UNIVERSITÄT LEIPZIG

Institut für Medizinische Informatik, Statistik und Epidemiologie (IMISE)

Konstruktionspatterns für Informationssysteme im Gesundheitswesen

Diplomarbeit

Leipzig, August 2011

vorgelegt von:

Frank Stephan

geb. am: 02.03.1986

Betreuer:

Dipl.-Inf. Alexander Strübing

Inhaltsverzeichnis

Begriffs- und Abkürzungsverzeichnis

3LGM²	Three-layer graph-based metamodel / Drei-Ebenen-Metamodell; UnternehmensMetamodell für die Modellierung von Informationssystemen im Gesundheitswesen.
XDS	Cross-enterprise document sharing; Bezeichnung eines Integration Profile der IHE für den einrichtungsübergreifenden Dokumentenaustausch im Gesundheitswesen.
Deskriptivität	Eigenschaft eines Konstruktionspatterns, welche deren Repräsentation des Entwurfswissens in Modellform bezeichnet.
Konstruktivität	Eigenschaft eines Konstruktionspatterns, welche deren Einsatzmöglichkeit bei der Konstruktion von Modellen bezeichnet.
≡	Logische Äquivalenz („Genau dann, wenn“); Binäre Relation über logischen Ausdrücken, welche dann besteht, wenn beide Ausdrücke denselben Wahrheitswert besitzen.
∨	Logisches Oder; Binäre Relation über logischen Ausdrücken, welche dann besteht, wenn wenigstens einer der beiden Ausdrücke wahr ist.

1 Einleitung

1.1 Gegenstand und Motivation

1.1.1 Gegenstand

Die zentrale Aufgabe von Informationssystemen im Gesundheitswesen ist die Unterstützung einer bestmöglichen Patientenversorgung zu geringstnötigen Kosten durch effektive und zugleich effiziente Informationsprozesse. Diese Aufgabe repräsentiert zum einen die Motivation für das Erreichen einer hohen Qualität der Informationsverarbeitung und gibt zugleich die Anforderungen vor, denen bei allen Entscheidungen bezüglich der Entwicklung eines Informationssystems Rechnung getragen werden muss.

Informationsverarbeitung im Allgemeinen und insbesondere auch im Gesundheitswesen stellt einen Qualitätsfaktor dar. Dieser schlägt sich im Erreichen der gesetzten Anforderungen, wie etwa konkrete Änderungen am Gesundheitszustand des Patienten sowie in darüber hinaus gehenden Zielen, nieder. Das Maß der Erfüllung dieser Anforderungen wird als *Ergebnisqualität* bezeichnet. Den Beitrag hierzu leisten Informationssysteme durch die Effektivität und Effizienz der Informationsprozesse, gekennzeichnet unter anderem durch die Möglichkeit der mehrfachen Verwendung von Daten, dem Vermeiden von Medienbrüchen und insbesondere der Verfügbarkeit aller relevanten Informationen am richtigen Ort und zur richtigen Zeit [vgl. WINTER et al. 2011]. Das Erreichen einer hohen Ergebnisqualität hängt damit entscheidend von der Güte der Informationsprozesse (*Prozessqualität*) ab, wie es bereits in [Lehmann et al. 2005] formuliert wurde: „Medizinische Versorgung auf qualitativ hohem Niveau ist heute ohne die systematische Informationserfassung, -aufbereitung, und -verarbeitung nicht mehr möglich“. Systematische Informationssysteme erfordern allerdings auch immer eine systematische Planung. Dabei ist es Aufgabe des Informationsmanagements, deren Entwicklung gezielt zu lenken, genauer gesagt ist es notwendig, strukturelle Grundlagen zu schaffen, durch die dann erst eine hohe Qualität der Prozesse ermöglicht werden kann. Dazu zählt etwa die Angepasstheit der Anwendungssysteme an die zu unterstützenden Aufgaben, die Integrität der Daten oder auch die leichte Anpassbarkeit des Informationssystems selbst an neue Umstände.

Ein wichtiges Hilfsmittel hierbei ist die Modellierung. Sie versetzt das Informationsmanagement in die Lage, das Informationssystem aus einer abstrakten Sichtweise heraus zu betrachten, es analytisch zu bewerten und notwendige Änderungen zu identifizieren und zu planen. Im Bereich der Krankenhausinformationssysteme hat sich hier das Tool3LGM² [3LGM² TOOL] bewährt, welches auf dem Drei-Ebenen-Metamodell (kurz: „3LGM²“) [WINTER et al. 2003] basiert und der systematischen Dokumentation von Komponenten und deren Beziehungen innerhalb eines solchen Informationssystems dient (Abbildung 1). Eine weitere Möglichkeit für die Unterstützung der Planung und Entwicklung von Informationssystemen ist die Nutzung von Entwurfswissen. Typische Beispiele hierfür sind „Common Object Request Broker Architecture“ [OMG CORBA], OSI-Referenzmodell, oder – speziell im Bereich des Gesundheitswesens – die *Technical Frameworks* der IHE („Integrating the Healthcare Enterprise”) [IHE TF]. All diese umfassen bewährte Konzepte, Standards, „Best-Practise“-Verfahren etc. und stellen damit eine Richtlinie und Anleitung für die notwendigen Planungsaktivitäten bereit. Demnach dienen sowohl die Modellierung als auch der Einsatz von Entwurfswissen der Definition einer adäquaten Architektur von Informationssystemen, wodurch insbesondere die oben aufgeführten strukturellen Grundlagen – wie etwa Datenintegrität oder Funktionskonformität der Anwendungssysteme – erreicht werden und damit zur *Strukturqualität* beigetragen wird.

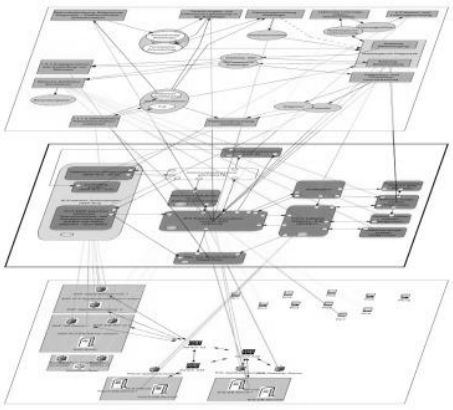

Abbildung 1 – Beispielmodell im Tool3LGM² (Quelle: www.3lgm2.de)

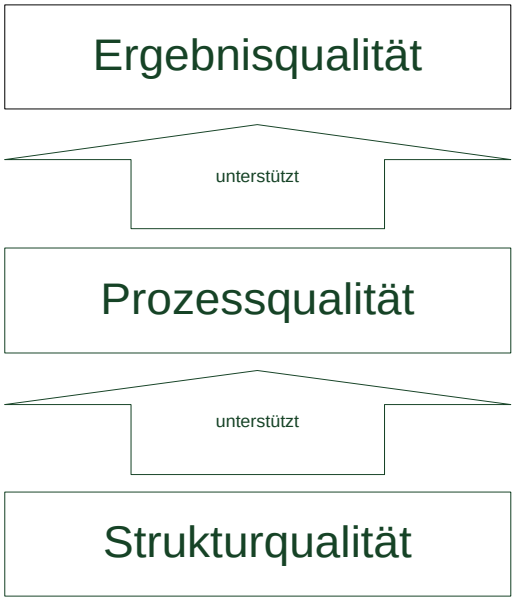

Abbildung 2 – Qualitätsstufen (für Informationssysteme) [vgl. DONABEDIAN 1982]

Als Folge profitieren dann ebenfalls die Prozessqualität und aufbauend darauf die Ergebnisqualität (Abbildung 2).

Eine wichtige Rolle im Hinblick auf die strukturelle Qualität nimmt dabei die *Integration* ein. Einrichtungen im Gesundheitswesen – insbesondere Krankenhäuser – sind historisch gewachsen, und mit ihnen auch ihre Informationssysteme, was zur Folge hat, dass die jeweilige Form der Unterstützung der Prozesse ein eher zufälliges Resultat dieser Entwicklung ist. Genauer gesagt, mangelt es an notwendiger Standardisierung und Zusammenfassung der beteiligten Anwendungssysteme. Dies hat eine schlechte Qualität der Prozesse zur Folge, unter anderem ausgelöst durch Medienbrüche, inadäquate Aufgabenunterstützung, mehrfache Erfassung derselben Daten etc., was zur potentiellen Verminderung der Ergebnisqualität führt und damit auch Einfluss auf die Patientenversorgung selbst hat.

Die IHE ist eine Initiative, welche sich in diesem Zusammenhang der Verbesserung des computerunterstützten Informationsaustausches im Gesundheitswesen angenommen hat. Kerngedanke ist der koordinierte Einsatz etablierter (Defacto-) Standards wie HL7 [HL7], DICOM [DICOM] oder ebXML [EB XML] sowie deren Anpassung auf bestimmte Bedürfnisse im klinischen Umfeld mit dem Ziel der Unterstützung einer optimalen Patientenversorgung. Die IHE veröffentlicht hierzu verschiedene sogenannte Technical Frameworks, welche unterschiedliche Problembereiche bezüglich deren Integration adressieren, wie zum Beispiel „Patient Care Coordination“, „Cardiology“ oder auch „IT Infrastructure“. Letzteres etwa beschreibt die Anforderungen an die Informations- und Kommunikationstechnologie selbst und zeigt Wege auf, hin zu einer gezielten Zusammenarbeit der beteiligten Computersysteme. Technical Frameworks werden von der IHE in Textform bereitgestellt. Dabei untergliedern sie sich jeweils in verschiedene Profile – sogenannte *Integration Profiles* – welche einen abgeschlossenen Bereich innerhalb der jeweiligen Domäne abdecken. Diese Profile beinhalten konkrete Anforderungen an die zu verwendenden IT-Produkte hinsichtlich ihrer Interoperabilität und der jeweils einzusetzenden Standards. Damit stellt die IHE einen umfangreichen Vorrat an Entwurfswissen für die Architektur der Informationssysteme bereit, hin zu einer besseren Strukturqualität und einer höheren Güte der darauf aufbauenden Prozesse.

