonsistenzen führen, muss beim Eintreffen einer Nachricht die Aktualität eines Tokens überprüft werden können.

Einzigartigkeit bestimmter Token

Von einigen Token existiert im α-Doc jeweils genau ein Exemplar. Im Sinne der Prozessrollen kann es nur einen einzigen Prozessinitiator geben. Genauso wird definiert, dass es immer nur einen Wortführer im Prozess geben kann. Es muss sichergestellt werden, dass dies jederzeit der Fall ist.

Ablehnung einer Prozessrollenübertragung

Bei der Weitergabe des Doyen-Tokens wird ein fachlicher Prozess abgebildet. Die Bestimmung eines neuen Wortführers erfolgt außerhalb der technischen Abbildung des Prozesses im α-Doc. Etwaige Abstimmungsprozesse müssen über einen anderen Kommunikationskanal ablaufen. Daher wird eine Bestätigung der Annahme des Tokens bzw. das Ablehnen eines Tokens, wie es bei Freundschaftseinladungen in sozialen Netzwerken üblich ist, nicht unterstützt. Sollte es zu einer irrtümlichen Weitergabe des Tokens kommen, so wird der Fehler behoben, indem das Token wieder weitergegeben wird.

5.3.2 Entwurf des Anzeige- und Bedienkonzeptes

Die im Zuge dieser Arbeit eingeführten Token sollen dem Benutzer angezeigt werden. Dazu ist es nötig, die Komponente des α-Editors anzupassen. Wie in Abschnitt 3.3 beschrieben stellt der α-Editor die Benutzerschnittstelle zum Systemkern dar. Er gibt den aktuellen Status des α-Docs wieder und nimmt Benutzereingaben entgegen. Der Editor soll also die Token grafisch abbilden und Änderungen an diesen zulassen.

Dynamische Visualisierung der Token

Das Token-Modell soll laut seinen funktionalen Anforderungen erweiterbar sein. Dazu ist es nötig, auch die Anzeige dynamisch zu erzeugen. Da aber auch die drei Standard-Token immer vorhanden sind, wird zwischen einer statischen Komponente für die Standard-Token und einer dynamischen für etwaige weitere Token unterschieden.

Weitergabe-Maske

Das Doyen-Token ist so entworfen, dass es an einen anderen Teilnehmer weitergegeben werden kann. Dazu muss ein geeigneter Dialog vorhanden sein, der es ermöglicht, aus einer Liste der teilnehmenden Ärzte einen neuen Wortführer zu bestimmen. Der Dialog leitet dann die getätigten Eingaben an den Systemkern weiter, der die tatsächliche Weitergabe der Token tätigt.

Des Weiteren muss es durch die Weitergabe-Maske möglich sein, während einer Behandlungsepisode das Patientenkontakt-Token zu ändern. Dies wird ermöglicht, indem diese Eigenschaft für einen Akteur im α-Doc an- und abwählbar gestaltet wird.

5.4 Fachlicher Lösungsansatz für das Token-Modell

In diesem Kapitel wird beschrieben, wie das Token-Modell konzipiert ist. Zu Beginn wird auf die Basis des Modells eingegangen: das α-Adaptive-Modell. Daran anschließend wird dieses erweitert, um es an die speziellen Gegebenheiten im Fall der Token anzupassen.

5.4.1 Benutzung eines adaptiven Schemas für die Token

Die Grundlage für das Datenschema der Token liefert das adaptive Schema für die α-Adornments. Das in [Sch11] ausgeführte Attributmodell erfüllt alle Voraussetzungen, die benötigt werden. Insbesondere die Eigenschaft, dass das Token-Modell zur Laufzeit

angepasst werden kann, ist erfüllt. Vom Adornment-Modell (cf. Abschnitt 3.2.3) werden fast alle Parameter übernommen, diese aber auch zusätzlich erweitert.

Jedes Token hat einen Namen und bekommt einen Wert zugewiesen. Der Name der Standard-Token wird dabei statisch festgelegt.

Der Wertebereich eines Tokens basiert auf dem Adornment-Datentyp der Aufzählung. Mit diesem Datentyp ist es möglich, dynamische Aufzählungen zu bestimmen. Die Werte der Aufzählung können damit selbst festgelegt werden und bei Bedarf auch verändert werden. Im Fall der Token umfasst der Wertebereich jeweils zwei Werte. Nämlich einmal das gesetzte Token und einmal das Nicht-Vorhandensein des Tokens. Der Wertebereich der Token ist in Abbildung 5.3 zu sehen.

Ein Instanz-Parameter, wie er bei den α-Adornments benutzt wird, ist an dieser Stelle allerdings nicht nötig, da momentan auf eine Unterscheidung in Token-Schema-Menge und Token-Instanz-Menge verzichtet wird. Es werden bei jedem Teilnehmer immer alle drei Standard-Token benutzt. Ein Teilnehmer besitzt entweder das Token oder er besitzt es nicht, was durch eine entsprechende Belegung des Tokens ausgedrückt wird. Der hohe Konsens über die Menge der Token zeigt sich auch darin, dass der Gültigkeitsbereich der Token generell domänenspezifisch ist (cf. Abschnitt 3.2.3).

5.4.2 Versionierung der Token

Um der Anforderung nach der Bestimmung der Aktualität eines Tokens Rechnung zu tragen, wird auf Token-Ebene ein logischer Zeitstempel eingeführt. Eine kausale Beziehung zwischen zwei Versionen eines Prozessartefakts bzw. eines Tokens kann durch Versionsvektoren hergestellt werden (cf. Abschnitt 3.2.4). Die von Wahl eingeführten logischen Zeitstempel basieren auf eben diesem Prinzip der Versionsvektoren (cf. [Wah11]). Sie können auch dazu benutzt werden, aus zwei Token das aktuellere herauszufinden.

Bedeutung der kausalen Ordnung

Die in Abschnitt 3.2.4 erläuterten Zeitstempel ermöglichen es, eine kausale Ordnung zwischen zwei Versionen eines Artefakts herzustellen. Im weiteren Verlauf dieser Arbeit wird insbesondere wichtig sein, zu wissen, ob ein Token neuer ist als ein anderes. Wenn ein Token geändert wird oder es weitergegeben wird, dann wird eine neue Version des Tokens erstellt. Beim Vergleich von ankommenden Nachrichten, die eine Token-Änderung oder -Weitergabe betreffen, können dann zwei Fälle eintreten: Nämlich, dass das ankommende Token neuer ist, und damit in irgendeiner Form verarbeitet werden muss, oder dass die lokal vorhandene Token-Version neuer ist. Dann muss die eingehende Nachricht

verworfen werden. Ebendiese zeitliche Abhängigkeit kann mittels der in Abschnitt 3.2.4 vorgestellten Zeitstempel ermittelt werden.

5.4.3 Impact-Scope

Wie bereits in Abschnitt 5.3.1 angedeutet, können grundlegend zwei Arten von Prozessrollen unterschieden werden. Um diese Unterscheidung im Token-Modell wiederzufinden, muss es um einen zusätzlichen Parameter erweitert werden. Eine Unterscheidung wird durch den Impact-Scope getroffen, der die Einflussgröße auf den Wert einer Prozessrolle beschreibt. Dabei werden Prozessrollen, die ein Teilnehmer für sich selbst festlegt, und solche, die einmalig im Modell existieren und durch Weitergabe vergeben werden, unterschieden.

Eine Art von Prozessrollen bezieht sich nur auf den Rolleninhaber selbst und kann im Allgemeinen von diesem selbst verändert werden. Der Wert der Prozessrolle ändert sich von innen heraus, daher auch die Bezeichnung `INTRINSIC`. Dieses Art von Prozessrolle gibt wie ein Attribut einer Klasse Auskunft über den momentanen Status des Objekts. Hierzu zählt die Prozessrolle des Prozessinitiators und auch der Patientenkontakt. Der Wert des Prozessinitiator-Rolle wird einmalig beim Erstellen des α-Docs festgelegt und ist unveränderlich bzw. wird im Fall der Patientenkontakt-Rolle beim Einladen eines Teilnehmers gesetzt und kann im weiteren Verlauf von diesem selbstständig geändert werden. Bei einer Wertänderung der jeweiligen Prozessrolle bezieht sich diese immer nur auf einen Teilnehmer. Mit Hinblick auf die Abfrage dieser Token könnte man auch von einer Selbstauskunft sprechen.

Die andere Art von Prozessrollen hingegen kann von Prozessteilnehmer zu Prozessteilnehmer weitergegeben werden. Ob ein Prozessteilnehmer also eine Prozessrolle dieser Art besitzt, wird von außen bestimmt, da sie ihm übergeben wird. Hierfür wird die Bezeichnung `EXTRINSIC` verwendet. Der Wert der Prozessrolle wird von außen bestimmt. Eine Änderung des Tokens betrifft also auch immer zwei Teilnehmer: den Sender und den Empfänger des Tokens. Zu dieser Token-Art gehört das Doyen-Token. Der momentane Doyen kann seine Prozessrolle an einen anderen Teilnehmer übergeben. Dies bewirkt, dass er sie nicht mehr innehat, dafür aber der andere Teilnehmer nun das Token besitzt und damit die Prozessrolle ausübt.

Die inhaltliche Abgrenzung dieser beiden Tokenarten soll im Modell durch einen zusätzlichen Parameter ausgedrückt werden. Der Parameter `IMPACT SCOPE` unterscheidet die Token hinsichtlich ihres Bezugs: also ob ein Token nur einen Selbstbezug in der Form

aufweist, dass der Wert des Tokens selbst festgelegt wird, oder durch Interaktion zwischen den Teilnehmern und Weitergabe des Tokens festgelegt wird (cf. Abbildung 5.3).

Token	Wertemenge	Impact-Scope
Initiator	{INITIATOR, NON_INITIATOR}	INTRINSIC
Patientenkontakt	{PATIENT_CONTACT, NON_PATIENT_CONTACT}	INTRINSIC
Wortführer	{DOYEN, NON_DOYEN}	EXTRINSIC

Bild 5.3: Impact-Scope von Token

5.4.4 Einordnung in das α-Flow Domänenmodell

Die Token-Menge kann während einer α-Episode einem Teilnehmer zugeordnet werden, der diese besitzt. Es bietet sich daher an, sie diesem durch ein weiteres Attribut anzuhängen.

Damit einhergehend sind einige weitere Umstrukturierungen des in Kapitel 3.2 beschriebenen α-Domänenmodells nötig. Der Teilnehmer wird um eine Liste an Token erweitert und es müssen die geeigneten Schnittstellen geschaffen werden, um diese zu erstellen, zu bearbeiten und sie zu lesen.

Dadurch, dass die Klasse `Participant` erweitert wird, muss diese refaktorisiert werden. Die momentane Kapselung von `Contributor` und `Endpoint` ist nicht mehr direkt möglich. Der Contributor beschreibt die Aktoren-relevanten Attribute des Namens, der Institutionszugehörigkeit und der Rolle, die dieser in der aktuellen α-Episode einnimmt. Der Endpoint ist der sog. Kommunikationsendpunkt, also die Information darüber, wo der Akteur für das α-Doc erreichbar ist. Diese beiden Klassen müssen losgelöst vom Participant mit seinen Token betrachtet werden können, da sie auf technischer Ebene den Teilnehmer identifizieren. Erweitert werden sie auf fachlicher Ebene durch die Token. Darum ist es nötig, die beiden Klassen `Contributor` und `Endpoint` in einem Knoten als technischen Identifikator zu kapseln. Ein `Participant` besteht dann aus diesem Node und seinen Token (cf. Abbildung 5.4).

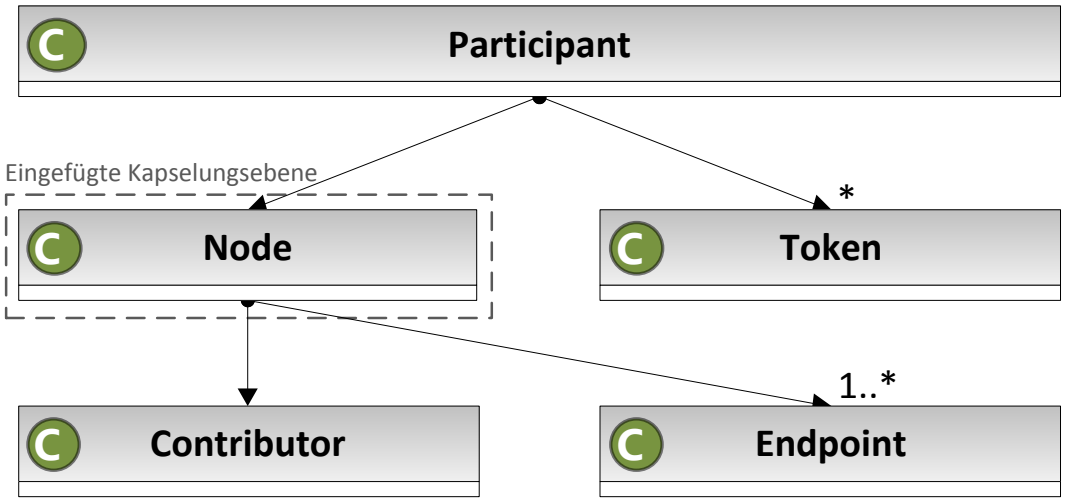


Bild 5.4: Refaktorisierte Participant-Klasse

Gegenüber der hier beschriebenen grobgranularen Einordnung der Token auf Teilnehmer-Ebene wäre prinzipiell auch eine feingliedrigere möglich. Denkbar wäre eine Ansiedlung der Token auf α-Card-Ebene. Dort müsste dann eine neue Struktur geschaffen werden, die jeweils eine Liste der Participants mit ihren Token enthält. Eine Wartbarkeit dieser Möglichkeit ist aber deutlich schwieriger, weil bei einer Änderung eines Token, die Strukturen aller α-Cards aktualisiert werden müssten. Darum wird von dieser Lösungsmöglichkeit Abstand genommen. Überhaupt erscheint es sinnvoller, die grobgranularere Variante zu wählen, weil die Participants für ein α-Doc einheitlich sind und die Token wiederum für diese.

5.5 Nachricht bei Tokenweitergabe

Wie in Abschnitt 5.4.3 deutlich geworden ist, erfolgt eine Token-Weitergabe im eigentlichen Sinne nur im Falle von Token, die den Impact-Scope `EXTRINSIC` haben, also im vorliegenden Fall dem Doyen-Token. Änderungen an Token mit Impact-Scope `INTRINSIC` können wie ein Update am CRA behandelt werden und bedürfen keines neuen Nachrichtentyps. Die Doyen-Weitergabe hingegen erfordert die Einführung eines neuen Ereignisses im α-Doc-Workflow.

[Wah11] beschreibt die Struktur der zur Synchronisation einer verteilten Fallakte benötigten Nachrichten. Wenn durch einen Akteur lokal eine Änderung an einem Prozessartefakt durchgeführt wurde, wird diese mittels Nachrichten asynchron propagiert. Da durch die Annahmen über den Kommunikationskanal (cf. Abschnitt 5.3.1) zwar

sichergestellt werden kann, dass die Nachrichten ankommen, aber keine Aussage über die Empfangsreihenfolge gemacht werden kann, ist es nötig, die Änderungen mit einer gewissen Struktur zu versehen.

Die Token-Weitergabe-Nachricht wird durch denjenigen Teilnehmer, der das Token weitergibt, generiert und an alle Prozessteilnehmer verschickt. Dann wird sie auf der Empfängerseite deterministisch verarbeitet, sodass alle Teilnehmer nach der Verarbeitung wieder den gleichen Prozessstatus haben.

Zur erfolgreichen Weitergabe eines Token müssen folgende Attribute Bestandteil der Token-Weitergabe-Nachricht sein (cf. auch Abbildung 5.5):

- der Sender des Tokens,
- der Empfänger des Tokens und
- das weiterzugebende Token.

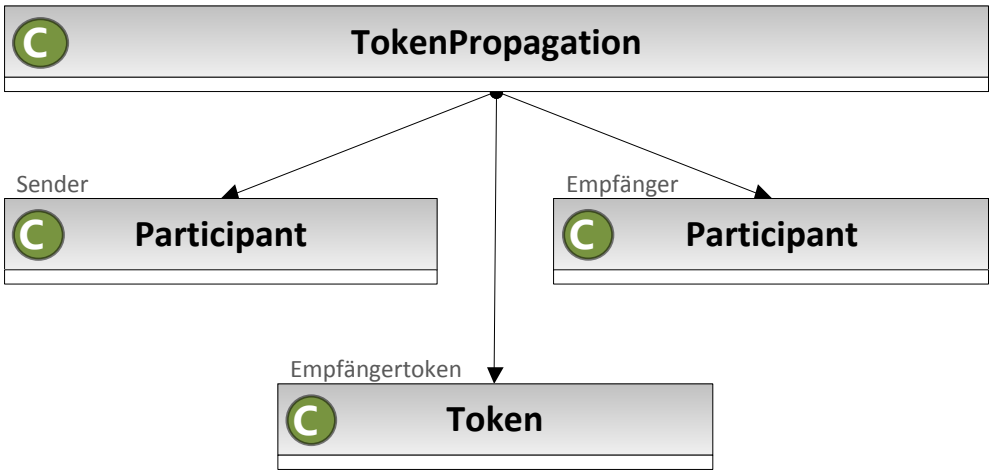


Bild 5.5: Nachricht bei Tokenweitergabe

Wie bereits erwähnt, muss eine Nachricht dahingehend beurteilt werden können, ob sie noch aktuell ist; also ob das Token, das sie enthält, bereits wieder weitergegeben wurde oder nicht. Um eine Ordnung auf dem Token aus der Nachricht und dem lokal vorliegendem Token herzustellen wird ein logischer Zeitstempel herangezogen. Dieser ist bereits im Token selbst enthalten. Bei einem Vergleich der logischen Zeitstempel werden nur Nachrichten mit einem neuerem Token, also einer neueren Version als der lokal vorhandenen, bearbeitet. Wird eine Nachricht mit einem älteren Token gelesen, darf sie zu keiner Änderung führen und muss verworfen werden. Da die logischen Zeitstempel sowohl bereits im lokalen Token als auch im der Nachricht angehängten Token enthalten sind ist ein zusätzliches Attribut innerhalb der Nachricht also nicht nötig.

5.6 Änderungskonflikte bei Token-Änderungen

Jeder Teilnehmer am Prozess kann eigenständig unabhängig von anderen Teilnehmern seine Token ändern. Jeder kann das Token Patientenkontakt ändern und der Wortführer kann seine Eigenschaft als solchen weitergeben. Treten diese Änderungen der verschiedenen Teilnehmer überschneidend auf, kann es in deren Folge zu inkonsistenten Zuständen kommen, was die Werte der Token angeht. Wenn ein Akteur nicht über alle anderen Änderungen informiert ist, liegt ein nebenläufiger Änderungskonflikt vor. Im Fall von α-Flow sind insbesondere zwei Fälle als kritisch zu betrachten und werden daher im Folgenden näher betrachtet: Zum einen kann es zu Token-Änderungen kommen, während ein neuer Teilnehmer im Begriff ist, dem Prozess beizutreten. Zum anderen können Teilnehmer wie bereits erwähnt nebenläufig Änderungen an ihren Token durchführen.

5.6.1 Änderungsanomalien während des Teilnehmerbeitritts

In [Wah11] wird ein Beitrittsprotokoll entwickelt, dass es den am Behandlungsprozess beteiligten Ärzten ermöglicht, problemlos neue Akteure einzuladen. Dabei werden den Beitretenden die aktuellen Informationen über den Prozessverlauf zur Verfügung gestellt. Für das Beitreten wird dem neuen Akteur das α-Doc in Form einer Kopie des bestehenden Dokuments mitgegeben. Beim ersten Öffnen des Dokuments werden die Adressinformationen des neuen Teilnehmers abgefragt, die für den Austausch von Aktualisierungen benötigt werden und dann der aktuelle Prozessstatus durch Nachrichten ausgetauscht.

Das Beitrittsprotokoll unterstützt sowohl den sequentiellen als auch den parallelen Beitritt von neuen Akteuren. Es stellt insbesondere sicher, dass es während des Beitritts zu keinen Inkonsistenzen kommt.

Informationsdefizite während des Beitritts

Zu dem Zeitpunkt, an dem die Einladung erstellt wurde, sind auf der einen Seite nicht zwingend alle Prozessbeteiligte allen Teilnehmern bekannt. Auf der anderen Seite können durch die Einführung der Token Inkonsistenzen entstanden sein, weil ein Teilnehmer seine Token geändert haben könnte, ohne die Änderung an den Beitretenden zu propagieren.

Damit alle Prozessbeteiligten nach dem Beitritt alle anderen kennen und an diese Änderungen verbreiten können, ist das Beitrittsprotokoll entsprechend aufgebaut. Während der Beitrittsphase wird die Liste aller Prozessbeteiligten ausgetauscht und aktualisiert, sodass nach Ablauf des Protokolls alle Teilnehmer auf dem aktuellsten Stand sind.

Genauso müssen, nachdem alle Prozessbeteiligten bekannt sind, deren Token-Werte mit den lokal vorliegenden Werten verglichen und ggf. aktualisiert werden. Dabei muss, wie in Abschnitt 5.4.2 bereits erläutert, darauf geachtet werden, dass nur Token mit einem neueren Zeitstempel zu Änderungen führen.

Um dieses Verhalten zu erreichen, wird das Beitrittsprotokoll um eine Merge-Routine auf Token-Ebene erweitert, die die Zeitstempel des lokal vorliegenden Tokens mit dem Token aus der während des Beitritts verschickten Nachricht vergleicht und bei einer neueren Version den Wert lokal ersetzt.

5.6.2 Nebenläufige Änderungsanomalien im Prozessverlauf

Während des Prozessverlaufs können die Teilnehmer nebenläufig Änderungen an ihren Token durchführen. Dabei sind verschiedene Änderungsanomalien denkbar und sollen im weiteren Verlauf dieses Kapitels betrachtet werden.

Die Nachrichten, die diese Änderungen verbreiten, müssen nicht zwangsläufig in der Reihenfolge des Versands eintreffen und verarbeitet werden (cf. Abbildung 5.6a). Die Abbildung zeigt, wie eine zweite Änderung von Akteur A vor der ersten Änderung bei Akteur B eintrifft. Wenn Akteur B die zweite Änderung erhält, muss sichergestellt werden, dass die erste Änderung nicht verloren geht.[1]

Durch nebenläufige Änderungen an Token können auf den ersten Blick ebenfalls Anomalien entstehen, die entsprechend behandelt werden müssen (cf. Abbildung 5.6b).

Entsprechend der Unterscheidung durch den Impact-Scope eines Token müssen die Änderungsanomalien gesondert betrachtet werden.

`INTRINSIC`-Token

Token mit Selbstbezug können nur durch den besitzenden Akteur geändert werden. Eine Änderung eines oder mehrerer Token führt zu einem CRA-Update-Ereignis. Allerdings wird dieses feingranular auf Teilnehmer-Ebene verteilt. D.h. eine Änderung betrifft immer wirklich nur einen Teilnehmer selbst. An dieser Stelle kann es also nicht zu dem beschriebenen potentiell verlorenen Update kommen, da eine nebenläufige Änderung nicht möglich ist.

1 Diese Anomalie wird auch als Lost Update bezeichnet (cf. [BBG+95]): Eine Transaktion T1 liest ein Datenelement und dann überschreibt T2 dieses Datenelement. Darauf überschreibt wiederum T1 basierend auf dem gelesenen Wert das Datenelement. Das Update von T2 ist damit verloren. Betrachtet man die Historie der Transaktionen ergibt sich folgendes Bild: $r1[x]...w2[x]...w1[x]...c1$.