Dabei spielt Integration nicht nur innerhalb von Einrichtungen eine wichtige Rolle:

Gesundheitssysteme weltweit stehen durch eine wachsende Mobilität der Bevölkerung neuen Herausforderungen gegenüber, unter anderem der Notwendigkeit, medizinische Daten überall dort verfügbar zu machen, wo sie bei der Versorgung eines Patienten erforderlich sind. So ist es notwendig, medizinische Dokumente – wie Arztbriefe, Befunde oder beispielsweise auch radiologische Bilddaten – am Standort des Patienten, zum Zeitpunkt, an dem sie benötigt werden, abfragen zu können, um eine

bessere Grundlage für weitere Entscheidungen zu erlangen. Ebenfalls gilt es, diese Datenverfügbarkeit bei einer verteilten Behandlung sicherzustellen. Häufig wird ein Patient im Rahmen eines Behandlungsfalles mehrere verschiedene Einrichtungen durchlaufen, zu denen neben Krankhäusern auch Arztpraxen, Physiotherapien, Rehabilitationseinrichtungen und weitere zählen. Speziell in diesem Zusammenhang sind es oft Entlassungsbescheide, welche bei der Folgeversorgung benötigt werden, dabei insbesondere Informationen zu bereits durchgeführten Maßnahmen und erstellten Befunden. Diese müssen den Patienten durch alle Stationen der Versorgung hindurch begleiten.

Zu dieser Problematik entwickelte die IHE das *XDS Integration Profile* (kurz für: „Cross-enterprise Document Sharing Integration Profile", [XDS]), welches sich innerhalb des Integrationskontextes von Informations- und Kommunikationstechnologie mit dem einrichtungsübergreifenden Austausch klinischer Dokumente befasst. Der Fokus liegt dabei auf der Identifikation von Rollen der beteiligten Einrichtungen – etwa „Document Source" oder „Document Consumer" – und den jeweils erforderlichen Funktionen, die durch sie unterstützt werden müssen, wie zum Beispiel „Provide and Register Document" oder „Retrieve Document"; hier im Szenario des Bereitstellens bzw. Abfragens der Dokumente. Darüber hinaus werden Methoden zur zentralen Registrierung von Dokumenten und zur Patientenidentifikationsverwaltung vorgestellt, um somit den Zugriff auf die Dokumente sowie deren Zuordnung zum Patienten über Einrichtungsgrenzen hinweg zu ermöglichen. Durch die so bereitgestellten medizinischen Daten – etwa von bereits durchgeführten Untersuchungen – ist eine bessere Entscheidungsgrundlage gegeben, wodurch eine adäquate Behandlung unterstützt und damit zur Steigerung der Qualität der Patientenversorgung beigetragen wird.

1.1.2 Problematik

Informationsverarbeitung ist ein Qualitätsfaktor, der sowohl durch die Güte der zugrundeliegenden Strukturen als auch durch die darauf aufbauenden Informationsprozesse einen erheblichen Einfluss auf das Erreichen der Ziele im Gesundheitswesen bzw. in den jeweiligen Einrichtungen hat. Informationssysteme als Gesamtheit der Informationsverarbeitung müssen daher systematisch durch das Informationsmanagement geplant werden, um den Anforderungen gerecht werden zu können. Aufgrund der Entwicklung des Gesundheitswesens bzw. dessen Informationssystemen ist damit ein Integrationsbedarf gegeben, der eine Anpassung der strukturellen Gegebenheiten hin zu effektiveren und effizienteren Informationsprozessen verlangt. Hierbei ist es von großer Bedeutung, entsprechendes Entwurfswissen in die notwendigen Planungsaktivitäten einfließen zu lassen, um von den bewährten Konzepten profitieren zu können.

Die Integration Profiles der IHE vermitteln dazu im Allgemeinen die für das Management der IT-Systeme relevanten Grundlagen für die Planung der Integration und stellen darüber hinaus die technischen Spezifikationen für deren Entwickler bereit, wodurch es möglich ist, zunächst eine geeignete Architektur zu entwickeln, und anschließend diese zu implementieren. Auf der einen Seite wird damit ein breit gefächerter und zugleich sehr detaillierter Einblick in die jeweilige Thematik gegeben, auf der anderen Seite allerdings, durch die daraus resultierende hohe Komplexität, der Zugang zu den Integration Profiles erschwert. Insbesondere sind die Zusammenhänge zwischen konzeptioneller Ebene und Implementierungsrichtlinien oft nur schwierig nachzuvollziehen, was den Weg von der Planung hin zur Umsetzung kompliziert.

Besser geeignet wäre ein Ansatz zum Einbeziehen derartigen Entwurfswissens mit Unterstützung durch Modellierung. Modelle – etwa auf Basis des 3LGM² – stellen eine geeignete Möglichkeit dar, Informationssysteme zu dokumentieren, zu analysieren und Planungsschritte abzuleiten. Demnach ist es nur naheliegend, Entwurfswissen ebenfalls über die Modellierung in die Planungsaktivitäten zu involvieren. Das Problem hierbei ist, dass zwar sowohl das Arbeiten mit Informationssystemmodellen beispielsweise durch das Tool3LGM² möglich ist und das nötige Entwurfswissen durch die IHE bereitgestellt wird, es allerdings keine Möglichkeit gibt, dieses geeignet in den Modellierungsprozess einbeziehen zu können; geeignet in dem Sinne, dass sowohl ***deskriptive*** Aspekte beachtet werden, genauer gesagt, das Entwurfswissen selbst als Modell abgebildet wird, und ***konstruktive*** Aspekte, sodass das so formalisierte Wissen im Tool3LGM² direkt abrufbar ist und unterstützend bei der Modellierung der Informationssysteme eingesetzt werden kann.

1.1.3 Motivation

Das Nutzbarmachen von Entwurfswissen in der eben beschriebenen Form bringt drei entscheidende Vorteile:

1. Das Wissen wird in Modellform festgehalten, wodurch eine schematisch übersichtliche Darstellung der enthaltenen Konzepte erzielt wird.
2. Durch die Möglichkeit dieses konstruktiv in den Modellierungsprozess einbeziehen zu können, wird ein direktes Übertragen dieser Konzepte in Informationssystemmodelle gewährleistet.
3. Entwurfswissen, auf diese Weise formalisiert, kann beliebig oft wiederverwendet werden und ermöglicht so eine schnellere Planung geeigneter Architekturen für Informationssysteme.

Diese Vorteile schlagen sich insbesondere in den Möglichkeiten des Informationsmanagements nieder, eine systematische Informationsverarbeitung zu etablieren, ausgehend von den so gegebenen strukturellen Grundlagen, aus denen eine höhere Güte der Informationsprozesse hervorgeht und damit die Möglichkeit für eine Steigerung der Ergebnisqualität geschaffen wird.

Das Konzept der Modellierung von Informationssystemen kann somit um eine wesentliche Komponente erweitert werden – dem gezielten Einsatz von Entwurfswissen für die notwendigen Planungsaktivitäten.

Vor allem im Gesundheitswesen nehmen in diesem Zusammenhang die Technical Frameworks bzw. Integration Profiles der IHE eine wichtige Rolle ein. Integration in Informationssystemen ist ein wesentlicher Aspekt, der sich in allen der drei Qualitätsstufen niederschlägt. Über das durch die IHE bereitgestellte Entwurfswissen kann eine Optimierung der Architektur des Informationssystems erreicht werden, sprich, es wird eine hohe Strukturqualität erzielt und damit die Grundlage für effiziente Prozesse und daraus resultierend eine bestmögliche Unterstützung für das Erreichen der jeweiligen Ziele der Einrichtung und des gesamten Gesundheitswesens geboten. Durch den Einsatz von Modellierung gewinnen dabei die notwendigen Planungsaktivitäten an Systematik, wodurch wiederrum die Voraussetzung für ein systematisch funktionierendes Informationssystem und eine entsprechende Informationsverarbeitung geschaffen werden.

Insbesondere wird die Bedeutung der Kombination von Modellierung und Entwurfswissen am obigen Beispiel des einrichtungsübergreifenden Dokumentenaustausches deutlich: Die Nichtverfügbarkeit behandlungsrelevanter Informationen bedeutet eine Einschränkung in den Möglichkeiten der Versorgung von Patienten. Mehrfachuntersuchungen, unzureichende Sicherheit und Verzögerungen in der Behandlungsplanung sowie kritische Fehlentscheidungen im Notfall sind die Folge. Entsprechend notwendig ist es, den Zugriff auf eben diese Informationen zu ermöglichen. Das XDS Integration Profile zeigt dazu Wege auf, einen transinstitutionellen, computerunterstützten Dokumentenaustausch zu planen und umzusetzen, hat allerdings – ausgelöst durch die hohe Komplexität – den Nachteil der schweren Nachvollziehbarkeit und Unübersichtlichkeit der fachlichen und technischen Konzepte und darüber hinaus – bedingt durch die textuelle Form – eine nicht ausreichende Unterstützung für einen systemtischen Planungsprozess. Aufgrund der Eignung des Tool3LGM² zur Modellierung von Informationssystemen im Gesundheitswesen ergäbe sich durch die beschriebene Verwendung des Entwurfswissens die Chance, das Informationsmanagement gezielt bei der systematischen Planung zu unterstützen.

Somit kann ein wesentlicher Schritt hin zu einem einrichtungsübergreifenden Dokumentenaustausch gemacht werden, und des Weiteren neue Perspektiven für den Einsatz weiterer IHE Profile und auch darüber hinausgehendem Entwurfswissen eröffnet werden.

1.2 Problemstellung

- Problem P1: Es mangelt an einem geeigneten Ansatz, konstruktives Entwurfswissen in die Modellierung von Informationssystemen mittels Tool3LGM² einbeziehen zu können.

- Problem P2: Der transinstitutionelle Dokumentenaustausch stellt eine der großen Herausforderungen für die zukünftige Entwicklung im Gesundheitswesen dar, die mit hohem Planungsaufwand verbunden ist. Das XDS Integration Profile der IHE bietet hierfür die geeigneten Konzepte, welche aber durch dessen hohe Komplexität nur bedingt für die Planung einsetzbar sind.

1.3 Zielsetzung

Ziele zu Problem P1:

- Ziel Z1: Um Entwurfswissen im Tool3LGM² anwenden zu können, soll die Spezifikation eines Frameworks erfolgen, welches die Beschreibung von 3LGM²-konformen *Konstruktionspatterns* ermöglicht, die es erlauben, dieses Wissen in geeigneter Weise zu präsentieren und gleichzeitig direkt für die Modellierung einsetzbar sind.