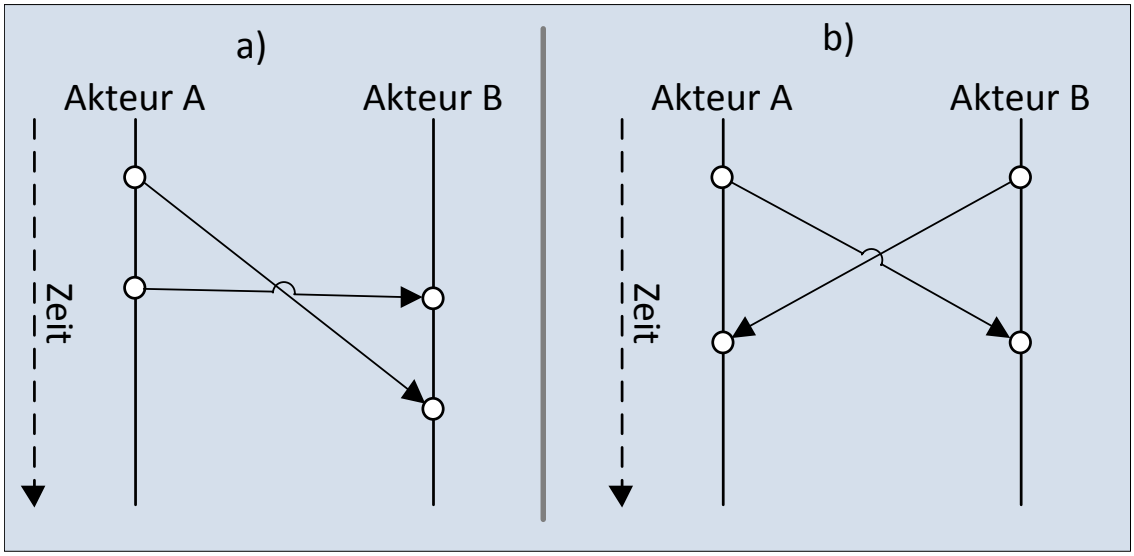


Bild 5.6: Änderungskonflikte während der Token-Änderung

Bei mehreren aufeinander folgenden Änderungen kann es aber sehr wohl der Fall sein, dass diese außerhalb der Reihenfolge bei einem anderen Teilnehmer eintreffen. An dieser Stelle muss mittels Zeitstempel eine Ordnung geschaffen werden und die Änderungen müssen wie bereits beschrieben entsprechend verarbeitet werden.

`EXTRINSIC`-Token am Beispiel des Doyen-Token

Auch beim Doyen-Token kann es zu keinen nebenläufigen Änderungsanomalien kommen, da immer nur der aktuelle Doyen, das Token besitzt und damit weitergeben kann. Genauso können die Nachrichten hier außerhalb der Reihenfolge eintreffen. Nur das aktuellere Token darf dabei verarbeitet werden. Beim Doyen-Token ist weiterhin zu beachten, dass zwei Token verändert werden: das Token des ehemaligen Doyen und das des neuen Doyen. Diese Änderung spiegelt quasi eine Transaktion wieder, die nur als Ganzes ausgeführt werden darf.[1]

1 Unter den ACID-Eigenschaften einer Transaktion sind Atomarität, Konsistenz, Isolation und Dauerhaftigkeit zusammengefasst. Atomarität steht dafür, dass eine Transaktion entweder vollständig oder gar nicht ausgeführt wird. Konsistenz beschreibt, dass eine Transaktion die Daten von einem konsistenten Ausgangszustand in einen konsistenten Endzustand überführt. Isolation bedeutet, dass eine Transaktion nebenläufige Änderungen nicht wahrnimmt. Dauerhaftigkeit legt fest, dass Änderungen einer Transaktion nach erfolgreichem Beenden bestehen bleiben (cf. [HR83, Seite 290f.]).

5.7 Technisches Lösungskonzept

In diesem Abschnitt wird das technische Fachkonzept für α-Doyen beschrieben. Dazu wird anfangs das Datenmodell der Token erklärt. Dann wird beschrieben, wie der Ablauf stattfindet, wenn im Prozessverlauf der Wert eines Tokens geändert wird bzw. ein Token weitergegeben wird. Im Abschluss wird die α-Editor-Komponente betrachtet, die um entsprechende Anzeige- und Bedienelemente erweitert wurde.

5.7.1 Token-Modell

Dieses Kapitel beleuchtet das Attributmodell der Token. Zunächst wird auf die Prozessteilnehmer eingegangen. Anschließend wird die technische Umsetzung der Token betrachtet und dabei auf die Umsetzung der in Abschnitt 5.4 konzipierten Parameter eingegangen. Dann werden die einzelnen Abläufe vorgestellt, die bei einer Token-Änderung vonstattengehen. Hierbei muss zwischen der Änderung eines Token mit Selbstbezug und der Weitergabe des Doyen-Tokens unterschieden werden.

5.7.1.1 Technischer Aufbau der Participant-Klasse

Die Abbildung 5.7 zeigt den Aufbau der Participant-Klasse nach seiner Refaktorisierung. Ursprünglich beinhaltete die Klasse `Participant` einen `Contributor` und einen `Endpoint`. Diese beiden wurden zu einer `NodeID` zusammengefasst und die Klasse um eine Liste von Token erweitert, die während der Laufzeit modifiziert werden kann.

Refaktorisierung

Der nun in der `NodeID` enthaltene `Contributor` beinhaltet folgende Informationen:

1. die Bezeichnung des Akteurs,
2. die Bezeichnung der Institution, für die der Akteur am Prozess teilnimmt und
3. die Bezeichnung der Rolle, die der Akteur im Prozessverlauf einnimmt.

Der Endpoint umfasst alle für die Kommunikation mit den am Prozess beteiligten Akteuren benötigten Informationen. Beide Attribute zusammen kennzeichnen einen Participant eindeutig als Kommunikationsendpunkt. Daher wurden sie im Verlauf der Arbeit als `NodeID` zusammengefasst.

Dies ist insbesondere von Vorteil, weil damit eine Kapselung stattfindet. Für die Konfiguration des α-Doc benötigten Informationen über den Endpunkt können damit losgelöst von den aktuellen Tokenzuständen abgespeichert werden.

Schnittstelle zum Token-Zugriff

In der Klasse `Participant` existiert eine Methode *readToken* zum Auslesen eines einzelnen Token-Wertes aus der Liste aller Token des Akteurs. Des Weiteren wird eine Methode zum Ändern eines Token-Wertes bzw. zum Hinzufügen eines Tokens zur Liste bereitgestellt: *updateOrCreateToken*. Die Methode *deleteToken* löscht ein bestehendes Token aus der Liste. Um einen Überblick über alle vorhandenen Token zu erhalten existiert die Methode *readTokens*, die alle Token des Participant als Rückgabewert hat. Der Zugriff auf die Liste erfolgt über den partiellen Schlüssel des Token-Namens.

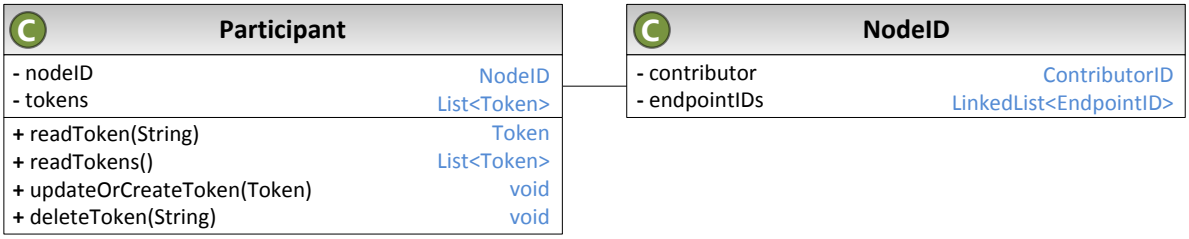


Bild 5.7: Struktur der Klasse `Participant`

5.7.1.2 Aufbau der adaptiven Token

Gemäß den in Abschnitt 5.3.1 gestellten Anforderungen wird die Klasse `Token` von der bereits existierenden Klasse `AdpAdornment` abgeleitet und entsprechend erweitert.

Abbildung 5.8 zeigt, dass Token einen Namen und einen Wert haben. Der Name dient als partieller Schlüssel während des Zugriffs auf die Tokenliste im `Participant`. Der Wertebereich wird über spezielle Aufzählungstypen, die die möglichen Werte vorgeben, bestimmt. Der in `AdpAdornment` festgelegte Datentyp ist im Fall der Token standardmäßig auf eine Enumeration eingestellt. Ebenso erhält der `ConsensusScope`, der das Maß des Konsens unter den Prozessteilnehmern angibt, den Wert „generisch“ zugewiesen. D.h., die Token sind prinzipiell System-relevant und werden dauerhaft eingesetzt.

Das Token selbst erweitert das `AdpAdornment` um einen ImpactScope und hat für die Nachvollziehbarkeit der Änderungsreihenfolge eine `VersionMap`.

Die Methode `getNegatedValue` liefert den negierten Wert des Token zurück. Die Menge an Werten, die einem Token zugeordnet werden kann, ist immer zweiwertig:

Entweder der Participant besitzt das Token oder er besitzt es nicht. Daher kann durch die Methode immer der entgegengesetzte Wert geliefert werden.

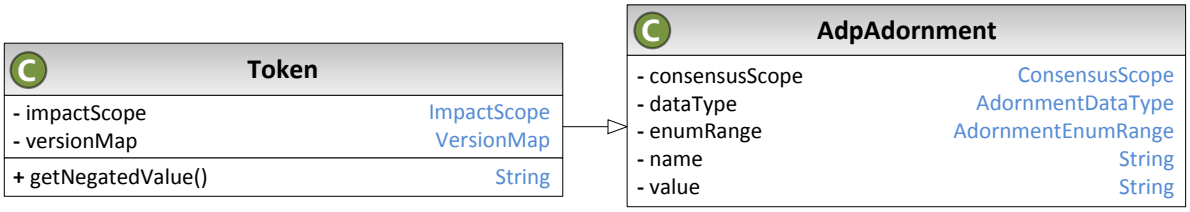


Bild 5.8: Struktur der Klasse `Token`

5.7.1.3 Änderung eines Tokens

Eine Token-Änderung wird durch den Benutzer im α-Editor angestoßen (cf. Abbildung 5.9). Dieser leitet sie an die `AlphaPropsFacade` weiter. Dann wird in der `AlphaPropsFacade` ein `ChangePayloadEvent` erzeugt, welches die benötigten Informationen enthält. Dazu zählen der Verweis auf die zu ändernde α-Card, nämlich das CRA, und die komplette Änderung, also den `Participant`. Dieses Ereignis wird in die Regelbibliothek eingefügt. Dort ändert die entsprechende Regel den `Participant` und speichert die Änderung im CRA ab. Kommt es zu einer Token-Änderung, wird dies erkannt, und die Version des Token erhöht. Im Anschluß wird das Ereignis durch α-Overnet an alle anderen Teilnehmer verbreitet (`sendUpdate`-Methode).

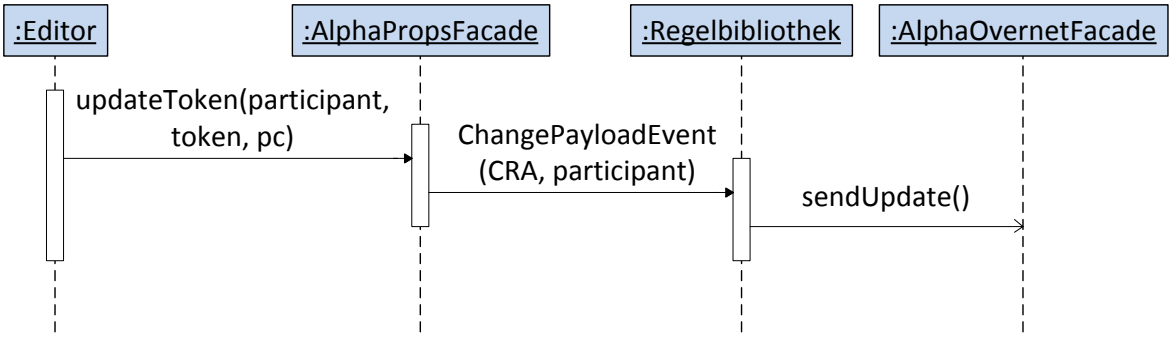


Bild 5.9: Eine Token-Änderung anstoßen

Wenn ein `ChangePayloadEvent` eintrifft, das über α-Overnet empfangen wurde, dann wird es in die Regelbibliothek eingefügt und dort analog verarbeitet. Allerdings entfällt in dem Fall das Versenden einer weiteren Nachricht und es entfällt das Erzeugen einer neuen Version für die geänderten Token. Eine Änderung findet nur statt, wenn es sich beim empfangenen Token um eine neuere Version als die lokal vorhandene handelt.

5.7.1.4 Weitergabe des Doyen-Tokens

Die Weitergabe des Doyen-Tokens verläuft grundsätzlich verschieden von einer einfachen Token-Änderung, wie sie im vorhergehenden Kapitel beschrieben wurde. Bei der Weitergabe eines Tokens sind immer zwei Akteure beteiligt (cf. Abschnitt 5.5).

Die Weitergabe wird dabei vom aktuellen Token-Inhaber eingeleitet. Sie wird vom Editor entgegengenommen und an die `AlphaPropsFacade` mit dem neuen Doyen als Parameter weitergegeben (cf. Abbildung 5.10). Dort wird ein Ereignis vom Typ `TokenPropagation` erzeugt und in die Regelbibliothek eingefügt. Die in `TokenPropagation` enthaltenen Parameter sind der sendende Akteur, der Empfänger des Tokens und das Token selbst (cf. Abbildung 5.11). Das Ereignis wird in der Regelbibliothek verarbeitet und an alle anderen Prozessteilnehmer versendet.

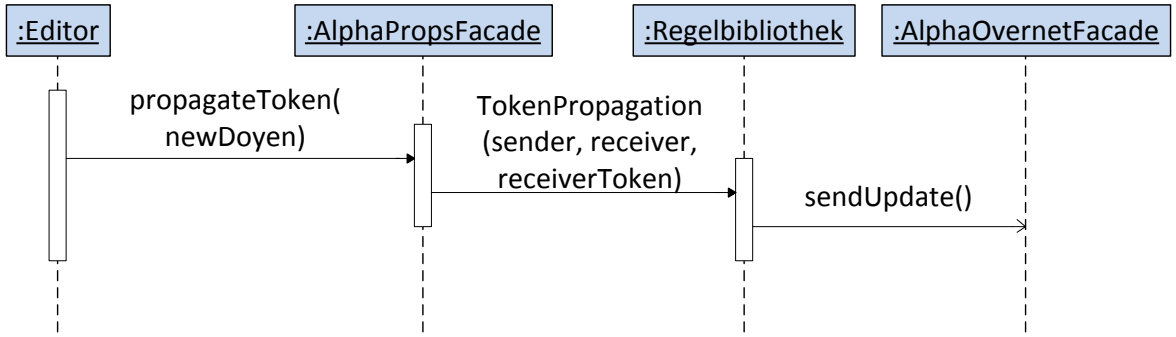


Bild 5.10: Ablauf bei der Token-Weitergabe

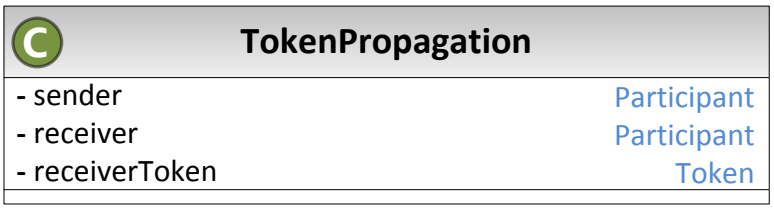


Bild 5.11: Struktur der Klasse `TokenPropagation`

Schnittstelle zur Token-Weitergabe

Die Erzeugung und Verarbeitung der Nachrichten zur Token-Weitergabe ist in der Werkzeug-Klasse `TokenUtility` gekapselt. Damit ist eine erhöhte Wartbarkeit und Übersichtlichkeit gewährleistet, da nur eine zentrale Stelle angesprochen werden muss (cf. Abbildung 5.12). Die Schnittstelle kommuniziert zur Verarbeitung der eingehenden Nachrichten mit der Bibliothek im α-Properties-Modul.

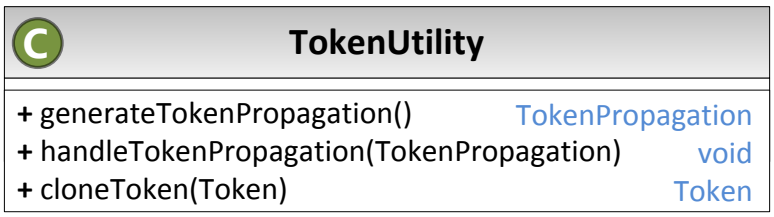


Bild 5.12: Schnittstelle zur Erstellung und Verarbeitung von Token-Weitergaben

Ein Aufruf von `generateTokenPropagation()` erzeugt ein neues Ereignis zur Token-Weitergabe. Dieses muss im Folgenden dann nur noch in die Regelbibliothek eingefügt werden, um die Weitergabe einzuleiten. Der Methode `handleTokenPropagation` werden eingehende Weitergabe-Ereignisse als Parameter übergeben (cf. Abbildung 5.13). Diese behandelt anschließend die eingehenden Weitergabe-Ereignisse entsprechend und führt die nötigen Änderungen an den Akteuren durch.

Dabei vergleicht die Methode `handleTokenPropagation` die Zeitstempel der Token aus dem Speicher der lokalen Regelbibliothek mit den Zeitstempeln der Token aus der eingehenden Nachricht. Auch hier wird durch den Parameter `propagateChange` festgelegt, ob Änderungen nach ihrer lokalen Anwendung auch propagiert werden. Bei eingehenden Nachrichten weist `propagateChange` den Wert `false` auf: die Änderungen werden also durchgeführt, aber nicht erneut verbreitet.

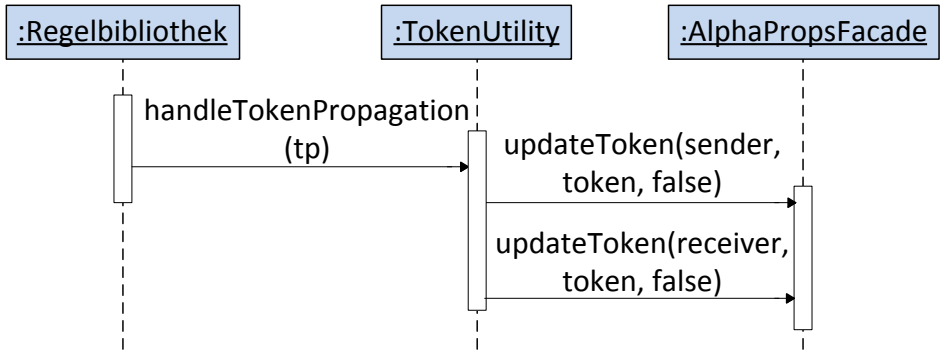


Bild 5.13: Verwendung der Schnittstelle `TokenUtility`

5.7.2 Anpassung der Editor-Komponente

Die Komponente des α-Editors wurde an mehreren Stellen um geeignete Konzepte zur Darstellung, Änderung und Weitergabe von Token erweitert.

Zur übersichtlichen Darstellung der Token wurde die Tabellenansicht aller Prozessteilnehmer um ein Panel erweitert, das beim ausgewählten Teilnehmer vorhandene Token als Symbol enthält. Diese Symbole sind in Abbildung 5.1 bereits vorgestellt worden. Ist ein Token nicht vorhanden, wird das entsprechende Symbol ausgeblendet. Dies sorgt beim betrachtenden Akteur für eine schnelle Übersicht.

Die Modifikation der Token wurde in einen Dialog ausgelagert (cf. Abbildung 5.14). Dieser weist zwei verschiedene Ansichten auf. Zum einen eine Standard-Ansicht für den normalen Benutzer, zum anderen einen sog. Admin-Modus. Die Standard-Ansicht kann dazu verwendet werden, das Patientenkontakt-Token zu ändern und das Doyen-Token weiterzugeben, sofern man Wortführer des Prozesses ist. Weiterhin zeigt sie nochmal den aktuellen Doyen des Prozesses an.

Der Admin-Modus greift auf die Anzeige-Komponente von α-Adaptive zurück und stellt alle Token generisch dar.[1] Soll ein Token geändert werden, muss dieses über einen Button markiert und der neue Wert aus einer Combobox ausgewählt werden. Dieser Modus ist dazu gedacht, das erweiterbare Token-Modell darzustellen und an ihm Wert-Änderungen durchzuführen. Im Fall des Doyen-Tokens bildet er allerdings nicht den transaktionalen Charakter ab, sondern ändert hart den Wert des Tokens, ohne das Token selbst an einen anderen Akteur weiterzugeben. Der Admin-Modus kann also insbesondere dazu genutzt werden, Fehler bei der Token-Weitergabe zu berichtigen und Inkonsistenzen zu bereinigen. Bei einer Fehlanwendung kann er allerdings auch zu einer Mehrfachvergabe bestimmter Token führen und den Token-Änderungs- und Token-Weitergabe-Algorithmus durcheinander bringen. Der Admin-Modus sollte daher einen Bestandteil der weiteren Arbeiten an α-Flow darstellen, in dem diese Punkte Berücksichtigung finden.

1 Ein Screenshot des Admin-Modus ist im Anhang in Kapitel D abgebildet.

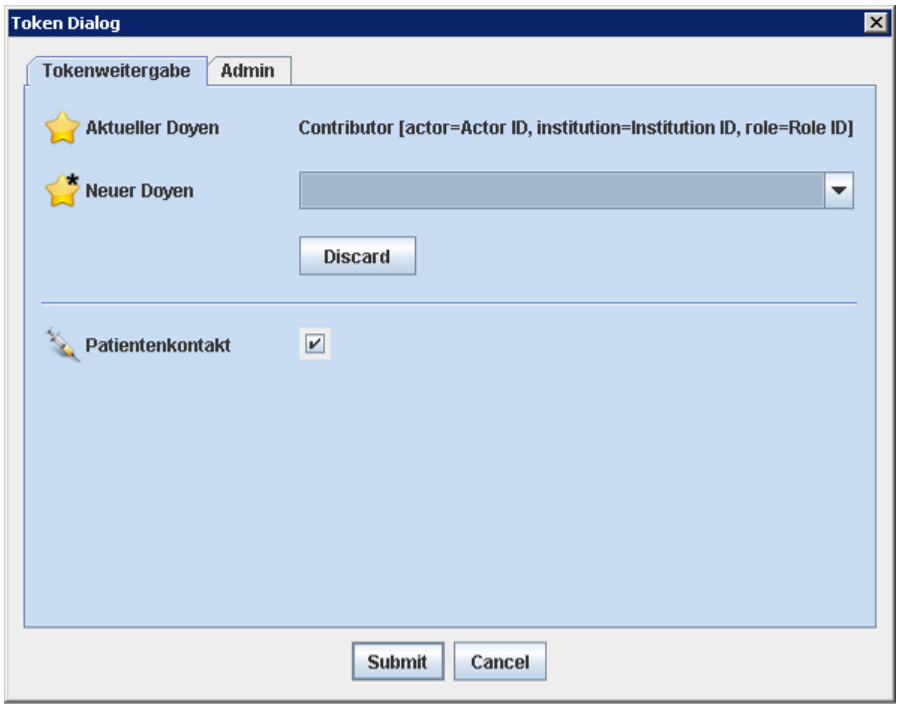


Bild 5.14: Ansicht des Token-Dialogs

5.8 Zusammenfassung

Im Rahmen dieser Arbeit soll ein Token-Konzept in das System-Modell integriert werden, das einzelne Prozessteilnehmer mit Rollen auszeichnen kann. Diese Token sollen weiterhin im verteilten System übertragbar sein.