Ziele zu Problem P2:

- Ziel Z2: Auf Basis von Z1 wird das Vorgehen geschildert für das Überführen des IHE XDS Integration Profile in ein solches Konstruktionspattern.

1.4 Aufgaben-/Fragestellung

Aufgaben zu Ziel Z1:

- Aufgabe A1.1: Zunächst wird eine kurze Übersicht zum 3-Ebenen-Metamodell (3LGM²) gegeben.

- Aufgabe A1.2: Aufbauend darauf, erfolgt die Einführung in die für diese Arbeit relevanten Funktionen des Tool3LGM².

- Aufgabe A1.3: Anschließend soll Klarheit über die Bedeutung von Entwurfswissen, Frameworks, Patterns und verwandter Begriffe, welche für das weitere Vorgehen vonnöten sind, geschaffen werden.

- Aufgabe A1.4: Auf Basis der vorherigen Arbeitspakete werden die Spezifikation des Frameworks zur Definition von Konstruktionspatterns vorgenommen und darüber hinaus die im Tool3LGM² umzusetzenden Funktionalitäten identifiziert.

Aufgaben zum Ziel Z2:

- Aufgabe A2.1: Als Grundlage für alle weiteren Schritte werden die allgemeinen Konzepte in den Integration Profiles der IHE vorgestellt.

- Aufgabe A2.2: Aufbauend darauf erfolgt eine Gegenüberstellung dieser Konzepte mit denen im 3-Ebenen-Metamodell.

- Aufgabe A2.3: Daraufhin wird ein kurzer Überblick zum IHE XDS Integration Profile gegeben.

- Aufgabe A2.4: Abschließend wird auf Basis des Frameworks aus A1.4 das XDS Integration Profile als exemplarische Anwendung formal in ein 3LGM²-konformes Konstruktionspattern überführt.

2 Einführung zum 3LGM²

Als ersten Schritt hin zu einer effizienten Nutzbarmachung von Entwurfswissen für die Planung und Entwicklung von Informationssystemen im Gesundheitswesen wird in diesem Kapitel in die Möglichkeiten zur Modellierung über das Drei-Ebenen-Metamodell und dem darauf aufbauenden Tool3LGM² eingeführt.

2.1 Das Drei-Ebenen-Metamodell

Das Drei-Ebenen-Metamodell, oder auch „three-layer graph-based metamodel“ (kurz: 3LGM²), dient der Beschreibung von Informationssystemen im Gesundheitswesen und bietet eine statische Repräsentation, welche essentielle Komponenten und deren Beziehungen beinhaltet. Diese werden drei unterschiedlichen Ebenen zugeordnet:

1. Die Fachliche Ebene beschreibt die Aufgaben, welche innerhalb einer Einrichtung ausgeführt werden sowie die Daten, die dabei verarbeitet werden.
2. Die Logische Werkzeugebene umfasst die Unterstützung dieser Aufgaben sowie die Kommunikation und Speicherung der Daten durch informationsverarbeitende Werkzeuge.
3. Auf der Physischen Werkzeugebene sind dann etwa PCs, Server, Aktenschränke, Personen und weitere gegenständliche Komponenten angeordnet, die die Informationsverarbeitung auf der Logischen Werkzeugebene ermöglichen bzw. durchführen.

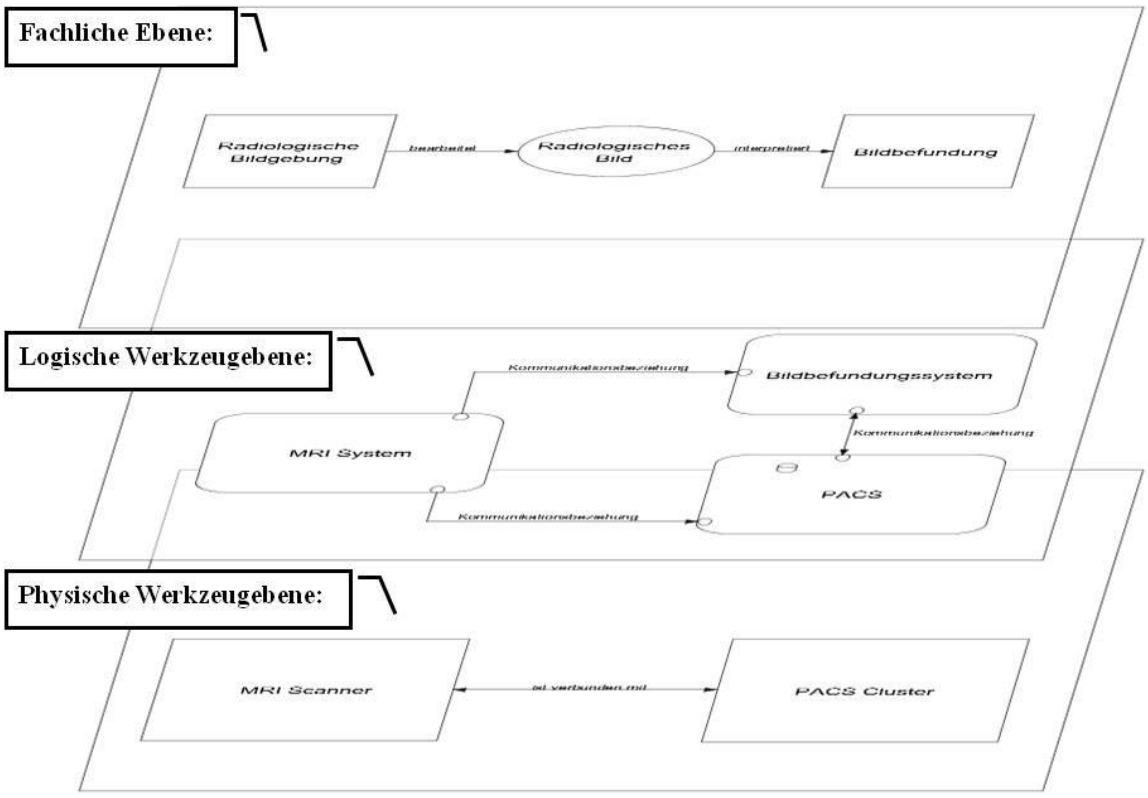

Abbildung 3 – 3LGM²-Modell eines Subinformationssystems der Radiologie mit Bildgebung, -befundung und -speicherung

Im Folgenden werden für jede der drei Ebenen, das Klassendiagramm, eine Erklärung zu den wesentlichen Klassen sowie ein Beispiel gegeben. Jedes Beispiel ist dabei eine Instanz der jeweiligen Ebene.

2.1.1 Die Fachliche Ebene

2.1.1.1 Klassendiagramm

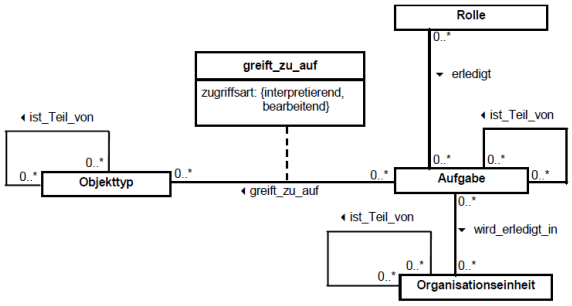

Abbildung 4 – Fachliche Ebene des 3LGM² (Quelle: [3LGM²])

2.1.1.2 Zentrale Klassen

Klasse	*Funktion*
Aufgabe	Die Aufgaben, die innerhalb einer Einrichtung erledigt werden
Objekttyp	Die Informationen, die durch die Aufgaben bearbeitet oder interpretiert werden
Organisationseinheit	Geben die organisatorische Struktur der Einrichtung wieder, inklusive der jeweils ausgeführten Aufgaben

2.1.1.3 Beispiel

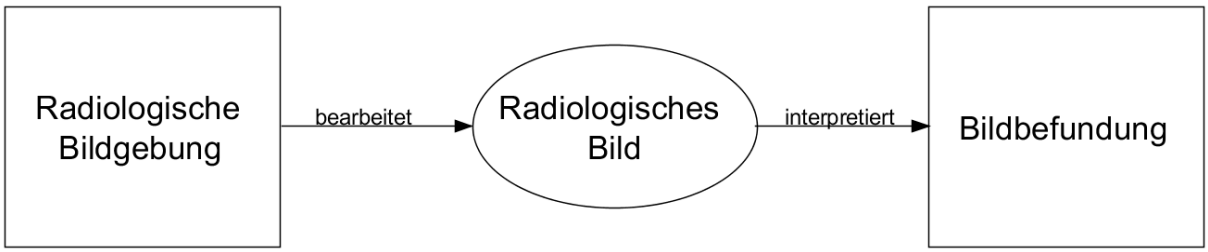

Abbildung 5 – Beispiel der Fachlichen Ebene

Das hier aufgeführte Beispiel zeigt die Fachliche Ebene aus Abbildung 3. Dargestellt werden darin die Aufgaben „Radiologische Bildgebung“ sowie „Bildbefundung“ und der Objekttyp „Radiologisches Bild“. Die Pfeile zwischen den Elementen symbolisieren hier die „greift_zu_auf“-Beziehung in jeweils einer der beiden Zugriffsarten. Die Bedeutung ist dabei, dass die Aufgabe der Radiologischen Bildgebung den Objekttyp Radiologisches Bild erzeugt. Dieser Zusammenhang wird durch den Zugriffstyp „bearbeitend“ ausgedrückt. Die Bildbefundung benötigt dann diese Radiologischen Bilder – sprich den Objekttyp Radiologisches Bild – um durchgeführt zu werden. Diese Beziehung wird dann wiederum durch den Zugriffstyp „interpretierend“ beschrieben. Die Art und Weise, wie dies konkret unterstützt wird und umgesetzt ist, wird auf der Logischen Werkzeugebene beleuchtet.