Es werden drei verschiedene Prozessrollen identifiziert: der Initiator, der Doyen und die Rolle des Patientenkontakts. Diese Prozessrollen weisen eine jeweils eigene Semantik auf.

Im Kontext dieser Arbeit werden die Prozessrollenbeschreibungen in Token gespeichert und auch durch diese weitergegeben. Dabei ist zwischen Token zu unterscheiden, die vom Teilnehmer selbst vergeben werden und solchen die nur einmal im Prozess existieren und zwischen den Teilnehmern weitergegeben werden können.

Die verschiedenen Anforderungen an das Token-Modell werden erläutert und im folgenden Lösungsansatz umgesetzt. Grundlage für das Modell bildet ein aus α-Adaptive abgeleitetes adaptives Schema, das zur Laufzeit angepasst werden kann. Um eine kausale Ordnung auf den Token-Änderungen zu erhalten werden diese versioniert. Des Weiteren werden zwei Arten von Token unterschieden: Der Impact-Scope gibt an, ob es sich bei einem Token um eine Selbstauskunft oder ein Token mit Fremdbezug handelt.

Ferner wird ein Verfahren entwickelt, das die Token-Weitergabe ermöglicht. In diesem Zusammenhang wird auch auf mögliche Änderungskonflikte eingegangen, die während der Änderung bzw. Weitergabe eines Tokens auftreten können. Dazu zählt auch eine Merge-Routine, die bei der Token-Weitergabe die aktuellsten Token auswählt und verwendet.

Abschließend wird die technische Umsetzung des Modells und die Anpassung der Editor-Komponente betrachtet. Das Token-Modell fügt sich in bereits bestehende Module von α-Flow ein. Damit bleibt ein hohes Maß an Kapselung erhalten. Zum einen wird der `Participant` im α-Model um die Token erweitert, zum anderen werden der Kommunikationsschnittstelle entsprechende Nachrichten zur Token-Weitergabe hinzugefügt. In α-Properties finden die benötigten Behandlungsroutinen ihren Platz.

6 Rückmeldungen über Nachrichtenflüsse per Empfangsbestätigungen

Empfangsbestätigungen werden in einer Netzwerkumgebung dazu verwendet, den fehlerfreien Empfang von Datenpaketen zu bestätigen. Bei Erhalt einer solchen Bestätigung kann der Sender der Daten sicher sein, dass diese beim Empfänger angekommen sind. Die Einführung von Empfangsbestätigungen, auch Acknowledgements genannt, macht eine Verbindung zuverlässig. Allerdings wird auch der Aufwand erhöht, da zu jeder versendeten Nachricht eine Bestätigung verschickt werden muss (cf. [Tan07, Seite 32f.] und [VST02, Seite 220ff.]).

In diesem Kapitel soll ein Konzept erarbeitet werden, das α-Flow um Empfangsbestätigungen erweitert. Änderungen an α-Cards sollen von den empfangenden Akteuren bestätigt werden, damit der Sender jederzeit unterrichtet ist, ob seine Änderung schon wahrgenommen wurde.

In Abschnitt 6.1 wird auf Basis eines Anwendungsfalls die Motivation für die Einführung von Empfangsbestätigungen gelegt. Im Anschluss werden die Anforderungen an ein solches Modell erörtert. Darauf aufbauend werden verschiedene Lösungskonzepte untersucht und bewertet. Die prototypische technische Umsetzung wird auszugsweise in Abschnitt 6.3 betrachtet.

6.1 Motivation für Empfangsbestätigungen bei neuen α-Card-Versionen

Im Verlauf des Behandlunsgprozesses erzeugen die behandelnden Ärzte neue α-Cards, fügen bestehenden ein neues Payload hinzu oder ändern Adornments. Jede dieser Modifikationen erzeugt eine neue Version der α-Card. Diese Version des Prozessartefakts wird einerseits individuell und dauerhaft archiviert und zum anderen an alle Prozessteilnehmer

propagiert. Zur Archivierung kommt in α-Flow das Versionsverwaltungssystem Hydra zur Anwendung, das über Schnittstellen in das Gesamtsystem integriert ist (cf. [Had11]).

Bei α-Flow wird von einem Kommunikationskanal ausgegangen, der die Zustellung der Nachrichten sicherstellt (cf. Abschnitt 5.3.1). Anhand dieser Annahme kann der Sender also sicher sein, dass die Empfänger seine Änderungen erhalten. Er hat aber keine Informationen darüber, wann dies der Fall sein wird. Besonders bei für den weiteren Behandlungsverlauf kritischen Änderungen kann die Bestätigung des Erhalts der dazugehörigen Nachricht wichtig sein.

Abbildung 6.1 zeigt noch einmal die Schritte, die dem Versand einer Empfangsbestätigung vorangehen: zuerst wird eine Änderung an einem Prozessartefakt durchgeführt. Diese wird daraufhin an alle Prozessteilnehmer versandt. Wenn diese Änderung beim Empfänger eintrifft, sendet dieser eine Empfangsbestätigung, in der er über den Erhalt der Änderung informiert.

Der Empfang der Änderung wird vom α-Doc quittiert. Es handelt sich hier also nicht um eine Lesebestätigung durch den Akteur selber, sondern es wird auf technischer Ebene bestätigt, dass eine Änderung erhalten und verarbeitet wurde. Der Prozessteilnehmer selbst kann diese Änderung auch erst später wahrnehmen.

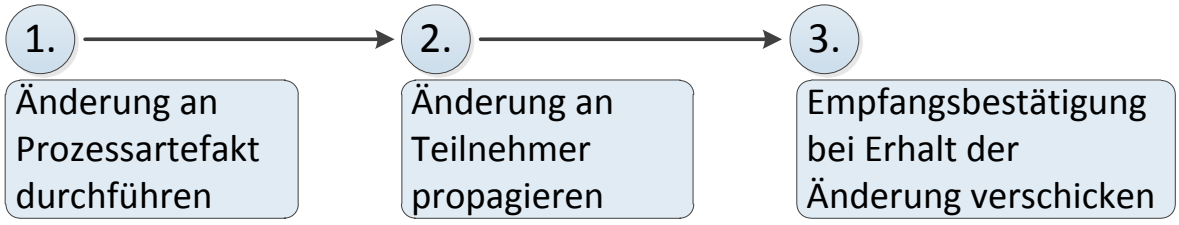


Bild 6.1: Ablauf beim Versand von Empfangsbestätigungen

6.2 Fachlicher Lösungsansatz

Dieser Abschnitt entwirft aufbauend auf den wissenschaftlichen Grundlagen einen fachlichen Lösungsansatz für das Versenden von Empfangsbestätigungen, sog. Acknowledgements (ACKs). Einerseits muss dazu eine geeignete Speicherstruktur entworfen werden, die es erlaubt, je Version eines Prozessartefakts die eintreffenden ACKs abzuspeichern und wieder auszulesen. Andererseits muss eine Acknowledgement-Nachricht konzipiert werden, die alle nötigen Informationen für eine eindeutige Zuordnung zu einer Version eines Prozessartefakts enthält und den Empfang einer Änderung quittiert.

Im weiteren Verlauf wird zuerst die Struktur der ausgetauschten Acknowledgement-Nachrichten entwickelt. Daran anschließend werden verschiedene Speicherstrukturen untersucht, um dann eine Auswahl für die in dieser Arbeit durchgeführte prototypische Implementierung zu treffen.

6.2.1 Nachrichtenformat

Lokale Änderungen eines Akteurs an einem Prozessartefakt werden durch den Versand von Nachrichten asynchron an die übrigen Akteure verteilt. Eine Änderung kann dabei die Modifikation einzelner Attribute oder Inhalte des betreffenden Prozessartefakts sein.

Das Ziel der Empfangsbestätigung ist es, dass der sendende Akteur darüber informiert wird, ob seine Änderung auf der gegenüberliegenden Seite eingetroffen ist. Das verwendete Nachrichtenformat muss also den sendenden Akteur als Attribut enthalten. Weiterhin kann die reine Information des Empfangs dahingehend erweitert werden, dass auch der Empfangszeitpunkt der Änderung, angegeben in physischer Zeit, in der Nachricht dokumentiert wird. Weiterhin muss eine eindeutige Zuordnung einer Empfangsbestätigung zu einer Änderung eines Prozessartefakts erfolgen. Dies wird durch die Verwendung des in Abschnitt 3.2.2 vorgestellten Identifizierers einer α-Card und seines logischen Zeitstempels (Version) erreicht.

Beim Erhalt einer Änderung soll das α-Doc also auf technischer Ebene bestätigen, dass es diese erhalten hat. Dies geschieht dadurch, dass es eine Empfangsbestätigung erzeugt und versendet wird, die folgende Attribute enthält (cf. auch Abbildung 6.2):

- den eindeutigen Bezeichner des Prozessartefakts,
- die logischer Zeitstempel des Prozessartefakts,
- den sendenden Akteur der Änderung und
- den physischen Empfangszeitpunkt der Änderung.

Verwendete physische Zeitstempel

Der physische Zeitstempel, der in der Nachricht verwendet wird, basiert auf der aktuellen Systemzeit zum Zeitpunkt des Erhalts der Änderung. Da es sich bei α-Flow um ein verteiltes System handelt, kann nicht sichergestellt werden, dass die Systemzeit auf allen Rechnern synchronisiert ist. Eine gemeinsame Zeitbasis, die es ermöglicht, getätigte Aktionen zu ordnen bzw. Aussagen über Zeiträume zwischen Aktionen treffen zu können,

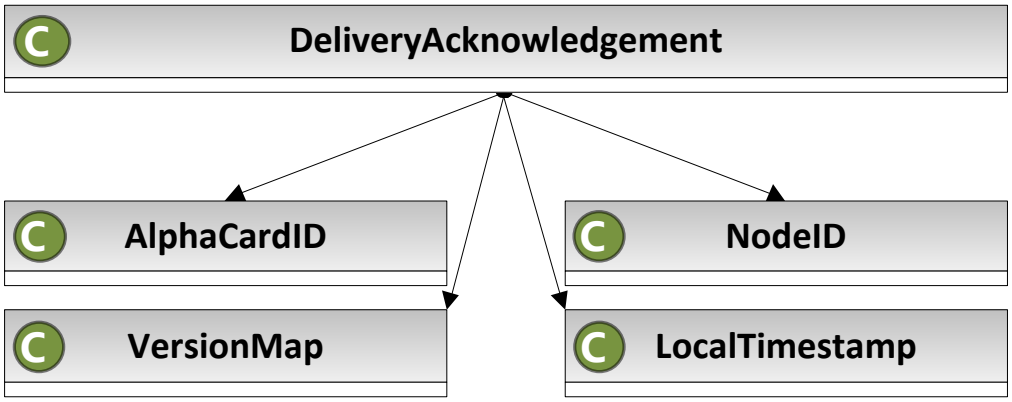


Bild 6.2: Empfangsbestätigung

wird im Fall des zusätzlichen Zeitstempels anders als bei der Ordnung der Versionen von α-Cards jedoch nicht benötigt. Der verwendete physische Zeitstempel soll hier lediglich einen Anhaltspunkt liefern und dient hauptsächlich dazu, eine Zusatzinformation in Form einer Erweiterung neben der Bestätigung des Erhalts der Änderung zu bieten. Es wird vielmehr davon ausgegangen, dass die Systemzeit mit akzeptabler Ungenauigkeit manuell gesetzt wurde. Im Rahmen der getroffenen Anforderungen ist dies als Genauigkeitskriterium ausreichend.

6.2.2 Entwurf einer Speicherstruktur

Die Verwaltung von Teilnehmerlisten zu den einzelnen Versionen eines Prozessartefakts muss bei jedem Teilnehmer vorhanden sein. Sie soll einen schnellen Zugriff auf die Information liefern, ob ein Teilnehmer eine bestimmte Version schon bestätigt hat.

Während der Bestätigung von Versionen kann es durchaus sein, dass Empfangsbestätigungen erst verschickt werden bzw. eintreffen, wenn schon neuere Versionen existieren. Daher muss es auch möglich sein, durch die Teilnehmerlisten zu navigieren und eintreffende Acknowledgements an der entsprechenden Position einzufügen.

Der Übersichtlichkeit halber sollen in der Editor-Komponente immer nur die Acknowledgements der aktuellsten Version eines Prozessartefakts angezeigt werden. Im Zusammenhang mit der Änderungshistorie spricht man hier auch vom sog. *HEAD* der Historie.

Die Abbildung 6.3 zeigt drei Akteure. Akteur *A* erstellt zwei neue Versionen *1* und *2*. Das Acknowledgement von Akteur *C* für Version *1* trifft bei Akteur *A* erst ein, nachdem dieser schon die zweite Version erzeugt hat. Der *HEAD* in der Versionshistorie ist damit

bereits eine Position weitergewandert. Zur Speicherung des Acknowledgements muss Akteur A in der Speicherstruktur an die entsprechende Stelle navigieren können.

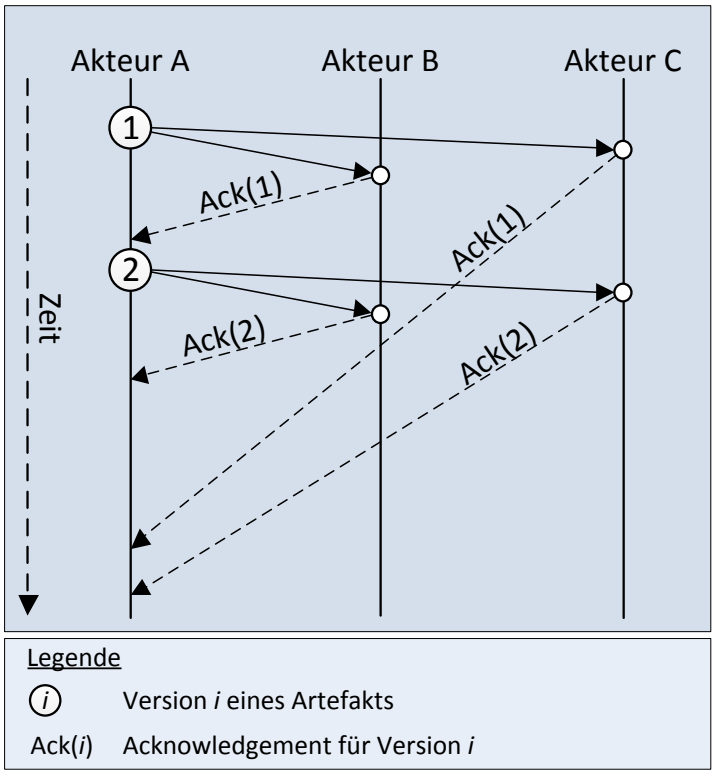


Bild 6.3: Verspätet eintreffende Acknowledgements

Prinzipiell gibt es verschiedenen Möglichkeiten bei der Wahl des Speicherungsorts der Teilnehmerlisten und auch bei deren Aufbau. Diese verschiedenen Möglichkeiten sollen im Folgenden näher betrachtet werden und anschließend aufgrund der gestellten Anforderungen eine Auswahl getroffen werden.

6.2.2.1 Kumulativ

Der Deskriptor einer α-Card kapselt alle für diese α-Card relevanten Informationen in Form von α-Adornments und die aktuelle Version der α-Card. Eine Änderung des Status eines Adornments kann dabei weitere Ereignisse auslösen.

Ein kumulativer Ansatz greift die Idee auf, alle zu einer α-Card gehörenden Informationen in dessen Deskriptor zu speichern. Darum erweitert er den Deskriptor einer α-Card um einen weiteren Parameter. Dieser enthält eine Liste, die wiederum für jede Version der α-Card Listen mit Acknowledgements für diese α-Card enthält. Bei jeder Erstellung einer neuen Version wird diese Liste von Listen um eine weitere Versionsliste erweitert; daher die Bezeichnung kumulativ.

Bei den bisher im Deskriptor enthaltenen Daten handelt es sich um Domänendaten, die im Verlauf der Behandlung von Relevanz sind und Einfluss auf diesen haben können. Die hinzugekommene Datenstruktur für die Acknowledgements kann als Bestandteil der Sekundärdaten beschrieben werden, da sie im Behandlungsverlauf eine sekundäre Rolle spielen und dieser auch ohne sie durchgeführt werden kann.

Abbildung 6.4 zeigt, wie bei der Erzeugung einer neuen Version eine neue Acknowledgement-Structure (*AS*) in der Acknowledgement-Structure-List (*ASList*) angelegt wird. Im weiteren Verlauf werden die eintreffenden Acknowledgements in dieser abgelegt.

In der ersten in Hydra erzeugten Version wird die Acknowledgement-Structure *AS1* angelegt. Bei der zweiten Version wird der Deskriptor (*D*) der α-Card geändert. Die neue Version bedingt die Erstellung von *AS2*. Eine Änderung des Payload (*P*) und damit der Erzeugung der dritten Version erweitert die *ASList* um *AS3*.

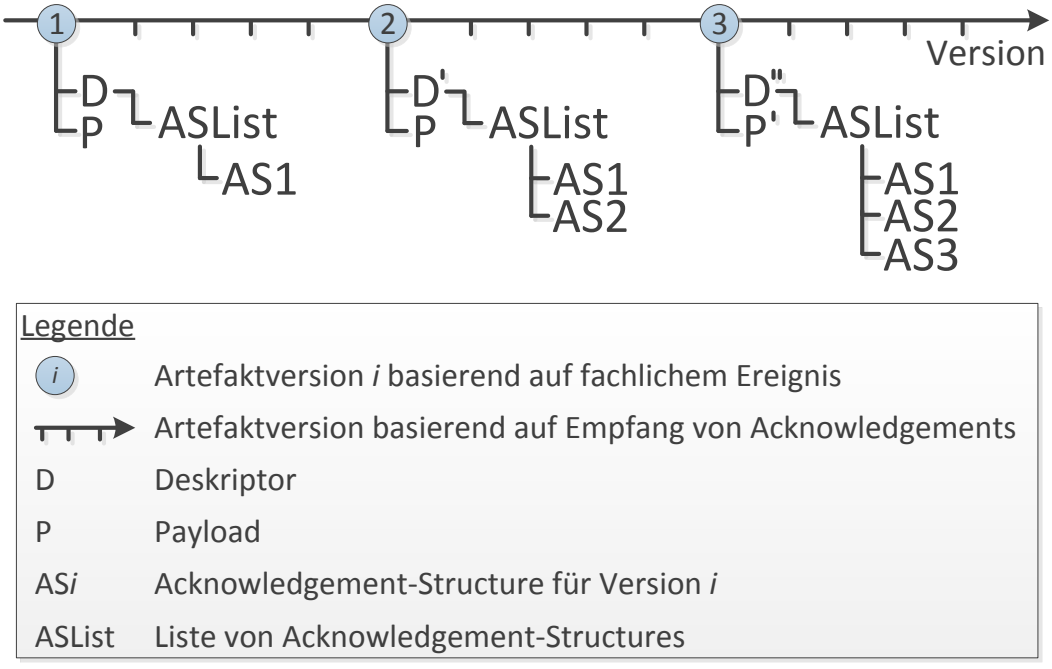


Bild 6.4: Empfangsbestätigungen kumulativ

Bei dieser Vorgehensweise gibt es wie beschrieben ein fachliches Ereignis: nämlich eine Änderung des Deskriptors oder des Payloads einer α-Card. Dieses Ereignis erzeugt eine neue Version der α-Card (in der Abbildung durch einen Kreis mit entsprechender Ziffer dargestellt). Die Version wird daraufhin an alle Prozessteilnehmer propagiert. Bei Erhalt dieser Version verschicken die Prozessteilnehmer wiederum ein Acknowledgement,

das den auslösenden Akteur über den Erhalt seiner Änderung und den Zeitpunkt der Verarbeitung der neuen Version informiert.

Neben dem fachlichen Ereignis der α-Card-Änderung kann ein weiteres Ereignis identifiziert werden: das Versenden der Empfangsbestätigungen beim Erhalt einer neuen α-Card-Version. Dieses muss entkoppelt vom fachlichen Ereignis betrachtet werden, denn es spielt sich auf einer anderen Ebene ab. Es handelt sich dabei vielmehr um ein Netzwerk-Ereignis als um ein fachliches Ereignis. Wenn eine Empfangsbestätigung zu einer Version eintrifft, muss die zu dieser passende *AS* gesucht werden und um das eingetroffene Acknowledgement erweitert werden. Diese Änderung des Deskriptors wird in Hydra zwar ebenfalls in einer neuen Version gespeichert (in der Abbildung durch Striche dargestellt), jedoch dürfen diese neuen Versionen, die lediglich durch veränderte Sekundärdaten hervorgerufen werden und die Domänendaten nicht berühren, nicht erneut Empfangsbestätigungen anfordern, da es sich nicht um ein fachliches Ereignis handelt, das wiederum Empfangsbestätigungen anfordern würde.

Ein Vorteil des kumulativen Ansatzes ist, dass die zu einer eintreffenden Empfangsbestätigung passende Version eines Artefakts einfach gefunden werden kann, denn es wird lediglich die Datenstruktur der *AS* im Deskriptor des Artefakts in der aktuellsten Version um den entsprechenden Eintrag erweitert.

Ein entscheidender Nachteil dieser Vorgehensweise ist hingegen, dass die Historie um Versionen proportional zur Teilnehmeranzahl anwächst. Je Version, die aus einem fachlichem Ereignis entsteht, werden im weiteren Verlauf genausoviele Versionen erzeugt wie es Teilnehmer gibt, denn jedes empfangene Acknowledgement erzeugt ebenenfalls eine neue Version. Dies kann insbesondere zu Skalierungsproblemen führen, wenn es zu einer großen Teilnehmerzahl kommt.

Auch die Unterscheidung von fachlichem und Netzwerk-Ereignis erfolgt aus dem Grund, dass für Netzwerk-Ereignisse keine Empfangsbestätigungen angefordert werden. Eine weitere Möglichkeit, das Anwachsen der Versionszahlen durch Netzwerk-Ereignisse zu verhindern, wäre das Überschreiben einer bestehenden Version. Auf der einen Seite werden hier nur die Sekundärdaten erweitert, auf der anderen Seite widerspricht dieses Vorgehen aber dem Grundsatz, jede Änderung nachvollziehbar zu dokumentieren.

6.2.2.2 Deskriptor-versionsindividuell

Die deskriptor-versionsindividuelle Variante verzichtet auf das Mitführen aller *AS* bei neuen Versionen. Hier wird für jede Version die dazugehörige *AS* gespeichert. Sobald ein neues Acknowledgement eintrifft, wird in der Historie an die dazugehörige Version

gesprungen und das *ACK* in die *AS* eingefügt. Auf diese Weise wird es über die Zeit gefüllt.

Dieses Vorgehen hat den Vorteil, dass die Acknowledgements nicht redundant gespeichert werden müssen. Damit wächst die Datenmenge nicht unnötig an und es werden Inkonsistenzen vermieden.

Nichtsdestoweniger existieren weiterhin die in Abschnitt 6.2.2.1 genannten Nachteile. Außerdem muss bei diesem Ansatz in der Versionshistorie nun zur Version navigiert werden, die zur Empfangsbestätigung gehört.

Bild 6.5: Empfangsbestätigungen deskriptor-versionsindividuell

6.2.2.3 Hydra-VVS

Eine Alternative zu den bisher genannten Ansätzen wäre die Erweiterung von Hydra-VVS um geeignete Schnittstellen für die Verwaltung von Empfangsbestätigungen. Abbildung 6.6 zeigt die Trennung von α-Cards und den in Hydra diesen zugeordneten *ASs*.

Die α-Cards werden wie bisher in die Historie eingepflegt. Hydra wird dahingehend erweitert, dass es zu jeder Version ein *AS* speichern kann. Über Schnittstellen werden Hydra dann die eintreffenden Empfangsbestätigungen übergeben. Diese fügt Hydra dann in die dazu passende *AS* ein und erzeugt eine neue Version.

Der Vorteil dieses Ansatzes liegt darin, dass die Speicherstruktur für die Acknowledgements einer Version direkt an diese angehängt werden. Dadurch wird eine geringe Suchtiefe für den richtigen Ablageort erreicht. Desweiteren kommt es analog zu dem in Abschnitt 6.2.2.2 beschriebenen Vorgehen zu keiner Redundanz bei der Speicherung.

Diese Variante hat aber den Nachteil, dass das bisher generisch entwickelte Hydra, um zusätzliche α-Flow-spezifische Elemente erweitert werden muss. Neue Schnittstellen und Datenstrukturen müssten tief in das Versionsverwaltungssystem integriert werden. Damit verliert es seine Unabhängigkeit von der α-Flow-Implementierung.

Zudem ist das Ändern von Versionen in einem Versionsverwaltungssystem sehr aufwändig, weil dazu eine komplette Kopie der Version aus dem Repository geladen werden muss, um diese nach der Änderung wieder zurückzuschreiben.

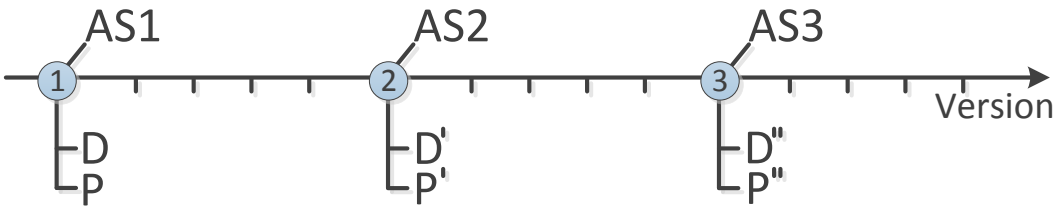


Bild 6.6: Empfangsbestätigungen bei Hydra-VVS

6.2.2.4 Eigene Datenstruktur

Zu den bisher genannten Varianten, die Teilnehmerlisten zu verwalten, besteht noch die Möglichkeit, eine eigene von der Versionsverwaltung losgelöste Datenstruktur zur Speicherung der Empfangsbestätigungen aufzubauen.

Die hier verwendete Datenstruktur ist prinzipiell analog wie die in Abschnitt 6.2.2.1 beschriebene kumulative Datenstruktur aufgebaut. Der Unterschied liegt aber in der Form der Speicherung. Im Fall der eigenen Datenstruktur löst die externe Ablage diese von der Versionierung. Eintreffende Empfangsbestätigungen werden lokal abgespeichert, aber nicht durch Versionen mit den anderen Teilnehmern ausgetauscht. Damit ist diese auch nur lokal bei jedem Teilnehmer individuell vorhanden und wird nicht mit den anderen Prozessteilnehmern synchronisiert. Dieses Vorgehen kann damit gerechtfertigt werden, dass ein Verlust dieser Datenstruktur für das Kollektiv irrelevant ist. Es ist nur knotenspezifisch. Auch trotz des Verlustes der Acknowledgement-Structure lässt sich am α-Doc weiterarbeiten.

Dadurch, dass die *AS* nicht in einer α-Card gespeichert werden, wird bewusst darauf verzichtet, diese zu synchronisieren. Es entsteht aber auch deutlich weniger Aufwand, der ansonsten bei jedem neu eintreffendem Acknowledgement durch die Synchronisierung entstanden wäre. Dieses Vorgehen könnte aber ggf. durch einen Schalter für die Synchronisierung gerechtfertigt werden (cf. Kapitel 8).

6.2.2.5 Zusammenfassung

Insbesondere das Auffinden der zu einem Acknowledgement passenden Version in der Versionshistorie führt zu einem großen Aufwand. Denn um eine Version zu finden, muss der gesamte Systempfad traversiert werden. In der momentan vorliegenden Implementierung von Hydra muss dazu die *Working-Copy* der jeweiligen Version auf die Festplatte geschrieben werden. Die unter Umständen sehr großen Payloads in den α-Cards würden dabei zu einem enormen Schreibaufwand führen. Weil es dazu bisher keine Alternative

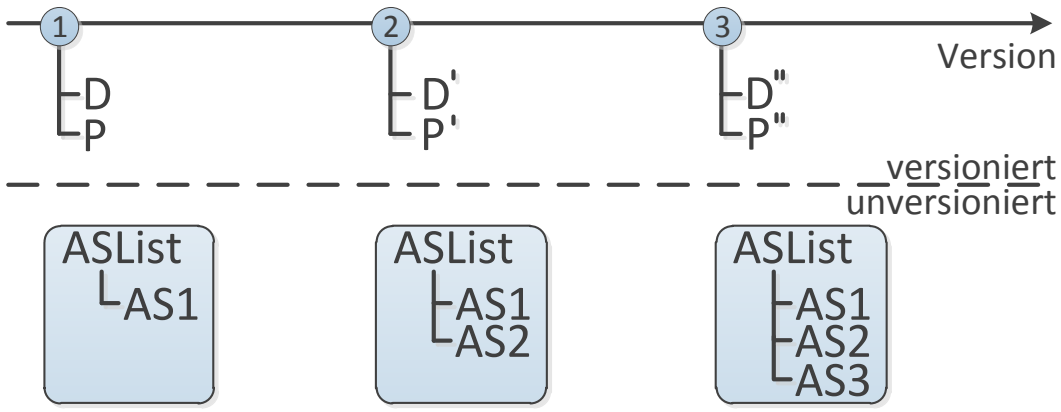


Bild 6.7: Empfangsbestätigungen bei eigener Datenstruktur

gibt, wird im weiteren Verlauf dieser Arbeit vorerst die zuletzt vorgestellte Variante einer eigenständigen Datenstruktur verfolgt. Mit dieser ist es möglich, mit einer Schlüssel-Wert-Struktur wahlfrei auf einzelne Versionen zuzugreifen, ohne dabei die Domänendaten zu berühren.

Des Weiteren wird davon ausgegangen, dass die lokale Datenstrukturen zur Speicherung von Acknowledgements nicht mit den anderen Akteuren synchronisiert werden muss. Die Speicherung von Acknowledgements ist für die Nachvollziehbarkeit vor Gericht durchaus wichtig, doch kann das Kollektiv an Akteuren auch bei deren Verlust den Behandlungsprozess fortführen.

Die jeweiligen Vor- und Nachteile der einzelnen Varianten sind in Tabelle 6.1 nochmals gegenübergestellt. Es wird deutlich, dass beim Ansatz der eigenen Datenstruktur momentan die Vorteile überwiegen.

Entwurfsvariante	Vorteile	Nachteile
kumulativ	geringer Suchaufwand für passende Version	Skalierungsproblem und großes Datenaufkommen
deskriptor-versionsindividuell	keine Redundanz der Daten	Aufwand bei Navigation zur passenden Version
Hydra-VVS	keine Redundanz der Daten und geringe Suchtiefe	Generik von Hydra geht verloren und größerer Schreibaufwand
eigene Datenstruktur	kein Synchronisationsaufwand und wahlfreier Zugriff	Verlust der Daten möglich

Tabelle 6.1: Gegenüberstellung der Entwurfsvarianten

6.3 Technisches Lösungskonzept

Dieses Kapitel stellt die einzelnen Bestandteile des Lösungskonzepts vor, die zum Versenden und Verwalten von Empfangsbestätigungen nötig sind. Zu Beginn wird die Erweiterung der Konfiguration des α-Doc beschrieben, die es ermöglicht, zum Prozessstart die Funktionalität der Empfangsbestätigungen zu aktivieren bzw. zu deaktivieren. Im Anschluss wird die Speicherstruktur erläutert, in der zu den einzelnen Versionen eines Prozessartefakts die Acknowledgements abgelegt werden. Daran anschließend wird die Nachricht als solches betrachtet. Abschließend werden die Anpassungen an der α-Editor-Komponente vorgestellt.

6.3.1 Erweiterung der AlphadocConfig

Das Verfahren, Empfangsbestätigungen für neue Versionen zu erstellen, erzeugt ein hohes Nachrichtenaufkommen. Es kann daher unter gewissen Umständen gewünscht sein, darauf zu verzichten. Vor diesem Hintergrund wurde die Klasse `AlphadocConfig` um ein weiteres Attribut ergänzt. Die `boolean`-Variable `acknowledgeDelivery` gibt an, ob Acknowledgements verschickt werden sollen. Sie wird bei der initialen Erstellung des α-Doc durch den Prozessinitiator mit einem Wert belegt. Die eingeladenen Prozessteilnehmer haben auf sie nur lesenden Zugriff und auch der Prozessinitiator kann im weiteren Verlauf des Behandlungsprozesses den Wert nicht ändern.

Der Wert dieser Variable wird beim Erhalt einer neuen Version geprüft. Je nach Ausprägung wird eine Empfangsbestätigung verschickt oder dieser Schritt übergangen.

6.3.2 Erweiterung des α-Doc

Die Klasse `AlphaDoc` wird um eine Liste von `AcknowledgementStructures` erweitert. Abbildung 6.8 zeigt diese Erweiterung und die Schnittstellen, die den Zugriff auf die Liste von Acknowledgements bieten.

Die Methode `getAS(AlphaCardID, VersionMap)` liefert die zu den übergebenen Parametern passende `AcknowledgementStructure` zurück. Sollte diese nicht existieren, wird eine neue `AcknowledgementStructure` mit den übergebenen Parametern erstellt, in die Liste von Acknowledgements eingefügt und im Anschluss diese zurückgegeben. Die für die Anzeige relevante *HEAD*-Version einer `AlphaCardID` kann mit der Methode `getHeadAs` abgefragt werden. Diese wählt aus der Liste aller Acknowledgements, die zu einer `AlphaCardID` gehören, diejenige aus, die die größte Version aufweist.

6.3.3 Acknowledgement-Structure

Die Klasse `AcknowledgementStructure` enthält die zu einem Prozessartefakt mit einer bestimmten Version eingetroffenen Acknowledgements. Eine Übersicht der Attribute und Methoden liefert Abbildung 6.8. Die Klasse `AcknowledgementStructure` kapselt eine Menge von Acknowledgements (`AcknowledgementStructureItem`). Beim Eintreffen neuer Acknowledgements können sie dieser Menge mit der Methode `addItem(AcknowledgementStructureItem)` hinzugefügt werden. Die Methode `getAcknowledgements()` wiederum gibt die komplette Menge an Acknowledgements zurück.

Bei der Speicherstruktur für die Acknowledgements handelt es sich um ein `HashSet`, da es durchaus möglich sein kann, dass es für eine α-Card einer bestimmten Version mehrere Acknowledgements eines Teilnehmers gibt. Dies ist insbesondere dann der Fall, wenn der entsprechende Akteur sein α-Doc an mehreren Arbeitsplätzen repliziert betrachtet (cf. [Wah11, Seite 59]).

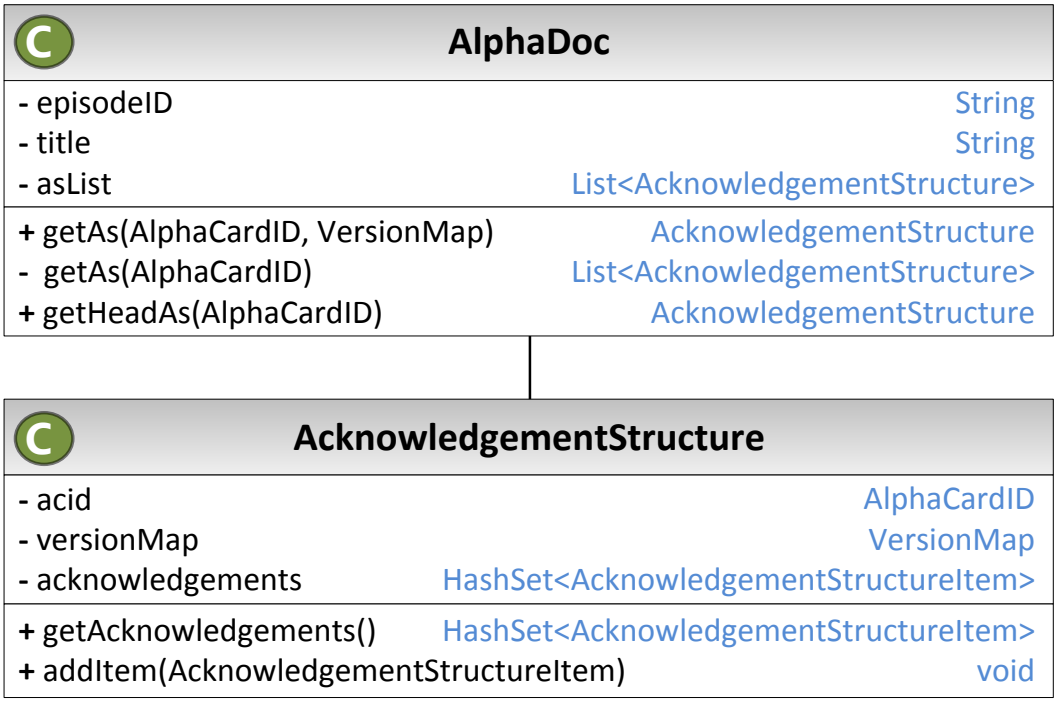


Bild 6.8: Struktur der Klasse `AcknowledgementStructure`

6.3.4 Acknowledgement-Structure-Item

Die Klasse `AcknowledgementStructureItem` enthält die Sender- und Empfänger-spezifischen Daten zu einem Acknowledgement. Dazu zählen die Sender- und Empfänger-`NodeID`, sowie die dazugehörenden Zeitstempel der Verarbeitung. Die Struktur der Klasse ist in Abbildung 6.9 dargestellt. Ein `AcknowledgementStructureItem` ist quasi ein Key-Value-Paar:

$key = \{NodeID_{Sender}, NodeID_{Empfänger}\}, value = \left\{TS^{local}_{Sender}, TS^{local}_{Empfänger}\right\}$

Für den Schlüssel aus Sender- und Empfänger-`NodeID` gibt es das Wertepaar aus Sender- und Empfänger-Zeitstempel zurück.

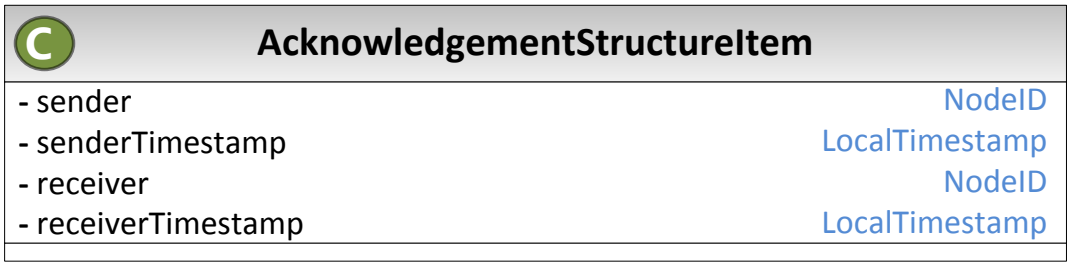


Bild 6.9: Struktur der Klasse `AcknowledgementStructureItem`

6.3.5 Delivery-Acknowledgement-Ereignis

Das Ereignis `DeliveryAcknowledgement` wird über die Komponente α-Overnet übertragen und kapselt die Informationen, die zur Empfangsbestätigung einer Nachricht benötigt werden. Dazu zählen die `AlphaCardID` und die `VersionMap`, die eine Änderung eines Prozessartefakts eindeutig kennzeichnen, sowie der dazugehörige Zeitstempel des Empfangs der Änderung (`senderTimestamp`) sowie den Identifizierer des Senders der Empfangsbestätigung (`sender`) (cf. Abbildung 6.10).

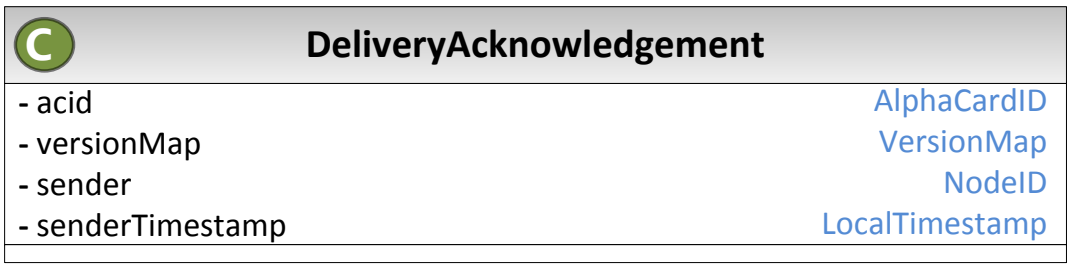


Bild 6.10: Struktur der Klasse `DeliveryAcknowledgement`

6.3.6 Empfangsbestätigung von Versionen

Die Abbildung 6.11 gibt einen Überblick über den Ablauf, der durch eine Änderung an einem Prozessartefakt hervorgerufen wird. Zuerst wird durch Akteur A eine Änderung durchgeführt und diese an die Teilnehmer B und C propagiert. Beispielhaft wird nun für Akteur B der Empfang dieser Änderung quittiert. Dazu erzeugt B ein Delivery-Acknowledgement-Ereignis und sendet es an die übrigen Teilnehmer und verarbeitet es auch selbst (cf. Abbildung 6.11(Schritt 2)). Die übrigen Teilnehmer wiederum erhalten diese Nachricht und verarbeiten sie entsprechend.

Zusammenfassend wird die Empfangsbestätigung also durch ein Ereignis ausgelöst, woraufhin sie dann versandt wird:

1. Eine Änderung an einem Prozessartefakt wird durchgeführt und propagiert.
2. Das α-Doc bestätigt den Empfang der Änderung an alle anderen Prozessteilnehmer.

In den folgenden Kapiteln wird zuerst dieser Ablauf genauer beschrieben und dann die Klasse `AcknowledgementUtility` näher betrachtet.

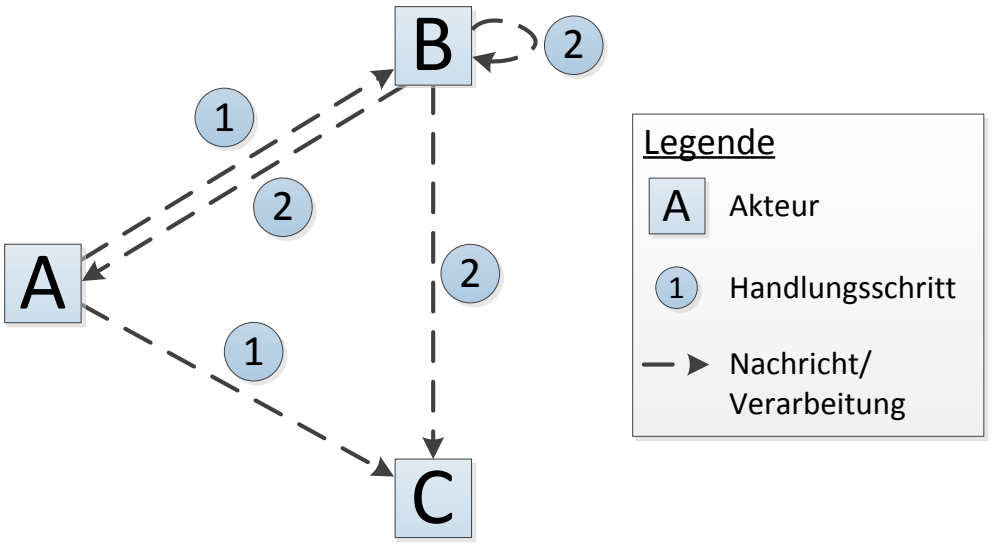


Bild 6.11: Ablauf bei der Bestätigung einer neuen Version

6.3.6.1 Detaillierter Ablauf bei der Erzeugung und Verarbeitung von Acknowledgements

Die Generierung einer Empfangsbestätigung wird durch den Erhalt einer neuen Version eines Prozessartefakts hervorgerufen. Im Folgenden wird der eigentliche Ablauf detaillierter beschrieben.

In der Regelbibliothek wird im Verlauf der Verarbeitung einer neuen Prozessartefakt-Version durch den Aufruf der Methode `generateDeliveryAcknowledgement()` der Klasse `AcknowledgementUtility` ein Acknowledgement erzeugt und anschließend in die Regelbibliothek eingefügt. Ein weiterer Schritt ist die Initialisierung der Speicherungsstruktur mit Platzhaltern für Acknowledgements aller momentanen Prozessteilnehmern (`initializeAcknowledgementStructure()`), die von der `AcknowledgementUtility` an die `AlphaPropsFacade` zur eigentlichen Verarbeitung delegiert wird. Das in die Regelbibliothek eingefügte `DeliveryAcknowledgement` wird mit Hilfe von α-Overnet an die übrigen Prozessteilnehmer versandt. Diesen Ablauf der Erstellung und Verbreitung von Acknowledgements beschreibt Abbildung 6.12.

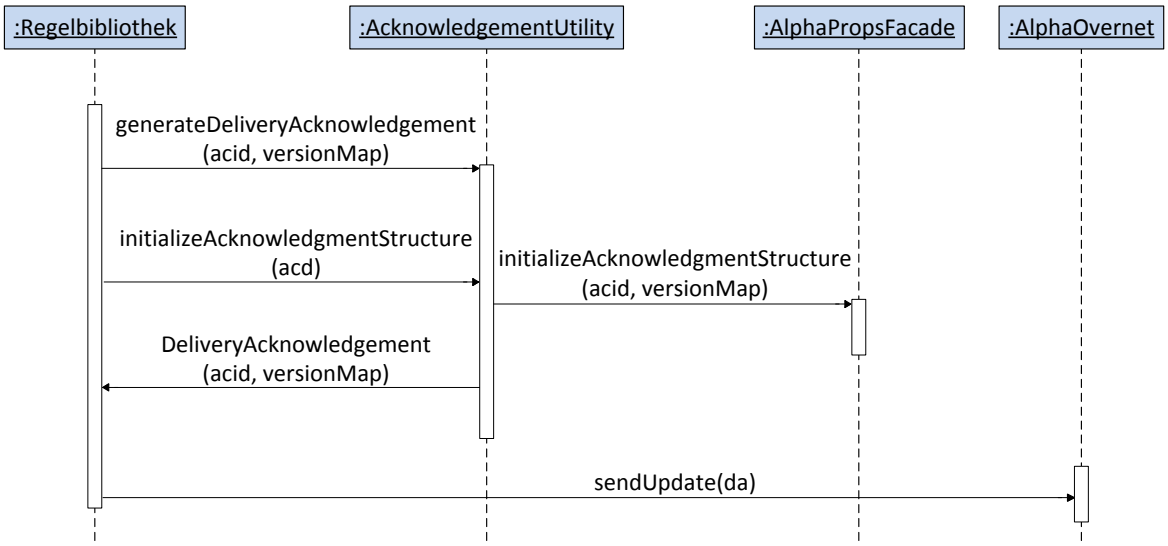


Bild 6.12: Ablauf bei der Erstellung und Verbreitung einer Empfangsbestätigung

Der Erhalt einer Empfangsbestätigung durch α-Overnet führt in der Regelbibliothek zur Ausführung einer Regel, die die Verarbeitung des Delivery-Acknowledgements durch die `AcknowledgementUtility` anstößt (cf. Abbildung 6.13). Diese wiederum weist daraufhin die `AlphaPropsFacade` an, die `AcknowledgementStructure` mit den entsprechenden Werten zu aktualisieren.