2.1.2 Die Logische Werkzeugebene

2.1.2.1 Klassendiagramm

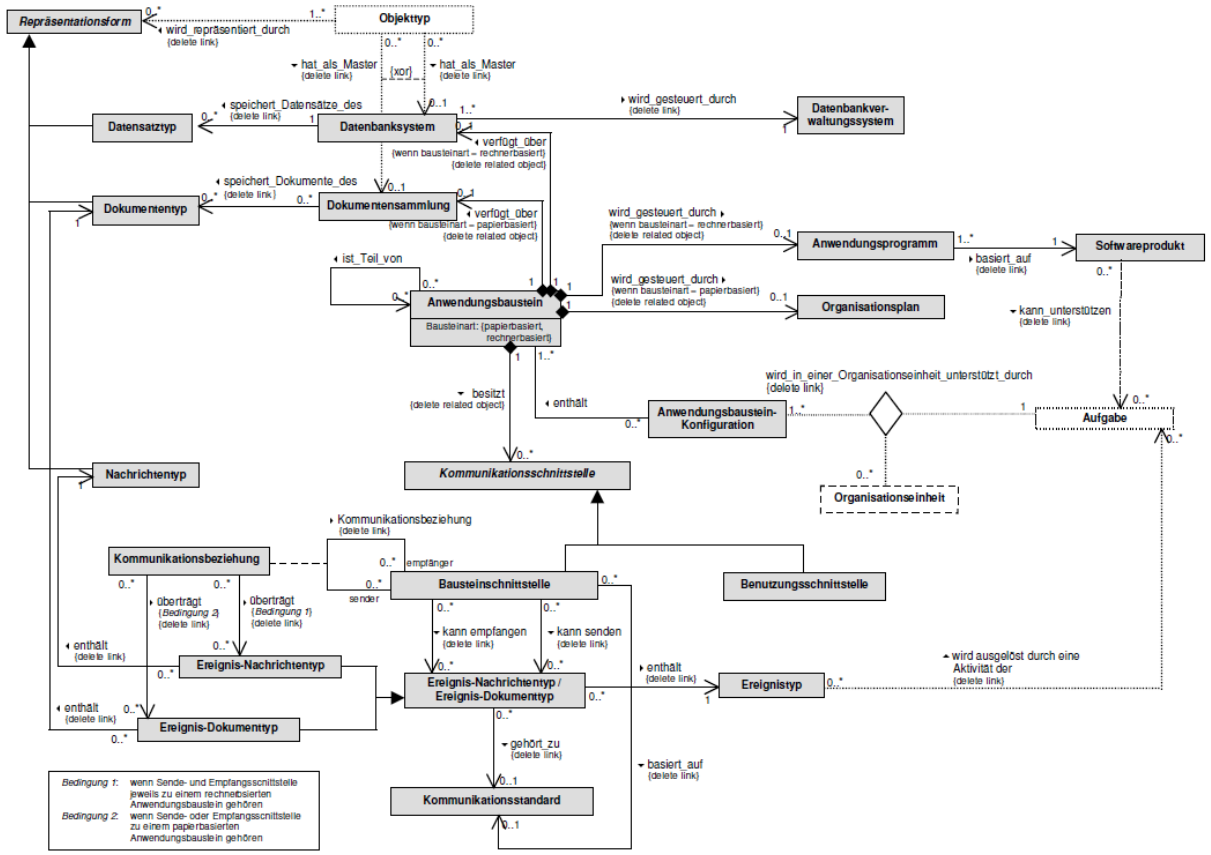

Abbildung 6 – Logische Werkzeugebene des 3LGM² (Quelle: [3LGM²])

2.1.2.2 Zentrale Klassen

Klasse	*Funktion*
Anwendungsbaustein	Funktionale Komponenten eines Informationssystems mit der Fähigkeit der Unterstützung von Aufgaben sowie Verarbeitung und Speicherung von Objekttypen
Ereignis-Nachrichtentypen	Grundbausteine zur Beschreibung einer ereignisgesteuerten Kommunikation, welche sowohl aus Ereignistypen bestehen – die durch Aufgaben ausgelöst werden – als auch Nachrichtentypen, welche die zu kommunizierenden Objekttypen repräsentieren
Kommunikationsbeziehung	Logische Verbindungen zwischen Anwendungsbausteinen zum Austausch von Objekttypen in Form von Ereignis-Nachrichtentypen
Bausteinschnittstelle	Dienen der Verbindung von Anwendungsbausteinen durch Kommunikationsbeziehungen und repräsentieren die darüber austauschbaren Ereignis-Nachrichtentypen
Kommunikationsstandard	Vorgabe für den Aufbau von Ereignis-Nachrichtentypen

2.1.2.3 Beispiel

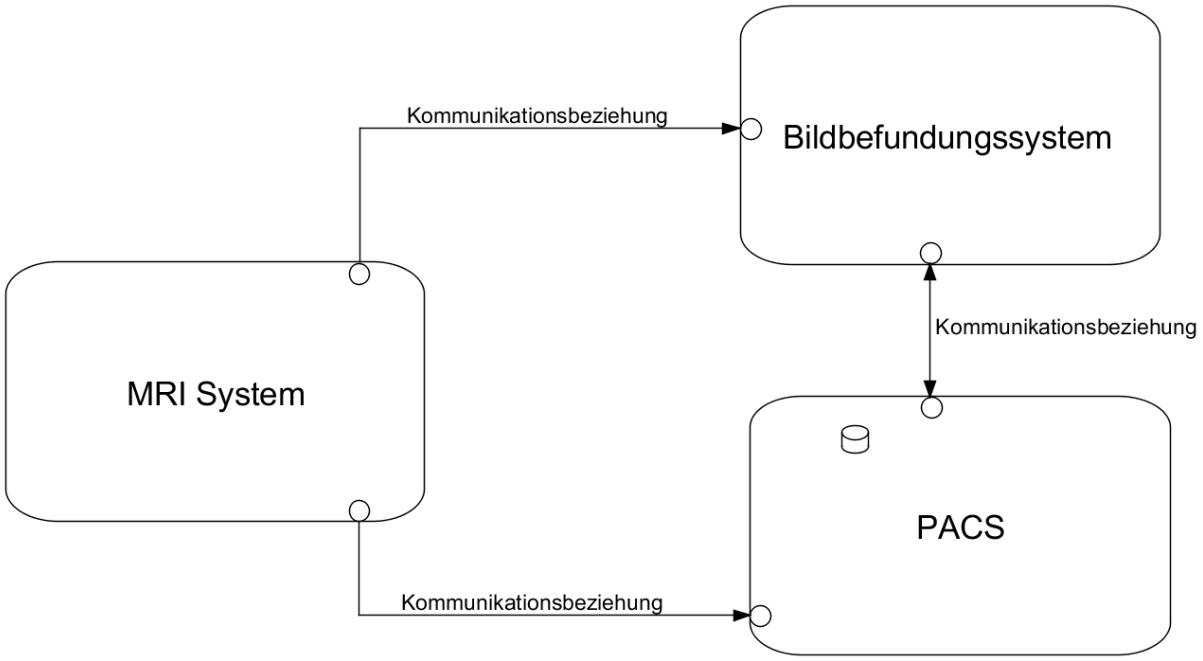

Abbildung 7 – Beispiel der Logischen Werkzeugebene

Das Beispiel zeigt die Logische Werkzeugebene aus Abbildung 3. Zu sehen sind hier essentielle Anwendungsbausteine, wie sie häufig für die Unterstützung der Aufgaben in einer radiologischen Abteilung vorzufinden sind, unter anderem „MRT System“, „Bildbefundungssystem“ und „PACS“. MRT Systeme unterstützen dabei die Erzeugung radiologischer Bilder, Bildbefundungssysteme, die Auswertung der Bilddaten sowie Befundschreibung und das PACS, die Speicherung und Verteilung der Bilder und Befunde. Hierzu ist allerdings eine geeignete Kommunikation erforderlich. Realisiert wird diese über Bausteinschnittstellen (in Abbildung 7 als kleine Kreise auf den Anwendungsbausteinen dargestellt) und Kommunikationsbeziehungen. Bausteinschnittstellen ermöglichen einen Datenaustausch zwischen Anwendungsbausteinen, indem sie das Senden oder Empfangen von Ereignis-Nachrichtentypen erlauben, welche die zu kommunizierenden Objekttypen repräsentieren. Der Datenfluss selbst wird dann durch die Kommunikationsbeziehung dargestellt. So können etwa neu erstellte Bilddaten von einer Modalität zur Befundung bzw. Speicherung an ein Bildbefundungssystem bzw. PACS übermittelt werden. Am Befundungssystem können dann geeignete Bilder selektiert sowie ein entsprechender Befund dazu verfasst und im PACS hinterlegt werden.

Anzumerken ist hier, dass auf dieser Ebene nicht beschrieben wird, wie derartige Kommunikationsbeziehungen konkret umgesetzt werden. Bedeutend ist ausschließlich, dass die Anwendungsbausteine in der Lage sind, Daten auszutauschen. Die technische Realisierung hierzu wird auf der Physischen Werkzeugebene festgehalten.

2.1.3 Die Physische Werkzeugebene

2.1.3.1 Klassendiagramm

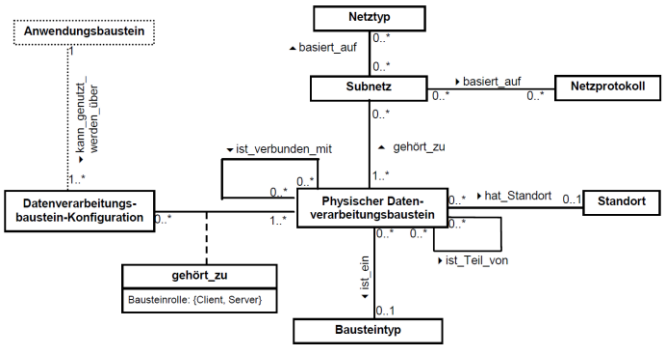

Abbildung 8 – Physische Werkzeugebene des 3LGM² (Quelle: [3LGM²])

2.1.3.2 Zentrale Klassen

Klasse	*Funktion*
Physischer Datenverarbeitungsbaustein	Die physischen Komponenten, die die Informationsverarbeitung auf der Logischen Werkzeugebene realisieren

2.1.3.3 Beispiel

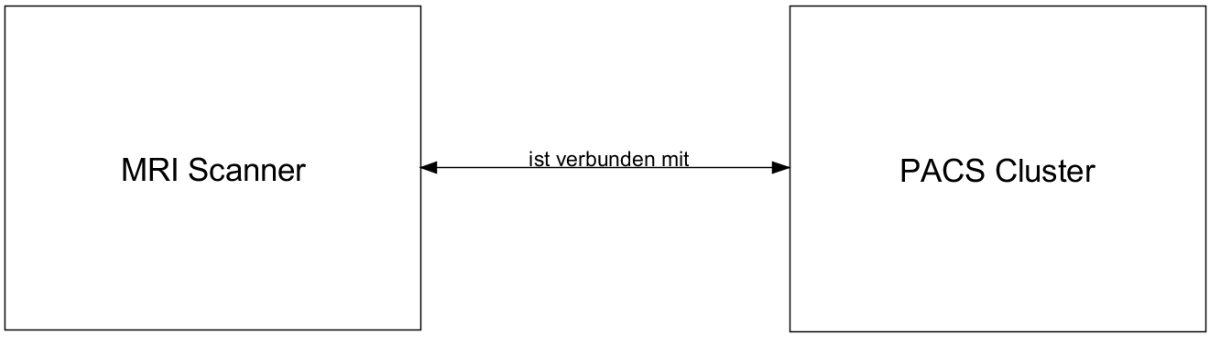

Abbildung 9 – Beispiel der Physischen Werkzeugebene

In diesem Beispiel ist die Physische Werkzeugebene aus Abbildung 3 zu sehen. Dargestellt sind MRI Scanner und Server Cluster, welche häufig anzutreffende, physische Komponenten der Informationsverarbeitung innerhalb einer Funktionsabteilung, hier insbesondere der Radiologie, sind. Der MRI Scanner beispielsweise repräsentiert dabei einen konkreten Magnetresonanztomographen, das Server Cluster einen konkreten Rechnerverbund. Im Gegensatz zur Logischen Werkzeugebene ist es hier nun möglich, eine physische bzw. in diesem speziellen Fall eine hardwaremäßige Kommunikation zu modellieren. Dies geschieht über die „ist_verbunden_mit“-Beziehung, was bedeutet, dass tatsächlich eine Verbindung vorhanden ist, etwa über Ethernet oder auch komplexere Kommunikationsstrukturen.

2.1.4 Interebenen-Beziehungen

Die bisherigen Erläuterungen haben beispielhaft die Verwendung der drei Ebenen des 3LGM² geschildert und die Elemente sowie deren Beziehungen untereinander dargestellt. Allerdings liegen die Ebenen selbst jetzt noch zusammenhangslos im Modell. Wünschenswert wäre eine Möglichkeit zur Verknüpfung von Elementen über die Ebenengrenzen hinaus. Dabei kommen jetzt sogenannte Interebenen-Beziehungen zum Einsatz, wie sie in Abbildung 10 dargestellt sind. Zu sehen ist dort das vorangegangene Beispiel, jetzt allerdings mit folgenden Verknüpfungen zwischen den Ebenen:

Die Linien von der Fachlichen Ebene zur Logischen Werkzeugebene deuten eine Unterstützung einer Aufgabe durch einen Anwendungsbaustein an. So kann beispielsweise ausgedrückt werden, dass das Bildbefundungssystem die Durchführung der Aufgabe Bildbefundung unterstützt. Analog dazu weisen Linien zwischen der Logischen und der Physischen Werkzeugebene darauf hin, dass ein Anwendungsbaustein durch einen Physischen Datenverarbeitungsbaustein realisiert wird, wie es etwa bei PACS und PACS Cluster zu sehen ist.

Außerdem können noch Interebenen-Beziehungen für Objekttypen definiert werden. Für diese ist insbesondere interessant, wo sie kommuniziert und gespeichert werden. In der Abbildung werden diese Zusammenhänge durch die Markierung der Elemente und Pfeile dargestellt. So wird der Objekttyp Radiologisches Bild übertragen durch die Kommunikationsbeziehungen vom MRI System zum Bildbefundungssystem und PACS, sowie beidseitig zwischen Bildbefundungssystem und PACS. Des Weiteren speichert das PACS die Radiologischen Bilder innerhalb des markierten Datenbanksystems.

Auch für Aufgaben existieren derartige Beziehungen: Für sie kann definiert werden, welche Ereignisse sie auslösen. Dieser Sachverhalt wird zwar nicht grafisch repräsentiert, dient aber in Verbindung mit der Kommunikation von Objekttypen dazu, festzulegen, welche Aufgaben die Datenübertragung bedingen.

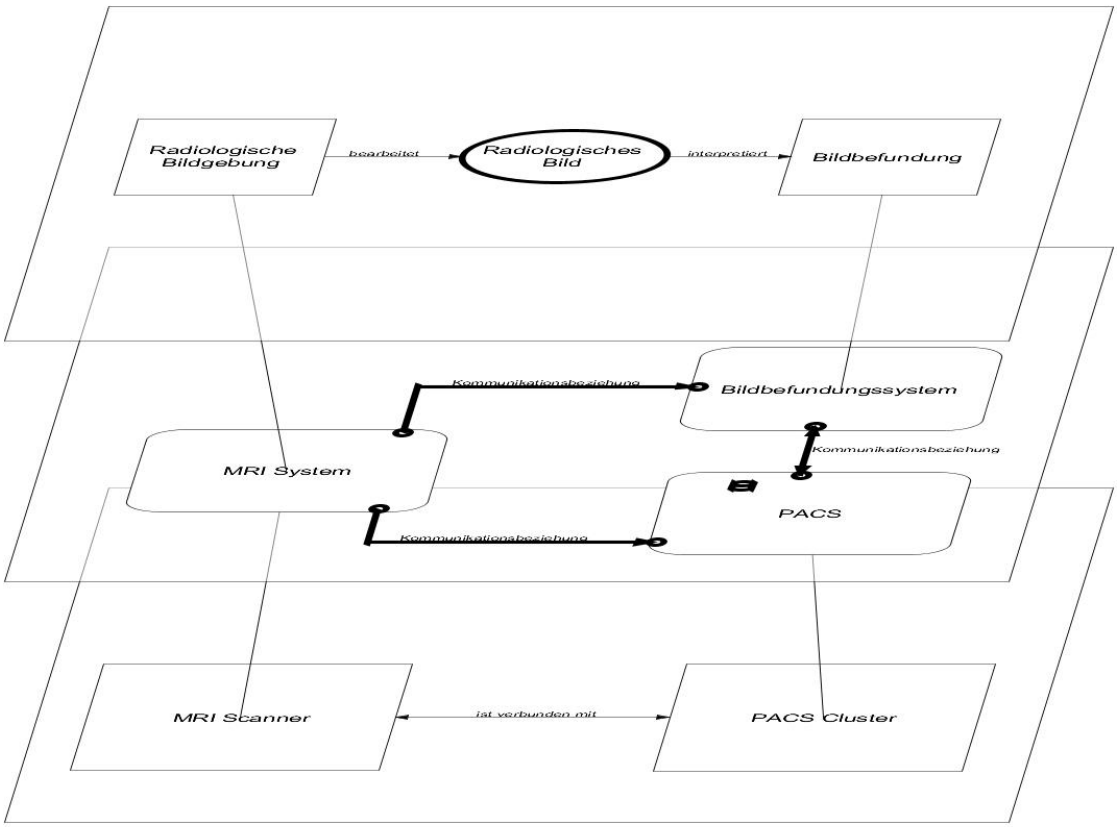

Abbildung 10 – 3LGM²-Modell mit Interebenen-Beziehungen

Hier wird nun die Bedeutung der Aufteilung des Informationssystems in drei Ebenen deutlich:

Die Fachliche Ebene diente vor allem der Beschreibung der zu erledigenden Aufgaben innerhalb einer Einrichtung sowie der durch sie verarbeiteten Informationen. Wesentlich war hier, dass diese Beschreibung unabhängig von der konkreten Umsetzung ist, das heißt, dass die Fragen nach dem „Wie werden die einzelnen Aufgaben erledigt?“ oder „Wie werden die Objekttypen dabei verarbeitet?“ hier nicht betrachtet wurden. Entscheidend war nur, welche Aufgaben wo zu erledigen sind und welche Objekttypen verarbeitet werden. Die Logische Werkzeugebene wurde verwendet für die Modellierung von funktionalen Komponenten, welche zur Verarbeitung, Kommunikation und Speicherung von Informationen dienen. Durch die Inter-Ebenen-Beziehungen zwischen Fachlicher Ebene und Logischer Werkzeugebene ist ein Modellieren der Aufgabenunterstützung möglich, ohne sich mit der Notwendigkeit konfrontiert sehen zu müssen, die konkreten physischen Komponenten, wie etwa Rechner oder auch Modalitäten, festzulegen. Diese wurden stattdessen auf der Physischen Werkzeugebene modelliert. Durch die Interebenen-Beziehungen zwischen Logischer und Physischer Werkzeugebene kann dann unter Kenntnis der benötigten Anwendungsbausteine eine geeignete Auswahl an physischen Komponenten getroffen werden, ohne die konkreten informationsverarbeitenden Prozesse auf der Fachlichen Ebenen kennen zu müssen. Durch eine solche „Aufteilung“ der Modellierung ist es insbesondere möglich, zunächst die Unterstützung eben dieser Prozesse auf einem logischen Level zu planen, sodass den Ansprüchen einer hohen Prozessqualität Genüge getan werden kann, etwa durch eine Konformität der Anwendungsbausteine mit den Erfordernissen der Aufgaben auf der Fachlichen Ebene oder auch durch das Vermeiden von Medienbrüchen. Aufbauend auf der so erzeugten Architektur auf der Logischen Werkzeugebene können dann geeignete Geräte, Computerhardware, personelle Ressourcen etc. auf der Physischen Werkzeugebene allokiert werden, die diese Architektur realisieren. Um allerdings eine effiziente Planung bzw. Modellierung zu gewährleisten, ist eine softwareseitige Umsetzung erforderlich. Ein Beispiel hierfür wird im folgenden Abschnitt vorgestellt.