6.3.6.2 Schnittstelle zur Erzeugung und Verarbeitung von Acknowledgements

Die `AcknowledgementUtility` kapselt die zur Erzeugung und Verarbeitung von `DeliveryAcknowledgements` benötigten Funktionen und erhöht damit die Wartbarkeit

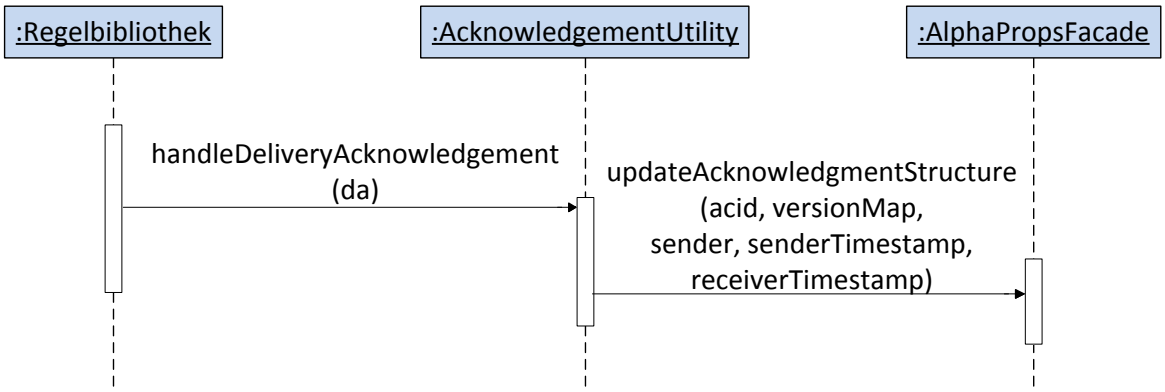


Bild 6.13: Ablauf beim Erhalt einer Empfangsbestätigung

der Anwendung. Diese Klasse kommuniziert zum einen mit der regelbasierten Bibliothek und zum anderen mit der Fassade von α-Properties.

Die Abbildung 6.14 zeigt die Schnittstelle der Klasse `AcknowledgementUtility`. Die Methode `generateDeliveryAcknowledgement` erzeugt ein neues Acknowledgement, versieht es mit der übergebenen `AlphaCardID` und der Version. Gleichzeitig wird ein aktueller Zeitstempel generiert und auch der lokale Akteur eingefügt. Ein eintreffendes Acknowledgement wird der Methode `handleDeliveryAcknowledgement` übergeben und von dieser entsprechend behandelt. Die Methode `initializeAcknowledgementStructure` initialisiert für das übergebene Prozessartefakt in seiner vorliegenden Version die Speicherstruktur, die zur Ablage der eintreffenden Acknowledgements nötig ist. Dazu wird eine neue `AcknowledgementStructure` erzeugt und dessen Liste mit ansonsten leeren `AcknowledgementStructureItems` für jeden Prozessteilnehmer gefüllt.

Dies hat insbesondere den Sinn, dass für jeden teilnehmenden Akteur ein Platzhalter mit leeren Zeitstempeln vorhanden ist. Sobald diese Zeitstempel im weiteren Verlauf beim Eintreffen von Acknowledgements gefüllt worden sind, ist sichergestellt, dass das entsprechende Acknowledgement erhalten wurde.

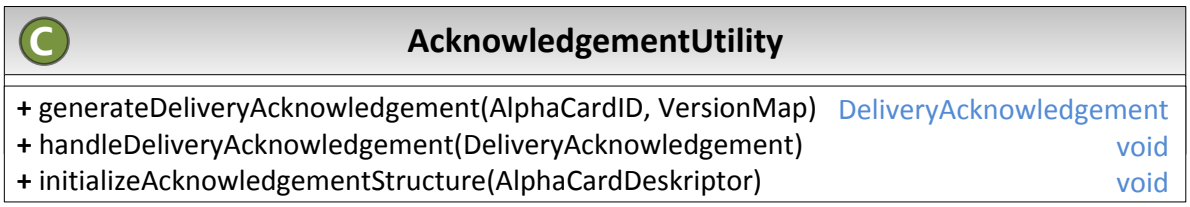


Bild 6.14: Struktur der Klasse `AcknowledgementUtility`

6.3.7 Anpassung der Editor-Komponente

Die Editor-Komponente wird dahingehend angepasst, dass sie die zu einer Version eines Prozessartefakts gehörenden Acknowledgements übersichtlich anzeigt. Dabei ist für den Nutzer weniger relevant, zu welchen Zeitpunkt der Erhalt einer Änderung stattgefunden hat, sondern vielmehr, dass dies überhaupt geschehen ist. Deshalb wird bei der Anzeige der Acknowledgements lediglich unterschieden, ob dieses für einen bestimmten Akteur eingetroffen ist oder nicht.

Wenn im α-Editor vom Benutzer eine α-Card ausgewählt wurde, dann kann zu dieser eine graphische Übersicht zu den Acknowledgements eingeblendet werden (cf. Abbildung 6.15). Im unteren rechten Teil des Fensters ist eine kreisförmige Anordnung der teilnehmenden Akteure eingeblendet. Das Symbol eines Akteurs, dessen Empfangsbestätigung bereits eingetroffen und verarbeitet worden ist, erscheint grün. Symbole von Akteuren ohne bisherige Empfangsbestätigung sind grau dargestellt.

Die Abbildung 6.16 stellt nochmal eine Vergrößerung des Acknowledgement-Bereichs aus Abbildung 6.15 dar. Im Teil a) ist zu erkennen, dass bisher nur der Akteur *Gyn A* den Erhalt dieser Version des Prozessartefakts bestätigt hat, denn nur *Gyn A* hat ein grünes Symbol; in Teil b) haben dann alle Akteure den Erhalt bestätigt.

Bis zu einer Anzahl von zehn Teilnehmern werden diese, wie in Abbildung 6.16 dargestellt, im Kreis angeordnet. Falls darüber hinaus noch mehr Akteure am Prozess beteiligt sein sollten, wird auf diese durch einen Hinweis verwiesen. Die Annahme, dass gewöhnlich nicht mehr als zehn Teilnehmer am Prozess mitarbeiten, wurde aus dem Anwendungsfall aus Abschnitt 3.4 abgeleitet.

Ein Akteur kann ein α-Doc an mehreren Arbeitsplätzen parallel bearbeiten. Daraus resultieren wiederholte Acknowledgements einer Änderung durch denselben Akteur. Zur Anzeige gebracht wird jedoch nur das erste eingetroffene Acknowledgement. Relevant ist also, ob die Änderung beim einzelnen `contributor` mindestens einmal angekommen ist, und nicht, ob sie jeweils bei dem einzelnen `Endpoint` angekommen ist (*is-at-any-endpoint*).

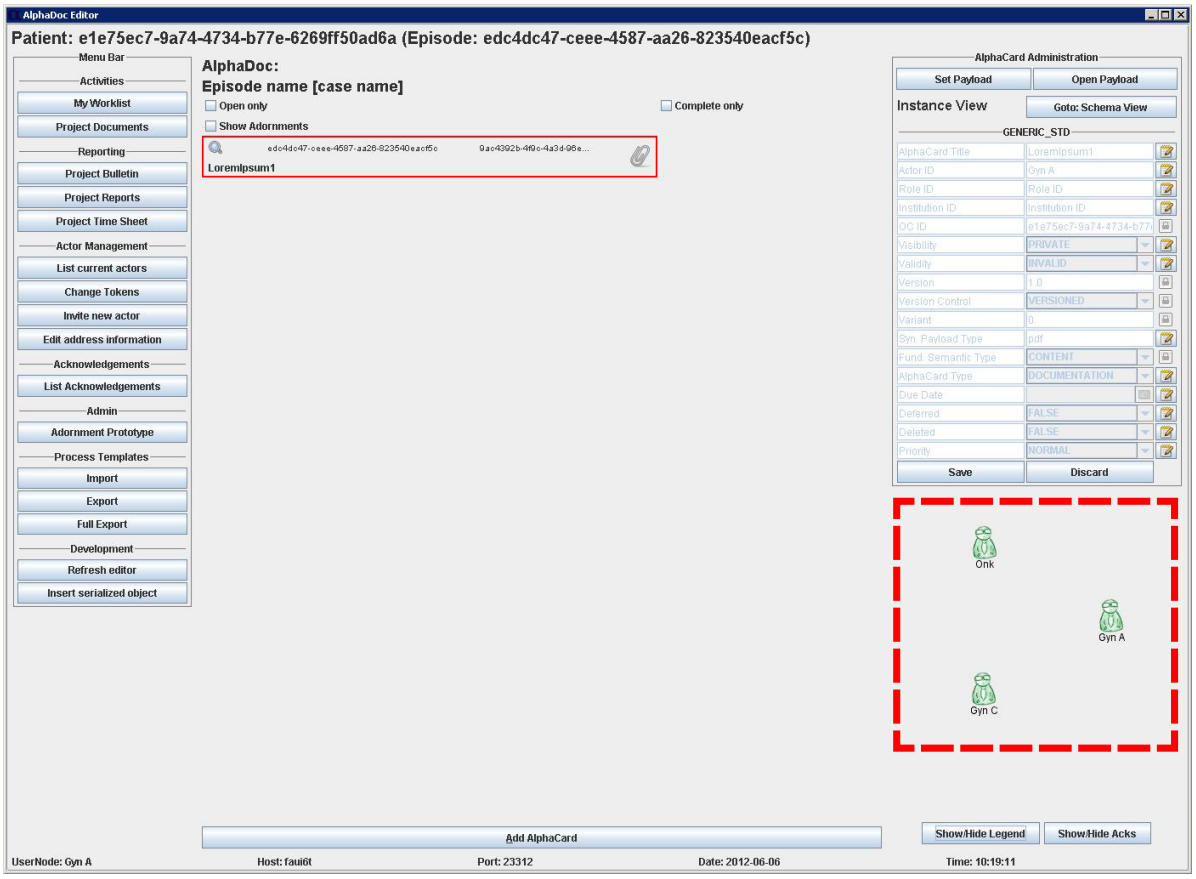


Bild 6.15: Ansicht des α-Doc-Fensters mit eingeblendeten Acknowledgements

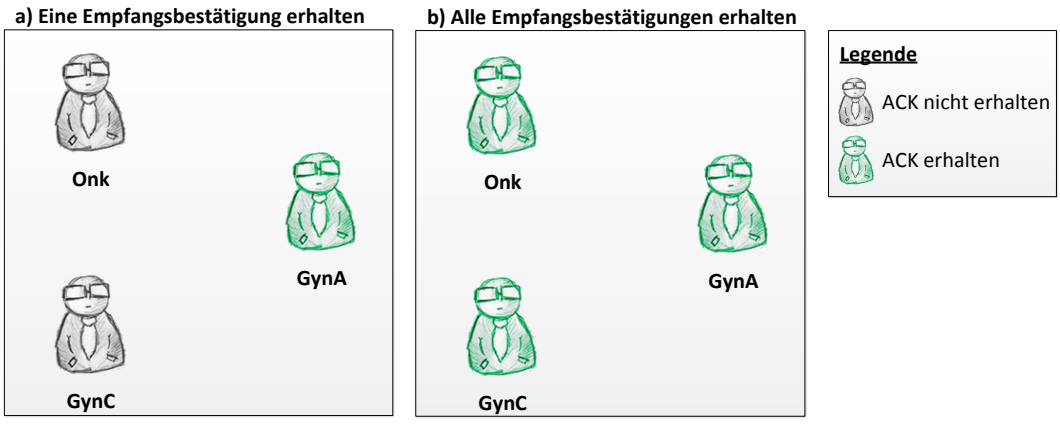


Bild 6.16: Vergrößerung des Acknowledgements-Bereichs

6.4 Zusammenfassung

Der vorgestellte fachliche Lösungsansatz kann den Akteur, der eine Änderung an einem Prozessartefakt durchgeführt hat, darüber informieren, ob diese Änderung bei den einzelnen anderen Akteuren angekommen ist. Das entwickelte Nachrichtenformat und die Verwendung lokaler Zeitstempel erlauben die Sammlung der benötigten Informationen. Verschiedene Speicherungsvarianten für die eintreffenden Acknowledgements wurden vorgestellt. Die eintreffenden Acknowledgements werden jedoch in einer eigenen Datenstruktur gespeichert. Dies ist aus Gründen eines performanteren Zugriffs sinnvoll und weil für die lokale Acknowledgement-Datenstruktur momentan keine Synchronisierung mit den anderen Teilnehmern nötig ist. Für eine übersichtliche Darstellung wurde die Editor-Komponente entsprechend um Anzeigekonzepte für die Acknowledgements erweitert.

7 Evaluation

Dieses Kapitel beschreibt eine quantitative Evaluation eines α-Docs im Hinblick auf die Anzahl und Größe der ausgetauschten Nachrichten, die während einer beispielhaft durchgespielten Behandlungsepisode auftreten. Zu Beginn werden die verschiedenen Nachrichtentypen untersucht und diese in Gruppen gegliedert. Außerdem werden Annahmen zu den verwendeten Payload-Dateien getroffen. Daran anschließend wird die Behandlungsepisode aus Abschnitt 3.4 aufgegriffen und für das Vorgehen während der Evaluation detailliert beschrieben. Es werden verschiedene Varianten gebildet, die ein unterschiedliches Vorgehen bei der Benutzung des α-Doc abbilden. Schließlich werden die Ergebnisse der Evaluation analysiert und bewertet.

7.1 Klassifikation der Nachrichten

Im Zusammenhang des Informationsaustauschs konnten verschiedene Typen von Nachrichten unterschieden werden (cf. Tabelle 7.1). Dies sind zum einen Nachrichten, die während des Ablaufs des Beitrittsprotokolls ausgetauscht werden. Dazu zählen die *parallele Rückmeldung* und die *parallele Synchronisation* sowie die *sequentielle Rückmeldung* und die *sequentielle Synchronisation*. Im Rahmen der Evaluation traten allerdings nur die `SequentialJoinCallback`- und `SequentialJoinSynchronisation`-Nachrichten auf, da die Teilnehmer nicht parallel beigetreten sind.

Weiterhin konnten Nachrichten identifiziert werden, die eine Änderung an einer Content- oder Coordination-α-Card hervorrufen bzw. eine solche erzeugen. Zu diesem Nachrichtentyp gehören das `AddAlphaCardEvent`, das `ChangeAlphaCardDescriptorEvent` und das `ChangePayloadEvent`. Desweiteren gibt es Nachrichten, die eine Empfangsbestätigung übermitteln (`DeliveryAcknowledgement`), und solche, die eine Token-Weitergabe propagieren (`TokenPropagation`).

Im Rahmen der Evaluation wurden Annahmen über die Dateien getroffen, die als Payload der α-Cards verwendet wurden (cf. Tabelle 7.2). Als in der Praxis gängige

Nachrichtentyp	Nachricht
Änderung an Content- oder Coordination-α-Card	`AddAlphaCardEvent`
	`ChangeAlphaCardDescriptorEvent`
	`ChangePayloadEvent`
Beitrittsprotokoll	`SequentialJoinCallback`
	`SequentialJoinSynchronisation`
	`ParallelJoinCallback`
	`ParallelJoinSynchronisation`
Empfangsbestätigung	`DeliveryAcknowledgement`
Token-Weitergabe	`TokenPropagation`

Tabelle 7.1: Identifizierte Nachrichtentypen

Dateiformate haben sich sowohl das PDF als auch das DOC-Format[1] für Textdokumente erwiesen. Bilder und eingescannte Berichte bzw. Dokumente werden häufig im JPEG-Format gespeichert. Auch für die beispielhaft verwendeten Dateien wurde auf diese Formate zurückgegriffen. Die einzelnen Dokumententypen sind mit dem jeweiligen Dateiformat und der Dateigröße in Tabelle 7.2 gegenübergestellt. Die Angabe der Dateigröße soll auch die später dokumentierten Nachrichtengrößen relativieren helfen.

Dokumententyp	Dateiformat	Dateigröße
Anamnesebericht	PDF	25 KB
Arztbrief	PDF	56 KB
Mammographiebericht	JPEG	56 KB
Sonographiebericht	DOCX	16 KB
Überweisungsschein	JPEG	17 KB

Tabelle 7.2: Verwendete Beispiele für Payload-Dateien

7.2 Verwendeter Anwendungsfall

Der Evaluation liegt das Anwendungsszenario aus Abschnitt 3.4 zugrunde. Dieser Anwendungsfall beschreibt den Ablauf, in dem eine Patientin mit einem auffälligen Knoten in der Brust erstmalig bei ihrem Gynäkologen vorstellig wird.

Aus Gründen der Vereinfachung wird der Anwendungsfall allerdings nur bis zur Ergebnisdokumentation des Radiologen exemplarisch durchgeführt. Im Folgenden wird

1 Seit dem Jahr 2007 wird stattdessen häufig der auf der XML basierende Standard DOCX verwendet (cf. [Int08]).

beschrieben, welche Schritte im Einzelnen ausgeführt wurden, um den Anwendungsfall in eine α-Episode zu überführen. Dabei wurde keine „Drag and Drop“-Funktionalität verwendet und alle Schritte einzeln ausgeführt.

Der auf α-Flow übertragene Ablauf der einzelnen Handlungen gliedert sich dabei wie folgt:

1. **Erstellung des α-Doc durch Gyn^A mit Eingabe der Patientendaten und Dokumentation des Anamnesebericht**
   Dabei wird die Injector-Funktionalität[1] des Editors verwendet, indem der Anamnesebericht als Payload für die erste α-Content-Card übergeben wird. Darauf werden die Adornments *Visibility* und *Validity* auf sichtbar bzw. gültig gesetzt.
2. **Report der Sonographie durch Gyn^A**
   Gyn^A erstellt eine neue α-Card und anschließend wird dieser als Payload der Sonographiebericht angehängt. Dann wird sie auf sichtbar und gültig gesetzt.
3. **Überweisung an Rad^A durch Gyn^A**
   Gyn^A erstellt ein Paar von α-Cards. Die erste enthält als Payload den Überweisungsträger, die zweite ist der Platzhalter, in den später Rad^A seine Ergebnisse ablegt. Nachdem die beiden α-Cards zuerst erzeugt wurden, wird der Überweisung das entsprechende Dokument angehängt und diese α-Card dann sichtbar und gültig geschaltet. Daran anschließend wird der verantwortliche Arzt für beide α-Cards auf Rad^A gesetzt, damit dieser den Auftrag zugewiesen bekommt.
4. **Beitritt von Rad^A zum α-Doc**
   Dazu erstellt Gyn^A eine Kopie seines α-Doc und lässt diese Rad^A zukommen. Nachdem dieser seine Kontaktdaten eingegeben hat, beginnt das Beitrittsprotokoll.
5. **Report der Mammographie-Ergebnisse durch Rad^A**
   Rad^A dokumentiert seine Ergebnisse in der für ihn erstellten Mammographie-α-Card. Anschließend setzt er sie auf sichtbar und gültig.

Dieses grundlegende Vorgehen wird noch einmal durch zwei Varianten erweitert, die sich darin unterscheiden, wann die Kopie des α-Doc, die zur Einladung von Rad^A benötigt wird, erstellt wird:

- Variante ***Start***: Die Kopie des α-Doc erfolgt gleich zu Beginn der Episode direkt nach dem Erstellen des α-Doc durch das Einfügen des Anamnesebericht.

1 cf. [Han10].

- Variante ***Join***: Die Kopie des α-Doc wird zu dem Zeitpunkt erstellt, an dem Rad^A der Episode beitreten soll.

In der Variante *Start* enthält die Kopie des α-Doc keine weiteren Änderungen, da sie direkt nach dem Öffnen des neu erstellten α-Doc erstellt wurde. Im Fall von Variante *Join* findet die Kopie des α-Doc zu einem späteren Moment statt. Dieser Zeitpunkt liegt nämlich nach der Erstellung des α-Card-Paars mit der Überweisung an den Radiologen und dem Platzhalter für den Mammographieberichts. In dieser Variante sind bereits weitere Änderungen in der Kopie des α-Doc enthalten. Daher wird erwartet, dass der Synchronisierungsaufwand während des Beitritts von Rad^A in dieser Variante geringer ist als in Variante *Start*.

Außerdem wird bei der Evaluation unterschieden, ob Acknowledgements verwendet werden (Variante *Start.Ack* bzw. *Join.Ack*) oder nicht (Variante *Start.UA* bzw. *Join.UA*). Einen Überblick über die verschiedenen Varianten der Evaluation liefert Tabelle 7.3.

		Acknowledgments	
		ohne	mit
Erzeugungs-zeitpunkt	Beginn	*Start.UA*	*Start.Ack*
	Beitritt	*Join.UA*	*Join.Ack*

Tabelle 7.3: Klassifikation der Evaluationsvarianten

7.3 Evaluationsergebnisse

In diesem Kapitel werden die Ergebnisse der Evaluation vorgestellt. Ausgehend von den in Abschnitt 7.2 vorgestellten Varianten wurden die Nachrichtenanzahl und die Nachrichtengröße gemessen. Diese Werte sind in Tabellen dargestellt, die nach dem in Tabelle 7.1 erstellten Schema untergliedert sind. Sie zeigen den Nachrichtennamen, die Häufigkeit des Auftretens dieser Nachricht und die kumulierte Nachrichtengröße der einzelnen Nachrichten. Des Weiteren wird die Summe aller verschickten Nachrichten und die Gesamtgröße angegeben. In den einzelnen Tabellen sind Änderungen zu anderen bzw. der vorhergehenden Variante „fett“ hervorgehoben.

7.3.1 Variante *Start.UA*

Die Tabelle 7.4 zeigt die in Variante *Start.UA* versandten Nachrichten, die als Folge von Änderungen am α-Doc erzeugt wurden.

Es kam im Verlauf des Anwendungsfalls zur Erstellung von drei neuen α-Cards, wobei es sich bei zwei α-Cards um ein α-Card-Paar gehandelt hat. Daraus ergeben sich drei `AddAlphaCardEvent`-Nachrichten.

Die Nachrichten vom Typ `ChangePayloadEvent` ergeben sich zum einen aus Änderungen an Content- sowie an Coordination-α-Cards. Durch die Erstellung der α-Cards kam es zu vier Änderungen am PSA. Außerdem wurde dreimal Payload zu einer α-Card hinzugefügt. Daraus ergeben sich vier `ChangePayloadEvent`-Nachrichten für die PSA-Änderungen und weitere vier `ChangePayloadEvent`-Nachrichten für das Anfügen von Payload, da diese Nachrichten auch an den handelnden Akteur selbst verschickt werden.[1]

Die Summe der Nachrichtengröße der `ChangePayloadEvents` relativiert sich etwas, wenn man ihr eine gesamte Payloadgröße für die Nutzdaten von ca. 89 KB gegenüberstellt. Eine `ChangePayloadEvent`-Nachricht besitzt also in etwa die doppelte Größe wie das angehängte Payload.

Nachdem Rad^A das Beitrittsprotokoll eingeleitet hat (`SequentialJoinCallback`), wird von Gyn^A eine Nachricht vom Typ `SequentialJoinSynchronisation` verschickt, die alle Änderungen am α-Doc enthält, seitdem der Klon für Rad^A erstellt wurde. Dadurch, dass der Klon des α-Doc zu Beginn des Prozesses erstellt wurde, ergibt sich die angegebene Nachrichtengröße von 241 KB, da insbesondere alle hinzugefügten Payloads mitgeschickt werden müssen.

Nachrichtentyp	Nachricht	Anzahl	∑Nachrichten-größe/KB
Änderung an Content- oder Coordination-α-Card	`AddAlphaCardEvent`	**3**	**25,8**
	`ChangeAlphaCardDescriptorEvent`	**16**	**144,4**
	`ChangePayloadEvent`	**8**	**271,4**
Beitrittsprotokoll	`SequentialJoinCallback`	**1**	**6,5**
	`SequentialJoinSynchronisation`	**1**	**241,0**
		29	**689,1**

Tabelle 7.4: Evaluation Variante *Start.UA*

1 Dies ergibt sich daraus, dass ein Akteur an mehreren Arbeitsplätzen parallel arbeiten könnte. Damit alle seine Kopien des α-Doc aktuell gehalten werden, müssen Änderungen auch an ihn selbst propagiert werden.

7.3.2 Variante *Join.UA*

Wenn der Klon des α-Doc zum Beitritt von Rad^A erst spät erstellt wird, dann werden prinzipiell die gleichen Nachrichten verschickt wie unter Variante *Start.UA*. Tabelle 7.5 zeigt jedoch, dass die `SequentialJoinSynchronisation`-Nachricht während des Ablaufs des Beitrittsprotokolls lediglich 5,4 KB groß ist. Die deutlich kleinere Nachrichtengröße wird erreicht, weil keine zusätzlichen Änderungen während des Beitritts synchronisiert werden müssen. Diese sind bereits Bestandteil des α-Doc-Klons.

Nachrichtentyp	Nachricht	Anzahl	∑Nachrichten-größe/KB
Änderung an Content- oder Coordination-α-Card	`AddAlphaCardEvent`	3	25,8
	`ChangeAlphaCardDescriptorEvent`	16	144,4
	`ChangePayloadEvent`	8	271,4
Beitrittsprotokoll	`SequentialJoinCallback`	1	**7,7**
	`SequentialJoinSynchronisation`	1	**5,4**
		29	**454,7**

Tabelle 7.5: Evaluation Variante *Join.UA*

7.3.3 Variante *Join.Ack*

Die Tabelle 7.6 fügt den bereits erläuterten Nachrichtentypen die Empfangsbestätigungen hinzu. Diese werden während der Änderungen an α-Cards von Rad^A verschickt. Während dieser Zeit kommt es insgesamt zu acht `DeliveryAcknowledgements`. Damit bestätigt auf der einen Seite Gyn^A den Erhalt der Änderungen, auf der anderen Seite werden diese Änderungen auch von Rad^A bestätigt.

Nachrichtentyp	Nachricht	Anzahl	∑Nachrichten-größe/KB
Änderung an Content- oder Coordination-α-Card	`AddAlphaCardEvent`	3	25,8
	`ChangeAlphaCardDescriptorEvent`	16	144,4
	`ChangePayloadEvent`	8	271,4
Beitrittsprotokoll	`SequentialJoinCallback`	1	7,7
	`SequentialJoinSynchronisation`	1	5,4
Empfangsbestätigung	`DeliveryAcknowledgement`	**8**	**46,8**
		37	**501,5**

Tabelle 7.6: Evaluation Variante *Join.Ack*

7.3.4 Variante *Start.Ack*

In der Variante *Start.Ack* werden weitere Empfangsbestätigungen verschickt, denn Rad^A muss die während des Beitrittsprotokolls ausgetauschten Änderungsversionen ebenfalls bestätigen (cf. Tabelle 7.7). Im Vergleich zu Variante *Join.Ack* werden acht zusätzliche `DeliveryAcknowledgements` verschickt.

In dieser Variante wird während des Beitrittsprotokolls wiederum eine Nachricht vom Typ `SequentialJoinCallback` mit einer Größe von 241 KB verschickt, da der α-Doc-Klon bereits zu Beginn des Prozesses erstellt wurde.

Nachrichtentyp	Nachricht	Anzahl	∑Nachrichten-größe/KB
Änderung an Content- oder Coordination-α-Card	`AddAlphaCardEvent`	3	25,8
	`ChangeAlphaCardDescriptorEvent`	16	144,4
	`ChangePayloadEvent`	8	271,4
Beitrittsprotokoll	`SequentialJoinCallback`	1	7,7
	`SequentialJoinSynchronisation`	1	**241,0**
Empfangsbestätigung	`DeliveryAcknowledgement`	**16**	**92,9**
		45	**783,2**

Tabelle 7.7: Evaluation Variante *Start.Ack*

7.4 Zusammenfassung

Die im Rahmen der Evaluation auftretenden Nachrichten und die Unterschiede in den verschiedenen Varianten ergeben sich insbesondere aufgrund von verschiedenen Nachrichtengrößen und der Anzahl an verschickten Nachrichten.[1] Dabei können auch Nachrichten des gleichen Typs verschiedene Größen aufweisen, denn der mitgeschickte Payload und Deskriptor haben Einfluss auf diese. Eine Übersicht der durchschnittlichen Nachrichtengrößen gibt Tabelle 7.8. Insbesondere unter Berücksichtigung der in Tabelle 7.2 definierten Nutzdaten und einer durchschnittlichen Payloadgröße der α-Content-Cards von 34 KB relativieren sich die durchschnittlichen Nachrichtengrößen im Vergleich zum reinen Versand des Payloads im Anhang einer E-Mail erheblich.

1 Für detailliertere Informationen, die die einzelnen ausgetauschten Nachrichten betreffen, wird auf das Anhangskapitel E verwiesen.

Der unterschiedliche Synchronisierungsaufwand beim Beitritt von Rad^A ausgelöst durch den verschiedenen Zeitpunkt für die α-Doc-Klon-Erstellung zeigt sich nicht in der Anzahl der verschickten Nachrichten. Der Unterschied liegt vielmehr in der Größe der Nachrichten, die während des Beitrittsprotokolls verschickt werden. Bei einer frühen Klon-Erzeugung ist die `SequentialJoinSynchronisation`-Nachricht erheblich größer als bei einer späten Klon-Erzeugung.

Diesen Zusammenhang stellt auch Tabelle 7.9 dar. Bei den beiden aufgelisteten Varianten ist der Anteil der versendeten Nachrichten am Gesamtnachrichtenaufkommen identisch. In der Variante *Start.UA* hat die Nachricht `SequentialJoinSynchronisation` aber einen deutlich höheren Anteil an der Gesamtnachrichtengröße aller Nachrichten.

Nachricht	Gesamtanzahl	ØNachrichten-größe/KB
`AddAlphaCardEvent`	12	8,6
`ChangeAlphaCardDescriptorEvent`	64	9,0
`ChangePayloadEvent`	32	33,9
`SequentialJoinCallback`	4	7,4
`SequentialJoinSynchronisation`	4	123,2
`DeliveryAcknowledgement`	24	5,8
	140	**188,0**

Tabelle 7.8: Durchschnittliche Nachrichtengröße aller Nachrichten

Die in den Varianten *Start.Ack* und *Join.Ack* versendeten `DeliveryAcknowledgements` weisen eine relativ kleine Größe auf. In der Übersicht von Tabelle 7.8 bilden sie die dritthäufigste Nachrichtenart mit der kleinsten durchschnittlichen Größe. Es ist zu bemerken, dass Acknowledgements nur verschickt werden, wenn es mehrere Akteure im Prozess gibt. Solange lediglich ein Akteur beteiligt ist, wird auf den Versand von Acknowledgements verzichtet.

Der Unterschied von den Varianten *Start.Ack* und *Join.Ack* liegt darin, dass in Variante *Start.Ack* auch die Änderungsversionen der Artefakte bestätigt werden müssen, die während des Beitrittprotokolls empfangen wurden. Diesen Zusammenhang zeigt auch Tabelle 7.10. Die Variante *Start.Ack* weist die doppelte Anzahl an versendeten `DeliveryAcknowledgements` auf wie die Variante *Join.Ack*. Die Acknowledgement-Nachricht selbst unterscheidet sich in den Varianten aber nicht.

Werden die Tabellen 7.9 und 7.10 miteinander verglichen, dann wird deutlich, dass durch das Benutzen von Acknowledgements die Summe der insgesamt verschickten Nachrichten um ca. 10% ansteigt. Der Vergleich der jeweiligen *Start-* mit der *Join-*

Variante ergibt einen Unterschied des ingesamt versandten Datenvolumens um ca. den Faktor 1,5.

Die Ergebnisse der Evaluation haben gezeigt, dass es ausgehend von realen Dateigrößen für medizinische Dokumente durch den Einsatz von α-Flow lediglich zu einer unwesentlichen Vergrößerung der Datenmenge kommt. Der durch α-Flow gewonnene Mehrwert durch eine verbesserte Koordination der beteiligten Akteure und eine automatische Synchronisation der an der Fallakte getätigten Änderungen wiegen diesen Mehraufwand durchaus auf. Der Einsatz von Acknowledgements wiederum führt zu einem nochmaligem Mehraufwand, der sich aber hauptsächlich in der Anzahl der verschickten Nachrichten und weniger in deren Größe wiederspiegelt.

	Variante *Start.UA*				Variante *Join.UA*			
Nachricht	Anzahl	Anzahl in %	∑Nachrichten-größe/KB	∑Nachrichten-größe in %	Anzahl	Anzahl in %	∑Nachrichten-größe/KB	∑Nachrichten-größe in %
`AddAlphaCardEvent`	3	10%	25,8	4%	3	10%	25,8	6%
`ChangeAlphaCardDescriptorEvent`	16	55%	144,4	21%	16	55%	144,4	32%
`ChangePayloadEvent`	8	28%	271,4	39%	8	28%	271,4	60%
`SequentialJoinCallback`	1	3%	6,5	1%	1	3%	7,7	2%
`SequentialJoinSynchronisation`	1	3%	241,0	35%	1	3%	5,4	1%
	29		**689,1**		**29**		**454,7**	

Tabelle 7.9: Vergleich der Evaluationsvarianten *Start.UA* und *Join.UA*

	Variante *Start.Ack*				Variante *Join.Ack*			
Nachricht	Anzahl	Anzahl/%	∑Nachrichten-größe/KB	∑Nachrichten-größe/%	Anzahl	Anzahl/%	∑Nachrichten-größe/KB	∑Nachrichten-größe/%
`AddAlphaCardEvent`	3	6,7	25,8	3,3	3	8,1	25,8	5,1
`ChangeAlphaCardDescriptorEvent`	16	35,6	144,4	18,4	16	43,2	144,4	28,8
`ChangePayloadEvent`	8	17,8	271,4	34,7	8	21,6	271,4	54,1
`SequentialJoinCallback`	1	2,2	7,7	1,0	1	2,7	7,7	1,5
`SequentialJoinSynchronisation`	1	2,2	241,0	30,8	1	2,7	5,4	1,1
`DeliveryAcknowledgement`	16	**35,6**	92,9	**11,9**	8	**21,6**	46,8	**9,3**
	45		**783,2**		**37**		**501,5**	

Tabelle 7.10: Vergleich der Evaluationsvarianten *Start.Ack* und *Join.Ack*

8 Ausblick

Im Laufe der Arbeit haben sich verschiedene Probleme und Erweiterungsmöglichkeiten aufgetan, die allerdings nicht Bestandteil dieser Arbeit waren. Für weitere im Rahmen von α-Flow durchgeführte Arbeiten soll an dieser Stelle auf sie eingegangen werden, sodass sie als Basis für weitere Überlegungen dienen können.

8.1 Rollen- und Berechtigungskonzept

Die eingeführten Token werden momentan als Prozessrollenbezeichnung verwendet. Sie haben neben der textuellen Beschreibung der Rolle eines Prozessteilnehmers bisher keine weitere Aussagekraft. Es wäre denkbar, sie als Grundlage für ein darauf aufbauendes Rollenkonzept mit Rechtevergabe heranzuziehen (cf. [KME+09] und [Kla11]). Durch Role-based Access Control (RBAC) könnten bestimmte Rollen mit besonderen Rechten ausgestattet werden. Damit wäre eine feingranulare Rechtevergabe für den einzelnen Teilnehmer möglich.

Besitzer eines Tokens könnten beispielsweise Sonderrechte haben, die sie von den übrigen Teilnehmern unterscheiden. So könnte der Wortführer besondere Änderungsrechte am α-Doc besitzen.

8.2 Token Prototype Artifact

Momentan sind die Token fest in das System integriert. Es gibt das Initiator, das Wortführer- und das Patientenkontakt-Token. Während des Entwurfs und der Implementierung wurde bereits darauf geachtet, das Modell generisch aufzubauen. Der Hintergedanke war es, dass Token in der Zukunft während der Laufzeit adaptiv hinzugefügt und geändert werden können – vorzugsweise durch den Wortführer des Prozesses.

Die Verwaltung der im Prozess verwendeten Token-Menge müsste dazu in eine eigene Datenstruktur ausgelagert werden, die zwischen den einzelnen Prozessteilnehmern

synchronisiert wird. Eine Einführung eines Token Prototype Artifact (TPA) analog zum APA wäre daher naheliegend.

8.3 Synchronisierungsverhalten

Momentan werden bei Änderungen an Prozessartefakten alle Teilnehmer über diese informiert und ihre Kopie des α-Doc auf den aktuellsten Stand gebracht, sobald sie online ist. In der Zukunft wäre es vielleicht denkbar, dass einzelne Teilnehmer von diesem Aktualisierungsprozess ausgeschlossen werden oder dass sie freiwillig darauf verzichten, also nicht mehr mit den aktuellsten Änderungen versorgt werden. Ein Teilnehmer könnte auch den Wunsch hegen, nur noch in bestimmten Intervallen informiert zu werden oder ganz aus dem Behandlungsprozess auszuscheiden, weil seine Arbeit bereits getan ist. Diese Szenarien sind insbesondere denkbar bei Ärzten wie Pathologen, die keinen Patientenkontakt haben und nach der Untersuchung einer Gewebeprobe keinen Anteil mehr an der weiteren Behandlung und ihrem Ergebnis haben. Für eine solche Unterscheidung könnte das Patientenkontakt-Token die Basis bilden.

8.4 Acknowledgement Repository Artifact

Im Zuge der Überlegungen zur Speicherstruktur für die eingeführten Acknowledgements (cf. 6.2.2) wurde bisher nicht die Möglichkeit betrachtet, diese in eine eigene neue Coordination-α-Card zu speichern. Das Einführen einer weiteren α-Card für Acknowledgements bildet eine weitere Variante, die im gesamten Prozess versandten Acknowledgements zu speichern. Eine solche Struktur könnte durch ein Acknowledgement Repository Artifact (ARA) in Form einer α-Card geschaffen werden.

Das ARA liefert eine zentrale Sicht auf die von allen Teilnehmern im Prozess versandten Empfangsbestätigungen. Es speichert je Artefaktversion die von den einzelnen Prozessteilnehmern versandten Empfangsbestätigungen. Bei Änderungen an dieser α-Card kann auf die bereits bestehende Infrastruktur von α-Flow wie beispielsweise die Versionierung und das Propagieren von Änderungen zurückgegriffen werden.

Allerdings käme es bei diesem Prozessartefakt zu besonders häufigen Änderungen und damit zu einem großen Synchronisierungsaufwand. Bei der Erzeugung einer neuen Version eines Prozessartefakts wird von jedem Prozessteilnehmer eine Empfangsbestätigung verschickt und diese dann einzeln im ARA gespeichert. Das bedeutet, dass je neuer Version eines Prozessartefakts genausoviele Versionen des ARA erzeugt werden wie

Prozessteilnehmer existieren. Diese Versionen müssen ebenfalls synchronisiert werden. Gegebenenfalls müsste daher ein neues Synchronisierungsschema entworfen werden, das das ARA weniger häufig synchronisiert, oder die Konfiguration sollte der Art sein, dass die Synchronisierung nur bei Bedarf angestoßen wird.

9 Zusammenfassung

Verteilte und heterogene Systeme bestimmen das Umfeld vieler Bereiche des Gesundheitswesens. Das α-Flow-Projekt bietet einen Ansatz, patientenbezogene Daten und Behandlungsdokumente auszutauschen. Durch seine dokumentenorientierte Vorgehensweise wird insbesondere die Tatsache berücksichtigt, dass die Anzahl der an der Behandlung eines Patienten mitwirkenden Teilnehmer zu Beginn unbekannt ist und auch der genaue Behandlungsablauf noch offen ist.

Die Koordination der Patientenbehandlung und der Austausch von Informationen wird durch die Anreicherung der medizinischen Dokumente um aktive Eigenschaften erreicht, die in einer elektronischen Fallakte gesammelt werden. Diese Fallakte stellt das aktive Dokument dar und ist das Mittel zum Informationsaustausch, über das neu gewonnene Erkenntnisse im Behandlungsverlauf auf elektronischem Weg an alle Prozessteilnehmer verbreitet werden. Durch dieses Vorgehen kann jeder Teilnehmer bei seiner Entscheidungsfindung immer auf die aktuellsten Informationen zurückgreifen und somit Fehlentscheidungen aufgrund einer unzureichenden Informationslage vorbeugen.

Im Rahmen dieser Arbeit wurde ein Modell entwickelt, das ermöglicht, am Prozess teilnehmende Akteure mit Prozessrollen auszuzeichnen. Die Prozessrollen können im Fall des Wortführer-Tokens weitergegeben werden, im Fall des Prozessinitiator- und des Patientenkontakt-Tokens wird der Wert einmalig festgelegt bzw. kann selbstständig geändert werden.

Das Modell wurde so gestaltet, dass es in seiner Grundlage die Möglichkeit bietet, zur Laufzeit angepasst zu werden. Dazu wurde auf die Konzepte des EAV-Ansatzes, die bereits in α-Adaptive verwendet wurden, zurückgegriffen. Eine Weitergabe der Prozessrollen wird durch ein geeignetes Nachrichtenprotokoll erreicht, das insbesondere Konflikte, die bei nebenläufigen Token-Änderungen auftreten, zu beheben weiß. Um eine kausale Ordnung auf den Token-Änderungen herstellen zu können, wurden Zeitstempel eingeführt.

Neben diesem Schwerpunkt wurden in dieser Arbeit Empfangsbestätigungen eingeführt, die Rückmeldung über Änderungsnachrichten geben. Ein Akteur, der eine Änderung an einer α-Card – beispielsweise hat er einen neuen Befund als Payload angehängt – durchführt, erfährt durch die Empfangsbestätigungen der anderen Akteure, wann diese

seine Änderung erhalten haben. Damit ist er in der Lage, bei Ausbleiben einer Bestätigung gegebenenfalls auf einem anderen Kommunikationsweg beim entsprechenden Akteur nachzufragen.

Als Speicherstruktur wurde eine lokale, nicht synchronisierte Variante gewählt. Der Verlust dieser Struktur würde den Prozessfortgang nicht behindern. Das minimale Nachrichtenformat der `Delivery Acknowledgements` umfasst neben den Zeitstempeln der Verarbeitung der ursprünglichen Nachricht Informationen über den sendenden Akteur und die Version des geänderten Prozessartefakts für eine eindeutige Zuordnung. Eine übersichtliche grafische Anzeigekomponente stellt die empfangenen Acknowledgements im Editor an zentraler Stelle je α-Card dar.

Die vorliegende Arbeit unterscheidet die bisher gleichberechtigten Prozessteilnehmer durch Prozessrollen voneinander. Mit diesen Rollen erhalten die einzelnen Akteure bestimmte Sonderrollen und Eigenschaften. Im Zuge des Voranschreitens des Behandlungsprozesses können sich die Rollen verändern. Damit lässt sich die sich ändernde Situation der Verantwortlichkeiten, wie sie im Rahmen der beispielhaft angenommenen Brustkrebsbehandlung stattfindet, im System abbilden.

Durch das Versenden von Empfangsbestätigungen erhält jeder Prozessteilnehmer eine Versicherung, dass seine Änderung bei seinem Gegenüber angekommen ist. Diese Funktion erweitert das bisherige Kommunikationsschema um zusätzliche Informationen, die den einzelnen Akteur über zeitliche Zusammenhänge unterrichtet.

Die vorliegende Arbeit ermöglicht somit zum einen die Auszeichnung einzelner Prozessteilnehmer durch Prozessrollen und zum anderen eine erhöhte Zuverlässigkeit durch Rückmeldungen beim Informationsversand. Damit wurde das Konzept von α-Flow sinnvoll und unterstützend erweitert, sodass es in Zukunft den Teilnehmern verteilter Systeme mit ihren heterogenen und komplexen Strukturen leichter fällt ihre Zusammenarbeit zu koordinieren.

Appendices

A Kernelemente der BPMN

Mit BPMN erstellte Prozessdiagramme bestehen hauptsächlich aus den in Abbildung A.1 dargestellten Symbolen. Ihre Anordnung legt fest, was in welcher Reihenfolge unter bestimmten Bedingungen ausgeführt wird.

Die Flussobjekte stellen die grafischen Hauptelemente dar, um das Verhalten eines Geschäftsprozesses zu definieren. Die drei Flussobjekte sind Aktivitäten, Ereignisse und Gateways. Aufgaben, die während des Ablaufs eines Prozesses ausgeführt werden müssen, werden in Aktivitäten abgebildet. Hierbei kann eine Aktivität auch einen Subprozess enthalten.

Teilweise ist die Ausführung von Aktivitäten an bestimmte Bedingungen geknüpft. Den Prozessfluss kann man mit Gateways steuern. Mit Gateways ist es möglich, auf Ereignisse zu reagieren und auch Abläufe zu parallelisieren und diese wieder zusammenzuführen.

Im Prozessverlauf können Ereignisse eintreten, die eine Wirkung auf den weiteren Prozessverlauf haben. Es wird unterschieden zwischen untypisierten und typsierten Ereignissen. Typisierte Ereignisse haben einen definierten Auslöser oder eine bestimmte Ursache bzw. eine Folgereaktion (Nachricht, Zeit, Fehler). Bei untypisierten Ereignissen fehlt diese Information. Weiterhin wird zwischen Start-, Zwischen- und Endereignissen getrennt, je nachdem wann diese im Prozess auftreten.

Die in einem Prozess verarbeiteten Informationen oder die zwischen Prozessen ausgetauschten Daten werden durch ein Datenobjekt dargestellt. Es beinhaltet die Daten, die von einer Aktivität zur Bearbeitung einer Aufgabe benötigt werden. Dabei kann es sich um sowohl Eingabe- als auch Ausgabedaten handeln.

Es gibt zwei Möglichkeiten, die primären Modellierungelemente zu gruppieren. Zu den gruppierenden Elementen zählen die Pools sowie die Lanes. Abschnitt 4.1.3 beschreibt diese Kernelemente genauer.

Weitergehende Informationen über den Prozess können durch Artefakte dem Diagramm hinzugefügt werden. Die Standard-Artefakte sind die Gruppierung und die Textanmerkung.