2.2 Grundfunktionen zum Tool3LGM²

Das Tool3LGM² ist ein Softwarewerkzeug, welches das 3-Ebenen-Metamodell implementiert und dem Nutzer das Erstellen und Analysieren 3LGM²-basierter Modelle erlaubt. Abbildung 11 zeigt

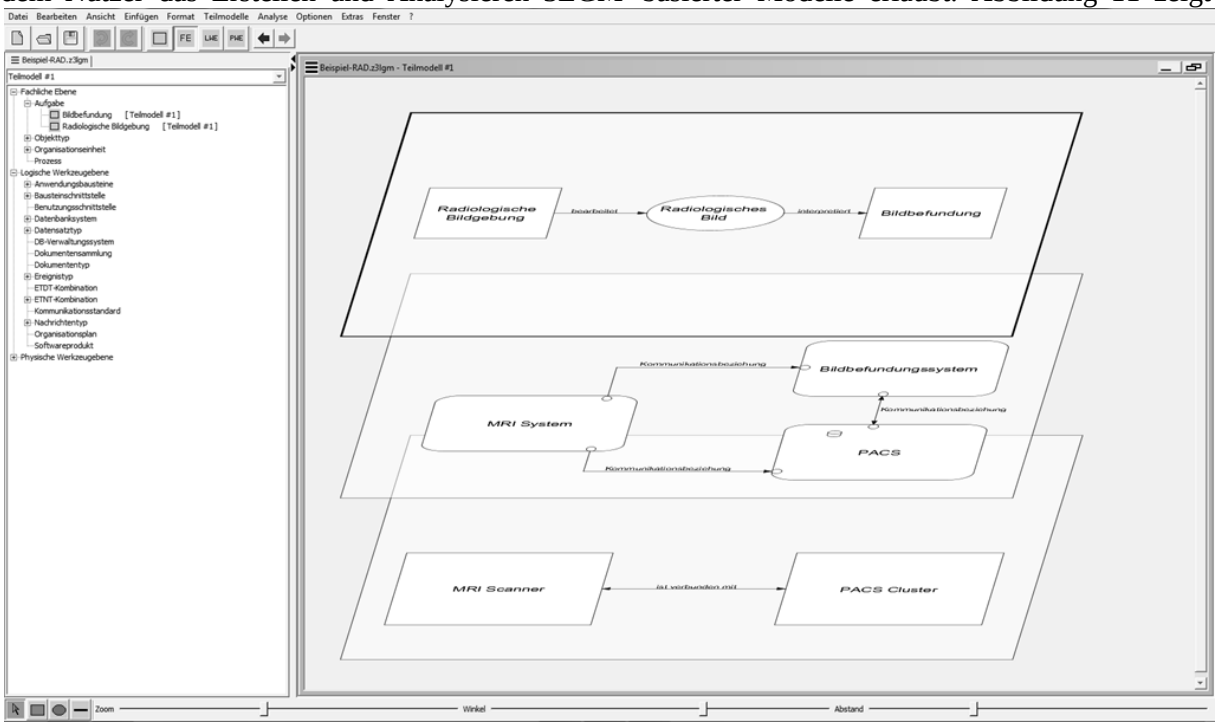

Abbildung 11 – Das Tool3LGM² mit 3-Ebenen-Ansicht des Beispiels aus 2.1

hierzu die grafische Oberfläche des Modellierungswerkzeuges, welche sich in vier Hauptbereiche untergliedert:

- Im oberen Teil des Fensters sind Menü- und Werkzeugleiste angeordnet, welche einen Großteil der Tool3LGM²-Funktionalität zugänglich machen, wie etwa das Erstellen, Laden und Speichern von Modellen, das Anlegen und Formatieren von Elementen sowie das Einstellen allgemeiner Optionen und das Zugreifen auf diverse Spezialfunktionen.
- Der linke Bereich beinhaltet den sogenannten Modell Browser, welcher in Form eines Baumes die Elemente im aktuellen Modell, sowie deren Zugehörigkeit zu den Klassen des 3LGM² darstellt. Dieser dient vor allem der erleichterten Navigation im Modell und kann ebenfalls zum Anlegen neuer Elemente benutzt werden.
- Der rechte Teil des Fensters dient der Darstellung verschiedener Sichten auf ein 3LGM²-Modell in grafischer Weise. Im Tool3LGM² kann jedes Modell in verschiedene Teilmodelle zerlegt werden, wobei jedes eine solche Sicht repräsentiert und mit einer entsprechenden grafischen Darstellung assoziiert ist. Diese grafischen Ansichten dienen vor allem der Übersichtlichkeit, können aber auch wiederum zum Anlegen und Editieren von Elementen verwendet werden.
- Im unteren Bereich sind abschließend noch verschiedene Mauswerkzeuge zu finden, wie etwa für das Anlegen neuer Elemente und Beziehungen, sowie Einstellmöglichkeiten für die grafische Darstellung im rechten Bereich.

Neben diesen Funktionen existieren noch eine Reihe weiterer Möglichkeiten für das Arbeiten mit Modellen. Basismäßig verfügt jedes Modell, jedes Teilmodell sowie jedes Element über einen Namen und eine Beschreibung, wodurch das Zuweisen sowohl einer Bezeichnung als auch einer detaillierten Erläuterung möglich ist. Des Weiteren wird für jedes Element eine Auswahl häufig benötigter Beziehungen angeboten, wie etwa die unterstützten Aufgaben eines Anwendungsbausteins (Abbildung 12).

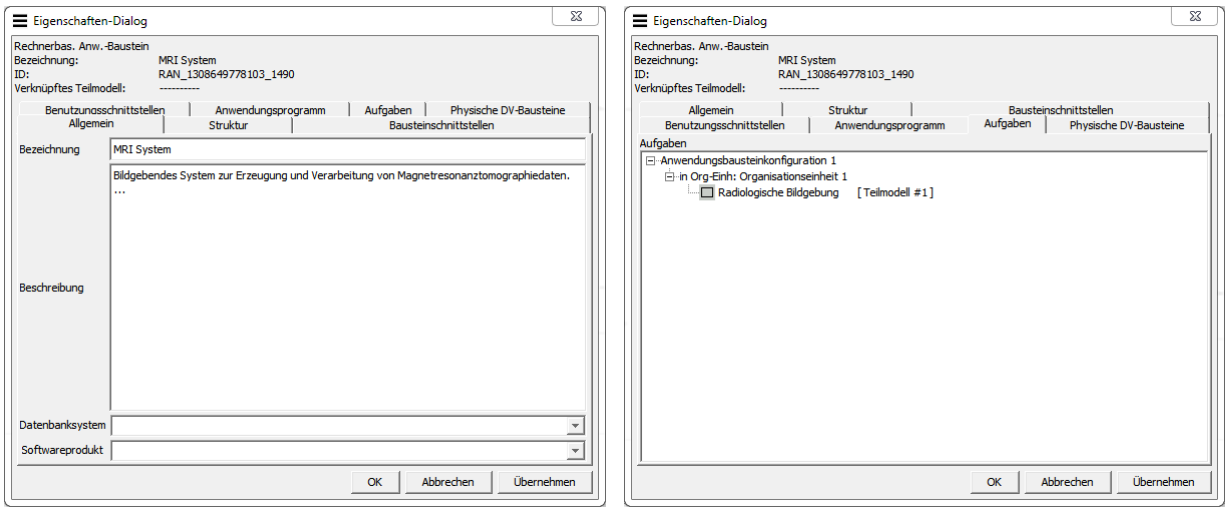

Abbildung 12 – Eigenschaftendialog zum Anwendungsbaustein MRI System

Das 3LGM² sieht in diesem Zusammenhang Beschränkungen vor, wie häufig derartige Beziehungen zwischen Elementen existieren dürfen. Beispielsweise sollte eine Bausteinschnittstelle höchstens einem Kommunikationsstandard angehören. Wird dies verletzt, bietet das Tool3LGM² eine sogenannte Konsistenzprüfung an, welche dem Nutzer die Abweichungen vom Metamodell anzeigt und ihm das Beheben dieser Probleme ermöglicht. Abbildung 13 zeigt hierzu das Tool mit aktivierter Konsistenzprüfung im unteren Fensterbereich. Als Modell wurde dabei das im Tool3LGM² verfügbare „Beispielmodell“ verwendet, für das nun derartige Verletzungen in Tabellenform aufgelistet werden.

Neben diesen Möglichkeiten existieren noch zahlreiche weitere Funktionen, wie etwa das Definieren benutzerdefinierter Eigenschaften sowie das Durchführen von Analysen. Das Tool3LGM² erweist sich damit, durch die Implementierung des 3LGM² sowie dem Anbieten darüber hinaus gehender Funktionalität, als geeignetes Mittel für die Modellierung von Informationssystemen.

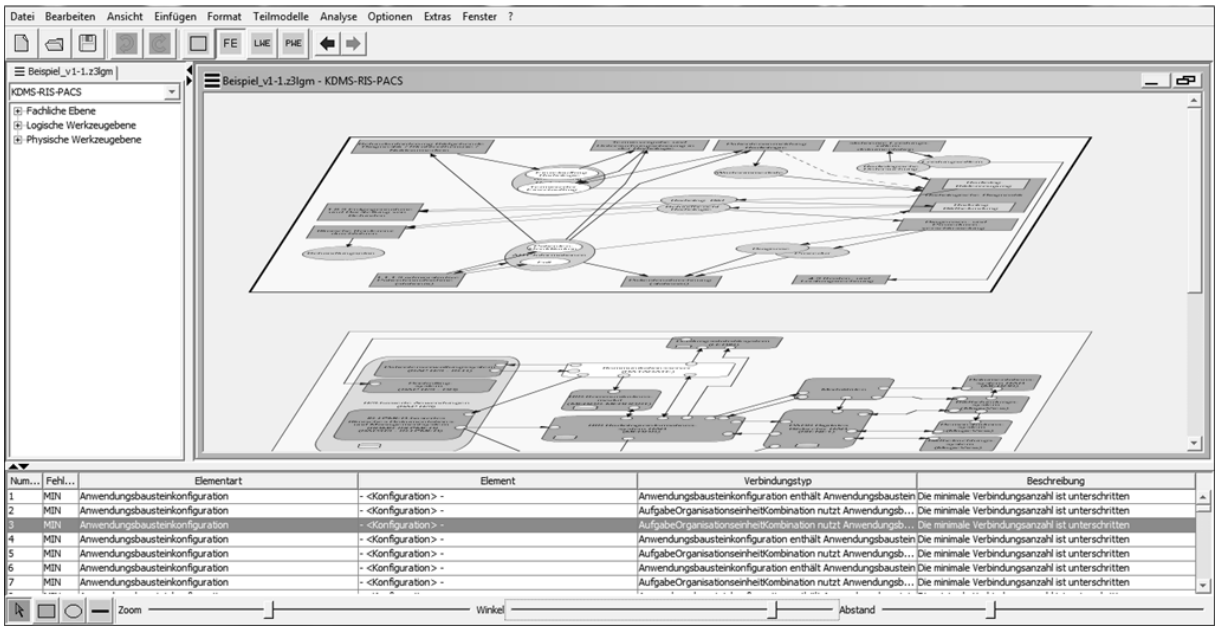

Abbildung 13 – Tool3LGM² mit aktivierter Konsistenzprüfung (unten)

Problematisch ist allerdings die Tatsache, dass ein Modellierer hierbei keine Entscheidungsunterstützung erfährt. Betrachtet man etwa das Beispiel aus Abbildung 5 – mit einer gegebenen Fachlichen Ebene – und stünde nun vor der Herausforderung, eine geeignete Logische Werkzeugebene zu entwickeln, wäre es wünschenswert zu wissen, dass etwa eine Kombination aus MRI System, Bildbefundungssystem und PACS inklusive der erforderlichen Kommunikationsbeziehungen eine geeignete Möglichkeit hierfür darstellt. Des Weiteren wären Angaben über den zu verwendenden Kommunikationsstandard hilfreich, in diesem Fall etwa DICOM. Derartiges Wissen ist allerdings höchstens indirekt, das heißt in Form von Literatur, verfügbar. Modellierer müssen es deshalb zunächst finden, erarbeiten und dann in die Modellierungssprache „übersetzen". Insbesondere bei gebräuchlichen und bewährten Architekturen stellt dies einen nichtvertretbaren Mehraufwand dar. Ginge man hingegen davon aus, dass derartiges Wissen direkt bei der Modellierung mit dem Tool3LGM² abrufbar wäre, könnte vor allem der zu inverstierende Zeitbedarf für das Erarbeiten und Übersetzen deutlich reduziert werden. Pauschal müsste man daher nur einmal das Wissen in geeigneter Weise formalisieren und als Vorlage zur Verfügung stellen. Diese Vorlagen können dann unterstützend im Tool3LGM² eingesetzt werden, was die Arbeit des Modellierers erheblich erleichtert. Insbesondere könnten dabei auch detaillierte Beschreibungen bereitgestellt werden, welche ein tieferes Verständnis für das so formalisierte Wissen schaffen. So wäre es dann etwa möglich, um zum obigen Beispiel zurückzukehren, Erläuterungen zur Implementierung für den Kommunikationsstandard DICOM zu hinterlegen. Dies hilft nicht nur dem Modellierer, sondern auch denjenigen, die die geplante Architektur des Informationssystems umsetzen.

Es soll nun primäres Ziel sein, eine derartige Unterstützung für das Tool3LGM² zu konzipieren. Konkret wird darauf in Kapitel 4 eingegangen. Zunächst werden allerdings noch einige dabei benötigte Begrifflichkeiten und Definitionen geklärt und erläutert.

3 Modelle – Patterns – Frameworks

Das folgende Kapitel beschäftigt sich mit der Interpretation und der Abgrenzung der Begriffe Modell, Pattern und Framework hinsichtlich der Nutzbarmachung von Entwurfswissen für die Modellierung.

3.1 Modell

Für den Modellbegriff gibt es eine Vielfalt an Definitionen, die entsprechend der jeweiligen Intention unterschiedliche Aspekte adressieren. So schlägt Schütte [SCHÜTTE 1998] beispielsweise vor:

„Ein Modell ist das Ergebnis einer Konstruktion eines Modellierers, der für Modellnutzer Elemente eines Systems zu einer Zeit als relevant mit Hilfe einer Sprache deklariert".

Problematisch bei dieser Definition im Hinblick auf die Nutzbarmachung von Entwurfswissen ist allerdings das Fokussieren auf das Modell als existierendes Ergebnis der Modellierung. Besser geeignet ist der Modellbegriff nach Brocke [BROCKE 2003], welcher zusätzlich den Konstruktionsprozess des Modelles selbst in den Blickpunkt rückt:

„Ein Modell ist die Verdichtung von Wahrnehmung zu Inhalten eines Gegenstands, um auf diese Weise einem spezifischen Zweck zu dienen. Die Gestaltung von Modellen erfolgt in Konstruktionsprozessen."

Ein Konstruktionsprozess ist eine inhaltlich abgeschlossene, zeitliche und sachlogische Folge von Funktionen, die zur Bereitstellung eines Modelles in einem spezifizierten Endzustand notwendig sind. Das Modell erfährt dabei innerhalb der Funktionsfolge selbst eine wesensgestaltende Zustandsveränderung.

Dieser Modellbegriff erlaubt es nun, Entwurfswissen eben als eine Folge von Funktionen aufzufassen, welche innerhalb des Konstruktionsprozesses zur Erstellung eines Modelles im gewünschten Endzustand beiträgt. Auf diese Weise ist die Basis geschaffen für das Zusammenführen von Entwurfswissen und Modellierung. Zur vollständigen Klärung der genauen Anwendung dieses Entwurfswissens soll im Folgenden näher auf den Begriff des „Patterns" eingegangen werden.

3.2 Pattern

Der Begriff Pattern ist vor allem im Bereich der Softwareentwicklung etabliert, wo er als Lösungsvorlage für bekannte und wiederkehrende Probleme verstanden wird. Ein bekanntes Beispiel ist dabei das Model-View-Controller Pattern, welches eine Trennung von Daten, Präsentation und Steuerung bei Entwicklung grafischer Anwendungen vorschlägt, um eine geeignete Programmstruktur insbesondere hinsichtlich der Wiederverwendbarkeit der Programmbestandteile zu gewährleisten. Das MVC-Pattern beinhaltet dabei Erfahrungswissen und Erkenntnisse, die aus Beobachtungen heraus als geeignet für die Strukturierung derartiger Programme identifiziert wurden und dient damit als Vorlage und Lösungsmuster.

Malich formulierte den Begriff der Patterns allgemein:

„Das Konzept der Patterns umfasst die systematische Dokumentation von Entwurfswissen, das durch die Analyse von existierenden und erprobten Entwürfen gewonnen wird." [MALICH 2007]

Derartiges Entwurfswissen soll nun, wie in 3.1 bereits angedeutet, in den Konstruktionsprozess von Modellen einfließen. Im Unterschied zur Softwareentwicklung ist es hier allerdings nicht das Ziel, den Einsatz von Programmiersprachen zu beschreiben, sondern den von Modellierungssprachen. Ein entscheidender Punkt ist nun, in welcher Form das Dokumentieren solcher Patterns erfolgen soll, damit diese sinnvoll in den Konstruktionsprozess von Modellen einbezogen werden können. Im Rahmen dieser Arbeit wird dazu eine Möglichkeit vorgestellt, diese Patterns bzw. das Entwurfswissen wiederum selbst als Modelle zu formalisieren. Die Wahl dieser Darstellungsform liegt insbesondere

im Hinblick auf die Zielsetzung darin begründet, dass Modelle eine geeignete Möglichkeit bieten, die im Entwurfswissen enthaltenen Konzepte in grafisch übersichtlicher Weise abzubilden und somit den Ansprüchen der Deskriptivität genügen. Die ebenso gewünschten konstruktiven Eigenschaften werden in den folgenden Abschnitten behandelt.

3.3 Konstruktionspattern für Informationssysteme

Zunächst stellt sich die Frage, ob und wie Entwurfswissen als Modell formalisiert werden kann und wie ein effizientes Involvieren derartiger Modelle in den Konstruktionsprozess von Informationssystemmodellen möglich ist:

Betrachtet man die Modelldefinition von Schütte (3.1), so kann unter der Annahme einer existierenden Beschreibungssprache ein Pattern als Ergebnis der Konstruktion eines Modellierers aufgefasst werden, der Entwurfswissen in eben diesem Pattern formalisiert. Der Modellnutzer – also in diesem Fall der Patternnutzer – ist allerdings wiederum selbst Modellierer. Möchte man nun diese Definition erneut verwenden für das Erstellen eines Informationssystemmodelles, ist es nicht möglich, das unterstützende Wesen des Patterns herauszustellen. Im Sinne von Brockes Modellbegriff kann nun allerdings einen Schritt weiter gegangen und dabei ein Pattern als Gegenstand einer Funktion aufgefasst werden innerhalb der Funktionsfolge des Konstruktionsprozesses eines solchen Modelles. Patterns sind dann also zum einen Ergebnisse einer Modellierung, unterstützen darüber hinaus aber wiederum selbst die Modellkonstruktion. Genau dieser Punkt macht deutlich, wie hier der Patternbegriff zu verstehen ist.

Nichtsdestotrotz ist es hier nicht das Ziel, eine einheitliche Interpretation dieses Begriffes zu postulieren. Vielmehr soll nun für das eben beschriebene Verständnis eines Patterns die Bezeichnung „Konstruktionspattern“ verwendet werden, um damit eine für die Zwecke der Konstruktion von Informationssystemmodellen dienliche Einordnung zu finden.

Definition (Konstruktionspattern):

*Ein **Konstruktionspattern** ist ein Modell, welches durch systematische Dokumentation von Entwurfswissen aus Analysen von existierenden und erprobten Entwürfen gewonnen wird, mit dem Ziel der Präsentation und Bereitstellung dieses Wissens sowie der konstruktiven Unterstützung des Modellierers bei dem Prozess der Erstellung von Informationssystemmodellen.*

Über diese Definition ist nun zum einen die Betrachtung des Konstruktionspatterns selbst als Modell geschaffen und zum anderen die Intention der inhärenten Fähigkeit zur Unterstützung der Erstellung von Informationssystemmodellen gegeben. Um erneut zur eingangs aufgeführten Fragestellung zurückzukehren, verbleibt es nun noch zu klären, wie Entwurfswissen in Konstruktionspatterns formalisiert werden kann und wie diese auf effiziente Weise in den Konstruktionsprozess von Informationssystemmodellen einbezogen werden können. Für die Nutzbarmachung von Konstruktionspatterns wird hierzu die Existenz einer Funktion gefordert, welche ein Konstruktionspattern auf die Modellierungssprache bzw. auf das Metamodell der Informationssystemmodelle abbildet.