Die Flussobjekte werden durch sog. verbindende Objekte miteinander verbunden. Des Weiteren können auch ergänzende Informationen in Form von Artefakten durch

verbindende Objekte an Flussobjekte gekoppelt werden. Zu den verbindenden Objekten gehören Sequenzflüsse, Nachrichtenflüsse und Assoziationen. Sequenzflüsse verbinden Flussobjekte untereinander und zeigen damit die Reihenfolge an. Muss eine Verbindung über Pool-Grenzen hinweg erfolgen, benötigt man Nachrichtenflüsse. Die zwei Teilnehmer, dargestellt durch zwei Pools, müssen auf eingehende und ausgehende Nachrichten vorbereitet sein. Assoziationen binden Artefakte an Flussobjekte.

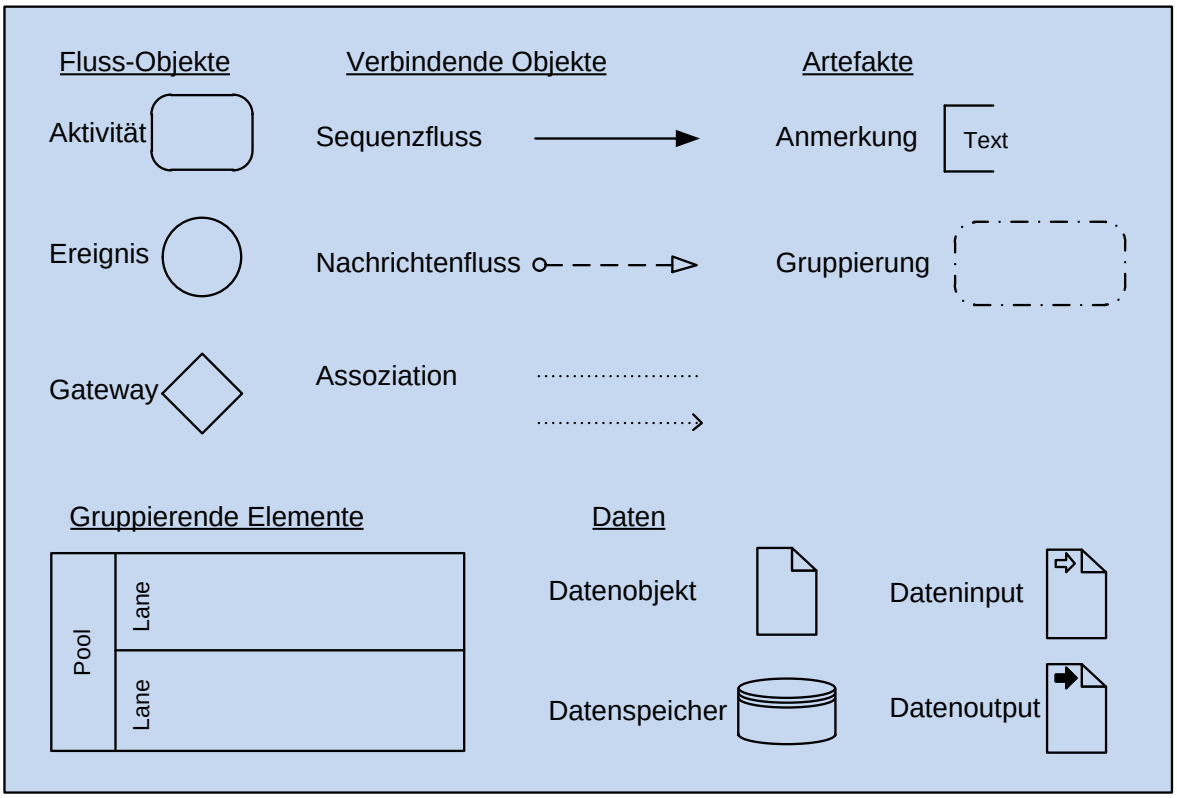


Bild A.1: Kernelemente der Business Process Model and Notation

B Gegenseitiger Ausschluss durch zentralisierten Koordinator-Ansatz

Diese Vorgehensweise zum Betreten und Verlassen eines kritischen Abschnitts durch einen zentralisierten Koordinator-Ansatz (cf. 4.2.2.1) ist in den folgenden beiden Listings zusammengefasst. In Listing B.1 ist der Pseudocode für den Koordinator-Prozess und in Listing B.2 der Pseudocode für alle weiteren Prozesse dargestellt.

```
P_0:
    List<request> reqlist;
    boolean haveToken = true;

    Upon receive(request):
    append(reqlist,request);
    if(haveToken) then checkReq();

    Upon receive(token):
    checkReq();

    checkReq():
    P = first(reqlist);
    send(token) to P;
    haveToken = false;
```

Listing B.1: Pseudocode des zentralisierten Koordinator-Ansatzes: P_0

```
P_i:
    boolean inCriticalSection = false;

    To request:
    send(request) to P_0;

    Upon receive(token) from P_0:
    inCriticalSection = true;
```

```
To release:
send(token) to P_0;
inCriticalSection = false;
```

Listing B.2: Pseudocode des zentralisierten Koordinator-Ansatzes: P_i

C E-Mail im Internet

Insbesondere aus der Benutzersicht ist die E-Mail neben dem World Wide Web die meistverbreiteste Internet-Anwendung (cf. [Hal05, Seite 515], [KR02, Seite 121]). E-Mails sind das digitale Pendant zur Briefpost und sind ebenso wie diese asynchron. Neben reinen Textnachrichten können heutzutage auch Hyperlinks, HTML-Nachrichten, Bilder und Videos verschickt werden.

Im Folgenden soll erst ein Überblick über die E-Mail-Infrastruktur gegeben werden, da sie im Rahmen von α-Offsync das Kommunikationsmedium zum Informationsaustausch darstellt. Dann werden die Nachrichtenformate näher betrachtet, die es ermöglichen, die verschiedenen Inhalte zu verschicken. Für den Versand von E-Mails hat sich SMTP als Protokoll durchgesetzt. Als Grundlage für diesen Abschnitt dienen [Hal05, Seite 515ff.], [KR02, Seite 121ff.].

C.0.1 Überblick über die E-Mail-Infrastruktur

Abbildung C.1 gibt einen Überblick über die Infrastruktur, die sich hinter der Internet-Anwendung E-Mail verbirgt. Zu den wichtigsten Komponenten zählen: der *User Agent*, der *E-Mail Server* und das *Simple Mail Transfer Protocol* (SMTP).

Der User Agent ist der E-Mail-Client. Er stellt die grafische Benutzeroberfläche zum E-Mail-System zur Verfügung, um E-Mails zu schreiben, diese zu senden und E-Mail-Nachrichten zu empfangen. Er ermöglicht auch, Nachrichten zu beantworten, sie weiterzuleiten und sie zu löschen.

Ein E-Mail-Server besitzt eine Warteschlange für abgehende Nachrichten, in die er Nachrichten vom Benutzer zwischenspeichert, bis er sie versendet. Auf der anderen Seite existiert für jeden registrierten Benutzer auf dem E-Mail-Server eine Mailbox. Nachrichten für einen bestimmten Benutzer werden dort abgelegt, bis dieser sie abholt.

Die Kommunikation zwischen den einzelnen E-Mail-Servern basiert auf SMTP. Ebenso schickt ein User Agent seine E-Mails per SMTP an seinen E-Mail-Server. Die Kommunikation zwischen E-Mail-Server und User Agent, also das Abrufen von E-Mails aus der Mailbox, kann beispielsweise mittels Post Office Protocol Version 3 (POP3) erfolgen.

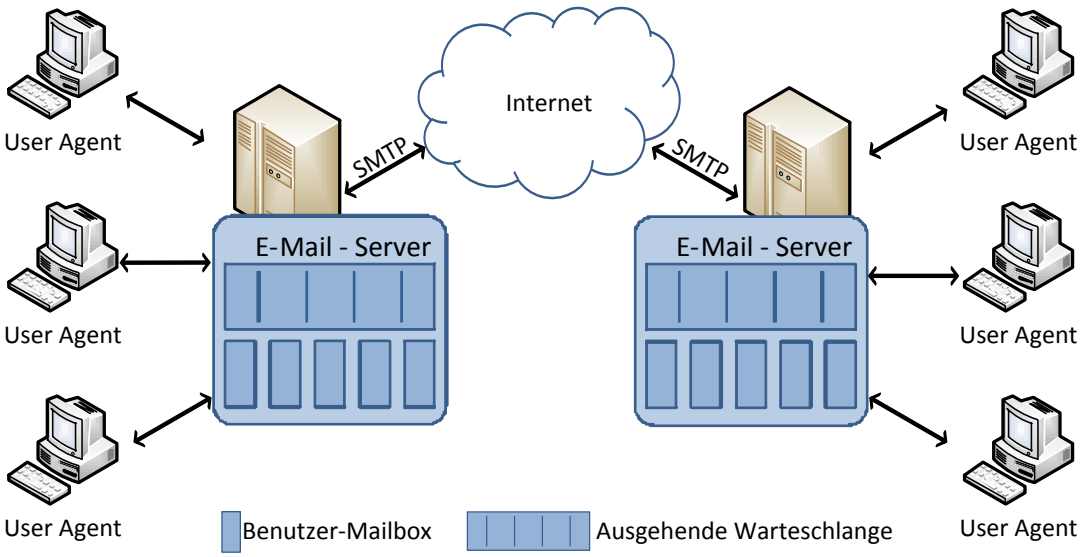


Bild C.1: E-Mail-Infrastruktur

Der Ablauf einer E-Mail-Konversation zwischen einem Sender (Alice) und einem Empfänger (Bob) soll nun einmal beispielhaft beschrieben werden ([KR02, Seite 121f.]): Alice schreibt mit ihrem User Agent eine Nachricht. Beim Senden wird die Nachricht an ihren E-Mail-Server übertragen und dort in der Nachrichtenwarteschlange abgelegt. Dort verweilt sie, bis die Nachricht fehlerfrei an Bobs E-Mail-Server übertragen werden konnte. Der Übertragungsweg muss dabei nicht zwingend direkt erfolgen, sondern es können auch weitere E-Mail-Server dazwischen geschaltet sein.

Im Fehlerfall wird Alices E-Mail-Server in einem bestimmten Intervall versuchen, die Nachricht erneut zu übertragen. Sollte dies in einem bestimmten Zeitraum nicht klappen, dann entfernt der E-Mail-Server die Nachricht aus der Warteschlange und informiert Alice über des Fehlschlagen des Sendens. Auf der Empfängerseite erhält der E-Mail-Server von Bob die eingehende Nachricht und kann sie in Bobs Mailbox ablegen. Nachdem sich Bob gegenüber dem E-Mail-Server authentifiziert hat, kann er die Nachricht aus seiner Mailbox abrufen.

C.0.2 Nachrichtenformate und MIME

Die Syntax von Textnachrichten, die zwischen zwei Benutzern im Rahmen von E-Mail verschickt werden, ist in [Cro82] und dessen Überarbeitung [Res01] definiert. Dieser Standard definiert reine Textnachrichten. Zum Versand von multimedialem Inhalt wie

beispielsweise Audio und Video müssen Erweiterungen wie Multipurpose Internet Mail Extensions (MIME) benutzt werden.

C.0.2.1 Einfache Textnachrichten

Ursprünglich wurden E-Mails dazu entworfen, reine Textnachrichten, die auf dem ASCII-Zeichensatz basieren, zu enthalten (cf. [Cer69]).

Eine Nachricht besteht aus Header-Feldern (Header), die Metainformationen zur Nachricht enthalten, und einem optionalem Body. Die in [Cro82] definierten Header-Zeilen bestehen aus lesbarem American Standard Code for Information Interchange (ASCII)-Text. Sie setzen sich zusammen aus einem Schlüsselwort und einem durch einen Doppelpunkt abgetrennten Wert. Einige Header-Zeilen sind zwingend (z.B. FROM: und TO:), andere hingegen sind optional (z.B. SUBJECT:). Ein typischer Nachrichten-Header sieht wie folgt aus:

```
From: christian@hunsen.de
To: claus@hunsen.de
Date: Date: Tue, 15 May 2012 14:52:52
Subject: Masterarbeit
```

Der in ASCII kodierte Nachrichten-Body folgt dem Header getrennt durch eine einzelne leere Zeile (CRLF). Der Body enthält die vom Benutzer eingegebene Nachricht. Die Nachricht besteht aus einzelnen Zeilen, die durch Wagenrücklauf und anschließendem Zeilenvorschub voneinander getrennt werden. Dabei schreibt der Standard eine maximale Zeilenlänge von 998 Zeichen vor. Die E-Mail-Nachricht endet mit einer Zeile, die nur einen Punkt enthält.

C.0.2.2 Multipurpose Internet Mail Extensions (MIME)

Im Laufe der Zeit entstand der Wunsch, neben reinen Textnachrichten auch multimediale Inhalte verschicken zu können. Das Konzept der Multipurpose Internet Mail Extensions (MIME) lässt es zu, E-Mail-Nachrichten in einer beliebigen Kodierung zu verfassen. Neben dem ASCII-Zeichensatz sind nun auch Texte mit Sonderzeichen und in anderen Sprachen möglich. Die Erweiterung zu [Cro82] ist in [FB96b], [FB96c], [Moo96], [FKP96] und [FB96a] definiert und hat als Ziel mit dem gleichen Nachrichtentransfersystem alternative Medientypen übertragen zu können. Damit etwas anderes als ASCII-Text übertragen werden kann, sind zusätzliche Header-Felder nötig.

Für die Unterstützung von multimedialen Inhalten sind insbesondere folgende Header-Felder wichtig:

- `MIME-Version:` Gibt die Version an, mit der die Nachricht erstellt wurde.
- `Content-Type:` Dieser Eintrag ermöglicht es dem empfangenden User Agent, die Nachricht entsprechend des Formats des Inhalts zu behandeln. So kann er beispielsweise das JPEG-Bild an eine Dekompressionsroutine weiterleiten.
- `Content-Transfer-Encoding:` Auch wenn nun andere Inhalte als ASCII-Text versendet werden können, müssen diese vor dem Versenden trotzdem noch in ein ASCII-Format umkodiert werden. Auf der Empfangsseite kann der User Agent anhand dieses Feldes bestimmen, mit welchem Kodierverfahren Nicht-ASCII-Inhalt kodiert wurde.

Wenn ein User Agent eine Nachricht mit diesen Header-Feldern erhält, dann kann er anhand des `Content-Transfer-Encoding` den Inhalt des Body in sein ursprüngliches Nicht-ASCII-Format umwandeln und es dann dem `Content-Type` entsprechend weiterbehandeln.

Die vorgegebenen Content-Typen (cf. [FB96c]) werden in der Form `type/subtype` angeben. Zu den Haupttypen zählen: `text, image, audio, video, application, multipart` und `message`.

C.0.2.3 Multipart-Nachrichten

Der Typ `multipart` verdient besondere Erwähnung. Ähnlich einer Website kann mit diesen Typ auch eine E-Mail verschiedene Objekte enthalten. So kann eine Nachricht mit diesem Typ z.B. einen einleitenden Text und dann eine Anzahl an Bildern und ein Video enthalten.

Diese Anzahl an Objekten verschiedenen Typs werden alle in einer einzigen Nachricht verschickt. Damit der User Agent die einzelnen Objekte wieder herausfinden kann, muss als `Content-Type multipart/mixed` oder ein ähnlicher Subtyp angegeben werden. Er definiert, dass die Nachricht aus mehreren unabhängigen Teilen besteht.

Um die einzelnen Objekte zu identifizieren, ist es nötig, zu definieren, wo jedes Objekt beginnt und endet, in welchem Format jedes einzelne Objekt kodiert ist und welchen Inhaltstyp jedes Objekt hat. Durch Einfügen von Begrenzungszeichen und den jeweiligen MIME-Header vor jedem Objekt wird dies ermöglicht. Der MIME-Header wiederholt sich also entsprechend der Anzahl an Objekten in der Nachricht.

C.0.3 SMTP

Das Simple Mail Transfer Protocol wurde ursprünglich in [Pos82] definiert und ist heute in der Version in [Kle08] aktuell. Es ist ein Standard für das Versenden von E-Mail-Nachrichten im Internet. Dabei ist SMTP verbindungsorientiert und arbeitet mit textuellen Befehlen.

Das Protokoll basiert auf folgendem Kommunikationsprotokoll (cf. Abbildung C.2): Wenn ein Benutzer das Senden einer Nachricht anstößt, baut der SMTP-Sender eine bidirektionale Verbindung zum SMTP-Empfänger auf. Der empfangende SMTP-Empfänger kann dabei das Ziel oder eine Zwischenstation sein. Wenn der SMTP-Empfänger der Server des Nachrichtenempfängers ist, dann wird er die Nachricht in das Benutzerpostfach ablegen. Falls er eine Zwischenstation ist, sendet er die Nachricht entsprechend an den nächsten SMTP-Empfänger. Damit entsteht eine Kette von SMTP-Sendern und Empfängern.[1] Zwischen den beiden Parteien werden im Folgenden sowohl Steuerungsbefehle wie auch die Nachricht selbst ausgetauscht.

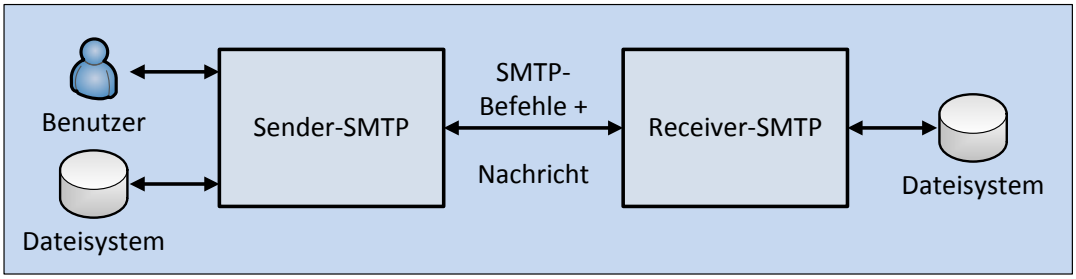


Bild C.2: Kommunikation zwischen SMTP-Client und SMTP-Server

Sobald die Verbindung zwischen Sender- und Empfänger-SMTP-Server aufgebaut ist, sendet der SMTP-Sender die entsprechenden SMTP-Kommandos und der SMTP-Empfänger antwortet entsprechend. Eingeleitet wird der Versand durch den `MAIL`-Befehl und dem Sender der E-Mail. Wenn der Empfänger Nachrichten annehmen kann, antwortet er mit `OK`. Dann sendet der SMTP-Sender it dem `RCPT`-Befehl den Empfänger der Nachricht. Wenn der SMTP-Empfänger Nachrichten für diesen Empfänger annehmen kann, dann sendet er wiederum `OK` zurück. Prinzipiell können nun mehrere Empfänger gesendet werden. Sobald die Empfänger ausgetauscht wurden, beginnt der SMTP-Sender

1 Die Sender-SMTP-Server werden dabei auch MTA und der Empfänger-SMTP-Server, der das Ziel ist, MDA genannt.

mit dem Senden der Daten, also dem eigentlichen Nachrichteninhalt. Dieser Nachrichteninhalt ist konform zum in Abschnitt C.0.2 dargestellten Standard formatiert und wird mit einer bestimmten Sequenz abgeschlossen. Wenn der SMTP-Empfänger die Daten erfolgreich erhalten hat, antwortet er zur Bestätigung noch einmal mit `OK`. Eine beispielhafte Kommunikation zeigt Listing C.1.

```
S: MAIL FROM:<christian@hunsen.de>
R: 250 OK

S: RCPT TO:<claus@hunsen.de>
R: 250 OK

S: RCPT TO:<test@hunsen.de>
R: 550 No such user here

S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Dies ist die Nachricht...
S: ...sie endet hier.
S: <CRLF>.<CRLF>
R: 250 OK
```

Listing C.1: Beispiel einer SMTP-Kommunikation

D Screenshoot des Token-Dialogs

Der folgende Screenshot zeigt den Token-Dialog im Admin-Modus (cf. D.1). Diese Ansicht dient in erster Linie administrativen Zwecken und kann zur manuellen Konfiguration des Prozessrollen verwendet werden. In seinem Aufbau greift er die Generik des zugrundeliegenden Datenschemas auf und kann ohne weitere Anpassung veränderte Mengen an Prozessrollen anzeigen und bearbeiten. Dabei verwendet er die gleichen Mechanismen wie auch bei der Anzeige der adaptiven Adornments genutzt werden.

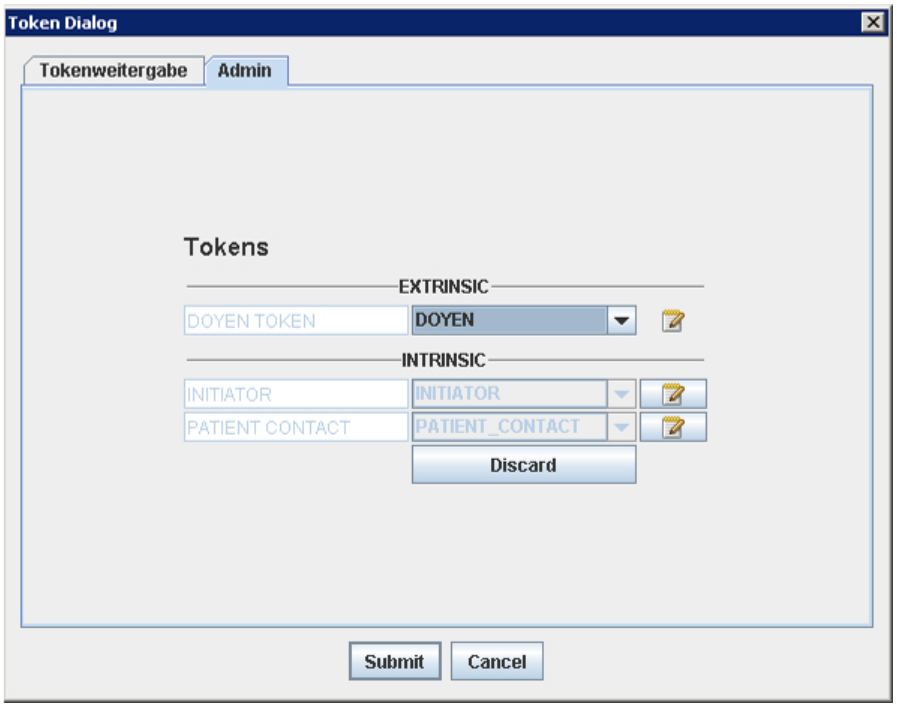


Bild D.1: Ansicht des Token-Dialogs in der Admin-Ansicht

E Detaillierte Evaluationsergebnisse

Die während der Evaluation in den einzelnen Varianten (cf. Kapitel 7) versandten Nachrichten sind in folgenden Tabellen aufgeführt. Sie zeigen das jeweilige Ereignis bzw. die durchgeführte Änderung am α-Doc und die daraufhin auftretenden Nachrichten. Des Weiteren werden die Nachrichtengröße und der Empfänger der Nachricht aufgelistet.

Die Tabellen bauen zum Teil auf anderen auf. Wenn Nachrichten identisch sind, werden diese nicht wiederholt, sondern mit einem Hinweis auf diese verwiesen. Änderungen in einer Tabelle im Vergleich zur vorhergehenden sind „fett“ hervorgehoben.

Ereignis	Nachricht	Nachrichtengröße/KB	Empfänger
Dokumentenerzeugung			
α-Card auf gültig & valide setzen	changeAlphaCardDescriptorEvent	8,1	Gyn^A
	changeAlphaCardDescriptorEvent	8,1	Gyn^A
Sonographie			
α-Card-Erstellung	addAlphaCardEvent	7,8	Gyn^A
	changePayloadEvent($PSA)	6,6	Gyn^A
Sonographiebericht als Payload hinzufügen	changeAlphaCardDescriptorEvent	8,1	Gyn^A
	changePayloadEvent	30,3	Gyn^A
α-Card auf gültig & valide setzen	changeAlphaCardDescriptorEvent	8,1	Gyn^A
	changeAlphaCardDescriptorEvent	8,1	Gyn^A
Überweisung an Rad^A			
Erstellung des α-Card-Paares	addAlphaCardEvent	10,1	Gyn^A
	changePayloadEvent($PSA)	6,6	Gyn^A
	addAlphaCardEvent	7,9	Gyn^A
	changePayloadEvent($PSA)	6,6	Gyn^A
	changePayloadEvent($PSA)	6,9	Gyn^A
Überweisung als Payload hinzufügen	changeAlphaCardDescriptorEvent	10,4	Gyn^A
	changePayloadEvent	41,4	Gyn^A
α-Card auf gültig & valide setzen	changeAlphaCardDescriptorEvent	10,4	Gyn^A
	changeAlphaCardDescriptorEvent	10,4	Gyn^A
Überweisung an Rad^A übertragen	changeAlphaCardDescriptorEvent	10,4	Gyn^A
Mammographieberichtplatzhalter an Rad^A schicken	changeAlphaCardDescriptorEvent	8,0	Gyn^A
Beitritt Rad^A			
Join-Protokoll einleiten	sequentialJoinCallback	6,5	Gyn^A
Sequentielle Rückmeldung	sequentialJoinSynchronisation	241	Rad^A
Mammographie			
Mammographiebericht als Payload hinzufügen	changeAlphaCardDescriptorEvent	8,2	Rad^A
	changePayloadEvent	85,8	Rad^A
	changeAlphaCardDescriptorEvent	9,9	Gyn^A
	changePayloadEvent	87,6	Gyn^A
α-Card auf gültig & valide setzen	changeAlphaCardDescriptorEvent	8,2	Rad^A
	changeAlphaCardDescriptorEvent	8,2	Rad^A
	changeAlphaCardDescriptorEvent	9,9	Gyn^A
	changeAlphaCardDescriptorEvent	9,9	Gyn^A

Tabelle E.1: Detaillierte Evaluation Variante *Start.UA*

Ereignis	Nachricht	Nachrichtengröße/KB	Empfänger
bis hierher wie *Start.UA*			
Beitritt Rad^A			
Join-Protokoll einleiten	sequentialJoinCallback	**7,7**	Gyn^A
Sequentielle Rückmeldung	sequentialJoinSynchronisation	**5,4**	Rad^A
ab hier wie *Start.UA*			

Tabelle E.2: Detaillierte Evaluation Variante *Join.UA*

Ereignis	Nachricht	Nachrichtengröße/KB	Empfänger
bis hierher wie *Join.UA*			
Mammographie			
Mammographiebericht als Payload hinzufügen	changeAlphaCardDescriptorEvent	8,2	Rad^A
	changePayloadEvent	85,8	Rad^A
	deliveryAcknowledgement	**5,8**	Rad^A
	deliveryAcknowledgement	**5,8**	Rad^A
	changeAlphaCardDescriptorEvent	9,9	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
	changePayloadEvent	87,6	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
α-Card auf gültig & valide setzen	changeAlphaCardDescriptorEvent	8,2	Rad^A
	changeAlphaCardDescriptorEvent	8,2	Rad^A
	deliveryAcknowledgement	**5,8**	Rad^A
	deliveryAcknowledgement	**5,8**	Rad^A
	changeAlphaCardDescriptorEvent	9,9	Gyn^A
	deliveryAcknowledgement	**5,9**	Gyn^A
	changeAlphaCardDescriptorEvent	9,9	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A

Tabelle E.3: Detaillierte Evaluation Variante *Join.Ack*

Ereignis	Nachricht	Nachrichtengröße/KB	Empfänger
bis hierher wie *Start.UA*			
Beitritt Rad^A			
Join-Protokoll einleiten	sequentialJoinCallback	6,5	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
	deliveryAcknowledgement	**5,8**	Gyn^A
Sequentielle Rückmeldung	sequentialJoinSynchronisation	241	Rad^A
ab hier wie *Join.Ack*			

Tabelle E.4: Detaillierte Evaluation Variante *Start.Ack*

Literaturverzeichnis

[Abb94] Abbate, Janet: *From Arpanet to Internet: A History of Arpa-sponsored Computer Networks, 1966-1988*. University of Pennsylvania, 1994 (UNI dissertation services)

[AH02] Aalst, Wil van d. ; Hee, Kees v.: *Workflow Management: Models, Methods, and Systems (Cooperative Information Systems series)*. MIT Press, 2002

[AWG05] Aalst, Wil van d. ; Weske, Mathias ; Grünbauer, Dolf: Case handling: A new paradigm for business process support. In: *Data and Knowledge Engineering* 53 (2005), Nr. 2, S. 129–162

[BBG+95] Berenson, Hal ; Bernstein, Phil ; Gray, Jim ; Melton, Jim ; O'Neil, Elizabeth ; O'Neil, Patrick: A critique of ANSI SQL isolation levels. In: *Proceedings of the 1995 ACM SIGMOD international conference on Management of data*. New York, NY, USA : ACM, 1995 (SIGMOD '95), S. 1–10

[BBKS08] Bengel, Günther ; Baun, Christian ; Kunze, Marcel ; Stucky, Karl-Uwe: *Masterkurs Parallele und Verteilte Systeme: Grundlagen und Programmierung von Multicoreprozessoren, Multiprozessoren, Cluster und Grid*. Vieweg+Teubner Verlag, 2008

[BCJ+81] Bux, Werner ; Closs, Felix ; Janson, Philippe ; Kummerle, Karl ; Muller, Hans R.: Reliable token-ring system for local-area communication. In: *NTC Conference Record - National Telecommunications Conference* 1 (1981), S. A2.2.1 – A2.2.6

[Böh09] Böhling, Benjamin: *Management mobiler Prozesse: Logging, Monitoring und Recovery*, Universität Hamburg, Diplomarbeit, 2009

[BMM90] Bochmann, Gregor v. ; Mondain-Monval, Pierre: Design Principles for Communication Gateways. In: *IEEE Journal on Selected Areas in Communications* 8 (1990), Nr. 1, S. 12–21

[Bor86] Borning, Alan H.: Classes versus prototypes in object-oriented languages. In: *Proc of 1986 ACM Fall joint computer conference* IEEE Computer Society Press, 1986, S. 36–40

[Cer69] Cerf, Vint G.: *ASCII format for network interchange*. RFC 20. `http://www.ietf.org/rfc/rfc20.txt`. Version: October 1969 (Request for Comments)

[Coa99] Coalition, The Workflow M.: *Workflow Management Coalition Terminology & Glossary*. Februar 1999

[Cro82] Crocker, David H.: *STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES*. RFC 822 (Standard). `http://www.ietf.org/rfc/rfc822.txt`. Version: August 1982 (Request for Comments). – Obsoleted by RFC 2822, updated by RFCs 1123, 2156, 1327, 1138, 1148

[DSM83] Dixon, Robert C. ; Strole, Norman C. ; Markov, J. D.: A token-ring network for local data communications. In: *IBM Syst. J.* 22 (1983), März, Nr. 1-2, S. 47–62

[FB96a] Freed, N. ; Borenstein, N.: *Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples*. RFC 2049 (Draft Standard). `http://www.ietf.org/rfc/rfc2049.txt`. Version: November 1996 (Request for Comments)

[FB96b] Freed, N. ; Borenstein, N.: *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. RFC 2045 (Draft Standard). `http://www.ietf.org/rfc/rfc2045.txt`. Version: November 1996 (Request for Comments). – Updated by RFCs 2184, 2231, 5335, 6532

[FB96c] Freed, N. ; Borenstein, N.: *Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types*. RFC 2046 (Draft Standard). `http://www.ietf.org/rfc/rfc2046.txt`. Version: November 1996 (Request for Comments). – Updated by RFCs 2646, 3798, 5147

[FKP96] Freed, N. ; Klensin, J. ; Postel, J.: *Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures*. RFC 2048 (Best Current Practice). `http://www.ietf.org/rfc/rfc2048.txt`. Version: November 1996 (Request for Comments). – Obsoleted by RFCs 4288, 4289, updated by RFC 3023

[FP78] Feinler, Elizabeth ; Postel, Jon: *ARPANET Protocol Handbook*. SRI International, Network Information Center, 1978

[FRH10] Freund, Jakob ; Rücker, Bernd ; Henninger, Thomas: *Praxishandbuch BPMN: Incl. BPMN 2.0*. Carl Hanser Verlag GmbH & CO. KG, 2010

[Gad01] Gadatsch, Andreas: *Management von Geschäftsprozessen. Methoden und Werkzeuge für die IT-Praxis: Eine Einführung für Studenten und Praktiker*. Vieweg Friedrich & Sohn Verlagsgesellschaft mbH, 2001

[Gar02] Garg, Vijay K.: *Elements of Distributed Computing*. 1. Wiley-IEEE Press, 2002

[GHS95] Georgakopoulos, Dimitrios ; Hornick, Mark F. ; Sheth, Amit P.: An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure. In: *Distributed and Parallel Databases* 3 (1995), Nr. 2, S. 119–153

[GM82] Garcia-Molina, Hector: Elections in a Distributed Computing System. In: *IEEE Trans. Comput.* 31 (1982), Januar, S. 48–59

[Had11] Hady, Scott A.: *Logical Unit Oriented Version Control System with Version Validity Support*, Lehrstuhl für Informatik 6 (Datenmanagement), Friedrich-Alexander-Universität Erlangen-Nürnberg, Diplomarbeit, 2011

[Hal05] Halsall, Fred: *Computer Networking and the Internet (5th Edition)*. 5th. Addison Wesley, 2005

[Han10] Hanisch, Stefan: *Konzeption und Implementierung einer Infrastruktur für aktive Dokumente*, Lehrstuhl für Informatik 6 (Datenmanagement), Friedrich-Alexander-Universität Erlangen-Nürnberg, Diplomarbeit, Oktober 2010

[HE10] Hellmann, Wolfgang (Hrsg.) ; Eble, Susanne (Hrsg.): *Ambulante und Sektoren übergreifende Behandlungspfade: Konzepte / Umsetzung / Praxisbeispiele*. 1. Mwv Medizinisch Wissenschaftliche Verlagsges., 2010

[HKO+70] Heart, Frank E. ; Kahn, Roger E. ; Ornstein, Severo M. ; Crowther, William R. ; Walden, David C.: The interface message processor for the ARPA computer network. In: *Proceedings of the May 5-7, 1970, spring*

joint computer conference. New York, NY, USA : ACM, 1970 (AFIPS '70 (Spring)), S. 551–567

[HNM12] Hansen, T. ; Newman, C. ; Melnikov, A.: *Internationalized Delivery Status and Disposition Notifications*. RFC 6533 (Proposed Standard). `http://www.ietf.org/rfc/rfc6533.txt`. Version: Februar 2012 (Request for Comments)

[HR83] Haerder, Theo ; Reuter, Andreas: Principles of transaction-oriented database recovery. In: *ACM Comput. Surv.* 15 (1983), Dezember, Nr. 4, S. 287–317

[HS04] Hagen, Cornelia R. ; Stucky, Wolffried: *Business-Process- und Workflow-Management: Prozessverbesserung durch Prozess-Management*. Vieweg+Teubner Verlag, 2004 (Lehrbuch Informatik)

[HV04] Hansen, T. ; Vaudreuil, G.: *Message Disposition Notification*. RFC 3798 (Draft Standard). `http://www.ietf.org/rfc/rfc3798.txt`. Version: Mai 2004 (Request for Comments). – Updated by RFCs 5337, 6533

[Int08] International Organization for Standardization: *Information Technology — Document Description and Processing Languages — Office Open XML File Formats — Part 1: Fundamentals and Markup Language Reference*. ISO/IEC 29500-1:2008, November 2008

[Isr09] Israel, Jeff: *Open Icon Library*. November 2009. – `http://openiconlibrary.sourceforge.net/` - zuletzt abgerufen am 11.07.2012

[Köh06] Köhl, Klaus: *Integrierte Versorgung im deutschen Gesundheitswesen: Von Managed Care zu einer neuen Wettbewerbsform in der GKV*. 1. VDM Verlag Dr. Müller, 2006

[Kla11] Klarl, Heiko: *Zugriffskontrolle in Geschäftsprozessen : Ein modellgetriebener Ansatz*. Vieweg+Teubner Verlag, 2011

[Kle08] Klensin, J.: *Simple Mail Transfer Protocol*. RFC 5321 (Draft Standard). `http://www.ietf.org/rfc/rfc5321.txt`. Version: October 2008 (Request for Comments)

[Kly98] Klyne, Graham: *Scenarios for Internet fax message confirmation.* April 1998

[KME+09] Klarl, Heiko ; Molitorisz, Korbinian ; Emig, Christian ; Klinger, Karsten ; Abeck, Sebastian: Extending Role-based Access Control for Business Usage. In: *Third International Conference on Emerging Security Information, Systems and Technologies*, 2009, S. 136–141

[KR02] Kurose, James ; Ross, Keith: *Computernetze. Ein Top-Down-Ansatz mit Schwerpunkt Internet.* Pearson Studium, 2002

[KSAH+11] Karneges, Justin ; Saint-Andre, Peter ; Hildebrand, Joe ; Forno, Fabio ; Cridland, Dave ; Wild, Matthew: *Stream Management.* XEP-0198, Juni 2011

[Lam78] Lamport, Leslie: Time, clocks, and the ordering of events in a distributed system. In: *Communications of the ACM* 21 (1978), Nr. 7, S. 558–565

[Lam87] Lamport, Leslie: *Distribution.* Mai 1987

[LED+99] LaMarca, Anthony ; Edwards, W. K. ; Dourish, Paul ; Lamping, John ; Smith, Ian ; Thornton, Jim: Taking the work out of workflow: mechanisms for document-centered collaboration. In: *6th conference on European Conference on Computer Supported Cooperative Work*, Kluwer Academic Publishers Norwell, USA, 1999, S. 1–20

[Len09] Lenz, Richard: Information systems in healthcare - state and steps towards sustainability. In: *Yearbook of medical informatics* 1 (2009), S. 63–70

[Len10] Lenz, Richard: *Vorlesungsskript Evolutionäre Informationssysteme.* 2010

[MCC+72] McQuillan, John M. ; Crowther, William R. ; Cosell, Bernard P. ; Walden, David C. ; Heart, Frank E.: Improvements in the design and performance of the ARPA network. In: *Proceedings of the December 5-7, 1972, fall joint computer conference, part II.* New York, NY, USA : ACM, 1972 (AFIPS '72 (Fall, part II)), S. 741–754

[Mik09] Mik, Eliza: The Effectiveness of Acceptances Communicated by Electronic Means, or — Does the Postal Acceptance Rule Apply to Email? In: *Journal of Contract Law* 26 (2009), S. 68–96

[Moo96] Moore, K.: *MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text*. RFC 2047 (Draft Standard). `http://www.ietf.org/rfc/rfc2047.txt`. Version: November 1996 (Request for Comments). – Updated by RFCs 2184, 2231

[Moo03] Moore, K.: *Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)*. RFC 3461 (Draft Standard). `http://www.ietf.org/rfc/rfc3461.txt`. Version: Januar 2003 (Request for Comments). – Updated by RFCs 3798, 3885, 5337, 6533

[MW08] Miers, Derek ; White, Stephen A.: *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. Future Strategies Inc, 2008

[NL09] Neumann, Christoph P. ; Lenz, Richard: alpha-Flow: A Document-based Approach to Inter-Institutional Process Support in Healthcare. In: ´09, ProHealth (Hrsg.): *Proceedings of the 3rd Int'l Workshop on Process-oriented Information Systems in Healthcare in conjunction with the 7th Int'l Conf on Business Process Management*, 2009, S. –

[NL10] Neumann, Christoph P. ; Lenz, Richard: The Alpha-Flow Use-Case of Breast Cancer Treatment - Modeling Inter-Institutional Healthcare Workflows by Active Documents. In: *Proceedings of the 2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, IEEE Computer Society, 2010 (WETICE '10), S. 17–22

[NL12] Neumann, Christoph P. ; Lenz, Richard: The alpha-Flow Approach to Inter-Institutional Process Support in Healthcare. In: *International Journal of Knowledge-Based Organizations (IJKBO)* 2 (2012), Nr. 3

[NM08] Newman, C. ; Melnikov, A.: *Internationalized Delivery Status and Disposition Notifications*. RFC 5337 (Experimental). `http://www.ietf.org/rfc/rfc5337.txt`. Version: September 2008 (Request for Comments). – Obsoleted by RFC 6533

[NMC+99] Nadkarni, Prakash M. ; Marenco, Luis ; Chen, Roland ; Skoufos, Emmanouil ; Shepherd, Gordon ; Miller, Perry: Organization of heterogeneous scientific data using the EAV/CR representation. In: *Journal of the American Medical Informatics Association* 6 (1999), Nr. 6, S. 478–493

[NS05] Niemeyer, Anna ; Stettin, Jürgen: Telematik und Qualität - Einführung einer Telematikplattform in Hamburg. In: *Bundesgesundheitsblatt* 48 (2005), S. 761–770

[NSWL11] Neumann, Christoph P. ; Schwab, Peter K. ; Wahl, Andreas M. ; Lenz, Richard: 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)*, 2011, S. –

[NWL12] Neumann, Christoph P. ; Wahl, Andreas ; Lenz, Richard: Adaptive Version Clocks and the OffSync Protocol. In: IEEE (Hrsg.): *Proc of the 10th IEEE Int'l Symposium on Parallel and Distributed Processing with Applications (ISPA-12)*, 2012, S. –

[Obj11] Object Management Group: *Business Process Model And Notation (BPMN) Version 2.0.* `http://www.omg.org/spec/BPMN/2.0/PDF`. Version: Januar 2011

[Pat02] Patel, Nandish V.: *Adaptive Evolutionary Information Systems.* Idea Group Inc, 2002

[PF05] Peschel-Findeisen, Thomas: *Nebenläufige und Verteilte Systeme: Theorie und Praxis.* 1. Mitp-Verlag, 2005

[Pos82] Postel, Jonathan: *Simple Mail Transfer Protocol.* RFC 821 (Standard). `http://www.ietf.org/rfc/rfc821.txt`. Version: August 1982 (Request for Comments). – Obsoleted by RFC 2821

[RA81] Ricart, Glenn ; Agrawala, Ashok K.: An optimal algorithm for mutual exclusion in computer networks. In: *Commun. ACM* 24 (1981), Januar, S. 9–17

[Res01] Resnick, P.: *Internet Message Format.* RFC 2822 (Proposed Standard). `http://www.ietf.org/rfc/rfc2822.txt`. Version: April 2001 (Request for Comments). – Obsoleted by RFC 5322, updated by RFCs 5335, 5336

[RQZ07] Rupp, Chris ; Queins, Stefan ; Zengler, Barbara: *UML 2 glasklar: Praxiswissen für die UML-Modellierung*. Carl Hanser Verlag GmbH & CO. KG, 2007

[SA11a] Saint-Andre, P.: *Extensible Messaging and Presence Protocol (XMPP): Core*. RFC 6120 (Proposed Standard). `http://www.ietf.org/rfc/rfc6120.txt`. Version: März 2011 (Request for Comments)

[SA11b] Saint-Andre, P.: *Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence*. RFC 6121 (Proposed Standard). `http://www.ietf.org/rfc/rfc6121.txt`. Version: März 2011 (Request for Comments)

[SAH11] Saint-Andre, Peter ; Hildebrand, Joe: *Message Delivery Receipts*. XEP-0184, März 2011

[SAST09] Saint-Andre, Peter ; Smith, Kevin ; TronCon, Remko: *XMPP: The Definitive Guide: Building Real-Time Applications with Jabber Technologies*. 1. O'Reilly Media, 2009

[Sch07] Schweiger, Andreas: *Agentenbasiertes Konstruieren von verteilten Systemen am Beispiel Gesundheitswesen*, Lehrstuhl für Wirtschaftsinformatik (I 17), Technische Universität München, Diss., 2007

[Sch08a] Schicker, Günther: *Koordination und Controlling in Praxisnetzen mithilfe einer prozessbasierten E-Service-Logistik*. Gabler, 2008 (Gabler Edition Wissenschaft)

[Sch08b] Schicker, Günther: Praxisnetze im Gesundheitswesen. In: Schubert, Herbert (Hrsg.): *Netzwerkmanagement*. VS Verlag für Sozialwissenschaften, 2008, S. 146–166

[Sch11] Schwab, Peter K.: *Ein adaptives Attributmodell als Baustein einer Prozessunterstützung auf Basis von aktiven Dokumenten*, Lehrstuhl für Informatik 6 (Datenmanagement), Friedrich-Alexander-Universität Erlangen-Nürnberg, Diplomarbeit, 2011

[SGW09] Sobin, Leslie H. (Hrsg.) ; Gospodarowicz, Mary K. (Hrsg.) ; Wittekind, Christian (Hrsg.): *TNM Classification of Malignant Tumours (Uicc International Union Against Cancer)*. 7th. Wiley-Blackwell, 2009

[SRC84] Saltzer, Jerome H. ; Reed, David P. ; Clark, David D.: End-To-End Arguments in System Design. In: *ACM Trans. Comput. Syst.* 2 (1984), November, Nr. 4, S. 277–288

[SSD06] Stock, Johannes ; Steiner, Michael ; Daul, Gisela: Deutschland: Hausarztmodelle der zweiten Generation. In: *Managed Care* 8 (2006), S. 27–29

[Tan03] Tanase, Matthew: *IP Spoofing: An Introduction.* 2003. – `www.securityfocus.com/infocus/1674` - zuletzt abgerufen am 11.06.2012

[Tan07] Tanenbaum, Andrew S.: *Verteilte Systeme.* 2. Addison Wesley Verlag, 2007

[Tau11] Tauber, Arne: Elektronische Zustellung in Europa. In: *Datenschutz und Datensicherheit - DuD* 35 (2011), S. 774–778

[TN11] Todorova, Aneliya ; Neumann, Christoph P.: alpha-Props: A Rule-Based Approach to 'Active Properties' for Document-Oriented Process Support in Inter-Institutional Environments. In: Porada, Ludger (Hrsg.) ; Gesellschaft für Informatik (Veranst.): *Lecture Notes in Informatics (LNI) Seminars 10 / Informatiktage 2011* Gesellschaft für Informatik, 2011, S. –

[Tod10] Todorova, Aneliya: *Konzeption und Implementierung eines leichtgewichtigen und autonomen Regel-basierten Systems als eine Realisierung von "Active Properties" im Kontext von aktiven Dokumenten*, Lehrstuhl für Informatik 6 (Datenmanagement), Friedrich-Alexander-Universität Erlangen-Nürnberg, Diplomarbeit, 2010

[VST02] Van Steen, Maarten ; Tanenbaum, Andrew: *Distributed Systems: Principles and Paradigms.* Prentice Hall, 2002

[Wah11] Wahl, Andreas M.: *alpha-Offsync: Verteilte Datensynchronisation in Form von IMAP-basiertem Mailtransfer*, Lehrstuhl für Informatik 6 (Datenmanagement), Friedrich-Alexander-Universität Erlangen-Nürnberg, Bachelorarbeit, 2011

[Wie01] Wiehe, Henning: *Alte Form und neue Kommunikation. Die Schriftform wesentlicher Verfahrenshandlungen.* Lit Verlag, 2001