Definition (Interpreterfunktion):

Das Modell $\mathcal{P}$ ist dann ein Konstruktionspattern für Modelle zu einem Metamodell $\mathcal{M}$, wenn eine Abbildung

$$\mathrm{i}: \mathcal{P} \longrightarrow \mathcal{M}$$

*existiert, welche jedem Element bzw. jeder Beziehung in $\mathcal{P}$ ein Element bzw. eine Beziehung in $\mathcal{M}$ zuweist. Sie wird im Folgenden als „**Interpreterfunktion**“ bezeichnet.*

Durch die Forderung nach einer solchen Funktion wird die Grundlage dafür geschaffen, dass das Konstruktionspattern tatsächlich auch für die Konstruktion von Modellen zum gegebenen Metamodell geeignet ist und somit für die Unterstützung der Modellierung zum Einsatz kommen kann. Insbesondere induziert die Definition der Interpreterfunktion auf Metaebene die Verwendbarkeit jedes Elements und jeder Beziehung des Konstruktionspatterns als ein Template für Instanzen zu einem

Element bzw. einer Beziehung im Metamodell. Hieran wird deutlich, dass zunächst noch eine wesentliche (bisher implizit vorausgesetzte) Vorannahme zu treffen ist, nämlich die der Instanziierbarkeit der Elemente und Beziehungen des Metamodelles. Dies liegt vor allem darin begründet, dass eine Unterstützung des Konstruktionsprozesses von Informationssystemmodellen offensichtlich nur dann sinnvoll ist, wenn auch Instanzen des Metamodelles erzeugt werden können. Im Sinne der Unified Modeling Language würde dies etwa bedeuten, dass alle Elemente und Beziehungen des Metamodelles wiederum Instanzen des Uml::Core::Abstractions::Classifiers::Classifier sein müssen, wie es vor allem beim UML Klassendiagramm der Fall ist (siehe „9.4 Classifiers Package“ in [UML IS]). Unter dieser Voraussetzung können dann die Elemente eines Konstruktionspatterns dafür eingesetzt werden, als Vorlage für Instanzen zu Elementen und Beziehungen des Metamodelles zu dienen, indem etwa Werte für bestimmte Attribute vordefiniert oder auch Einschränkungen bezüglich der Beziehungen dieser Instanzen untereinander eingeführt werden. Hierbei wird nun deutlich, dass auch für die Modellierung der Konstruktionspatterns selbst eine Modellierungssprache zum Einsatz kommen muss, die kompatibel mit dem Metamodell der Informationssystemmodelle ist, was nun zur verbleibenden Fragestellung führt: Wie sollen Konstruktionspatterns das Entwurfswissen formalisieren? Dieser Punkt stellt die größte Herausforderung dar, da sie auf der einen Seite Entwurfswissen aus einer potentiell beliebigen Domäne wiedergeben sollen, zugleich aber auch auf die Domäne der Informationssystemmodelle abbildbar sein müssen. Demnach ist es erforderlich, Kenntnisse über die hinter dem Metamodell stehenden Konzepte zu besitzen, um eine geeignete Modellierungssprache bzw. ein geeignetes Framework für die Definition der Konstruktionspatterns ableiten zu können.

3.4 Framework

Unter der Kenntnis der Konzepte eines Metamodelles gilt es nun, ein Framework zu erzeugen, welches der Beschreibung von Konstruktionspatterns dient und zudem die Abbildung derer auf das Metamodell – sprich die Interpreterfunktion – bereitstellt. Zuerst ist allerdings anzumerken, dass auch hier analog zum Begriff der Patterns verschiedene Interpretationsweisen des Framework-Begriffs bestehen. Das häufig anzutreffende Verständnis ist dabei das des Rahmenkonzeptes im Bereich der Softwareentwicklung, bei dem aufbauend auf einer Programmiersprache Vorgaben für das Lösen einer Klasse von Problemen bereitgestellt werden, welche auf abstrakter Ebene definiert und dann vom Programmierer konkretisiert werden. Frameworks in Bezug auf Konstruktionspatterns sind mit der Auffassung von Frameworks der Softwareentwicklung insoweit verwandt, dass auch sie einen Rahmen für das Lösen bestimmter Probleme vorgeben, sich aber im Gegensatz dazu nicht auf Programmiersprachen, sondern auf Modellierungssprachen beziehen. Die hier relevante Klasse der zu lösenden Probleme ist dabei die der Formalisierung von Entwurfswissen und dessen intendierte Interpretation im Metamodell der Informationssystemmodelle. Jedes Konstruktionspattern ist dann eine Instanz dieses Frameworks und umfasst die Lösung für das Formalisieren und Interpretieren einer konkreten Menge an Entwurfswissen in eben diesem Metamodell. Im folgenden Kapitel wird hierzu ein entsprechendes Framework für das 3LGM² vorgestellt.

4 Das 3LGM² – Construction Framework

Das 3LGM² - Construction Framework, welches im Folgenden vorgestellt werden soll, ist ein Framework im Sinne von 3.4, mit der Konkretisierung des 3LGM² als Metamodell. Dieses wurde insbesondere deshalb gewählt, da es geeignete Konzepte für die Modellierung von Informationssystemen im Gesundheitswesen bereitstellt und zugleich eine Implementierung durch das Tool3LGM² existiert, welches als Basis für eine spätere softwareseitige Umsetzung des Frameworks in Frage kommen könnte. Das 3LGM² - Construction Framework (kurz: 3LGM²-CF) erlaubt das Formalisieren von Entwurfswissen in Konstruktionspatterns, welche im 3LGM² interpretiert werden können. Im folgenden Abschnitt wird dazu zunächst auf die allgemeinen Konzepte eingegangen und im Anschluss daran die technische Spezifikation gegeben.

4.1 Grundlagen

Abbildung 14 zeigt eine Einordnung von 3LGM² -CF und Konstruktionspatterns in den Kontext von UML und 3LGM².

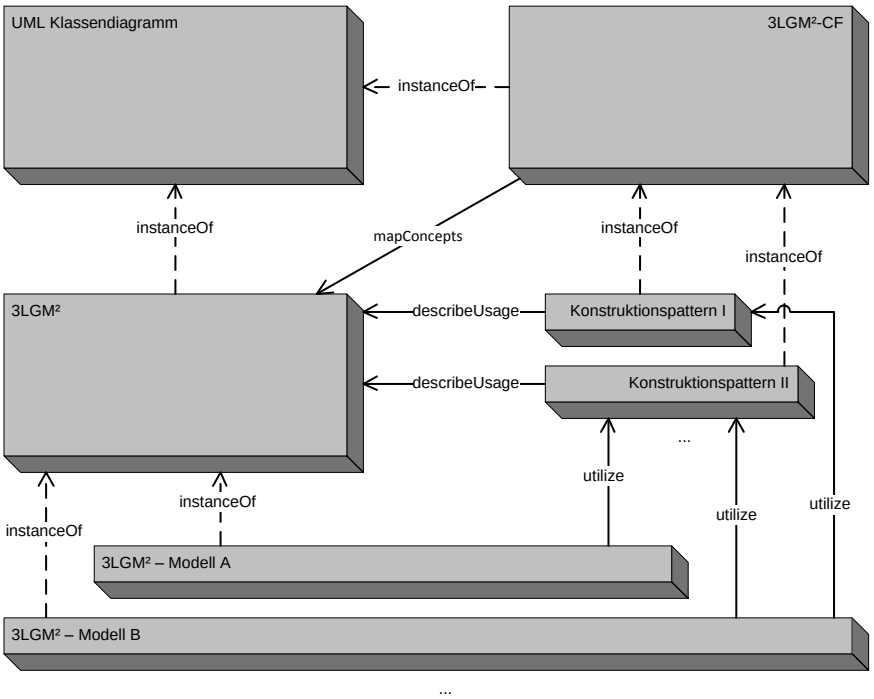

Abbildung 14 – Zusammenhänge zwischen 3LGM² und 3LGM² - Construction Framework

Sowohl 3LGM² als auch 3LGM²-CF sind in Form von UML Klassendiagrammen definiert. Entsprechend sind 3LGM²-Modelle wiederum durch das 3LGM² definiert und 3LGM²-Konstruktionspatterns wiederum durch das 3LGM²-CF. Dabei formalisiert das 3LGM² die allgemeinen Konzepte für das Modellieren von Informationssystemen, wo hingegen die Konstruktionspatterns bereits Konkretisierungen dieser allgemeinen Konzepte entsprechend dem zugrundeliegenden Entwurfswissen enthalten. Um die Verbindung zwischen Entwurfswissen und Informationssystemmodellen zu schaffen, bedient sich das 3LGM²-CF den Konzepten im 3LGM² und beschreibt abstrakt, wie die Konstruktionspatterns zu interpretieren sind. Dazu erlaubt es die Definition sogenannter PatternNodes und PatternRelations, welche als Vorlage für Elemente einer bestimmten Klasse bzw. Assoziation des 3LGM² dienen (Abbildung 15). PatternNodes repräsentieren dabei im Allgemeinen die im Entwurfswissen vorhandenen Entitäten und ermöglichen das Darstellen dieses Wissens über Attribute und Beziehungen. Dazu erhalten sie die Fähigkeit, etwa Beschreibungstexte anzulegen sowie Verbindungen – die PatternRelations – mit anderen PatternNodes einzugehen.

Zusammengefasst werden die PatterNodes und PatternRelations durch den Oberbegriff bzw. die Oberklasse PatternObject, welche deren gemeinsame Eigenschaften beinhaltet. Für die korrekte Abbildung im 3LGM² erhält dabei jedes PatternObject genau eine Zielklassifizierung („targetClassifier"), was bedeutet, dass alle von ihm abgeleiteten Elemente eines 3LGM²-Modelles Instanzen dieser Klassifizierung sein müssen. Ein PatternNode, wie beispielweise die „Vorlage PACS", wird dann auf die Elementklasse „Rechnerbasierter Anwendungsbaustein" abgebildet und dient damit als Vorlage für konkrete Instanzen eines „PACS" in 3LGM²-Modellen.

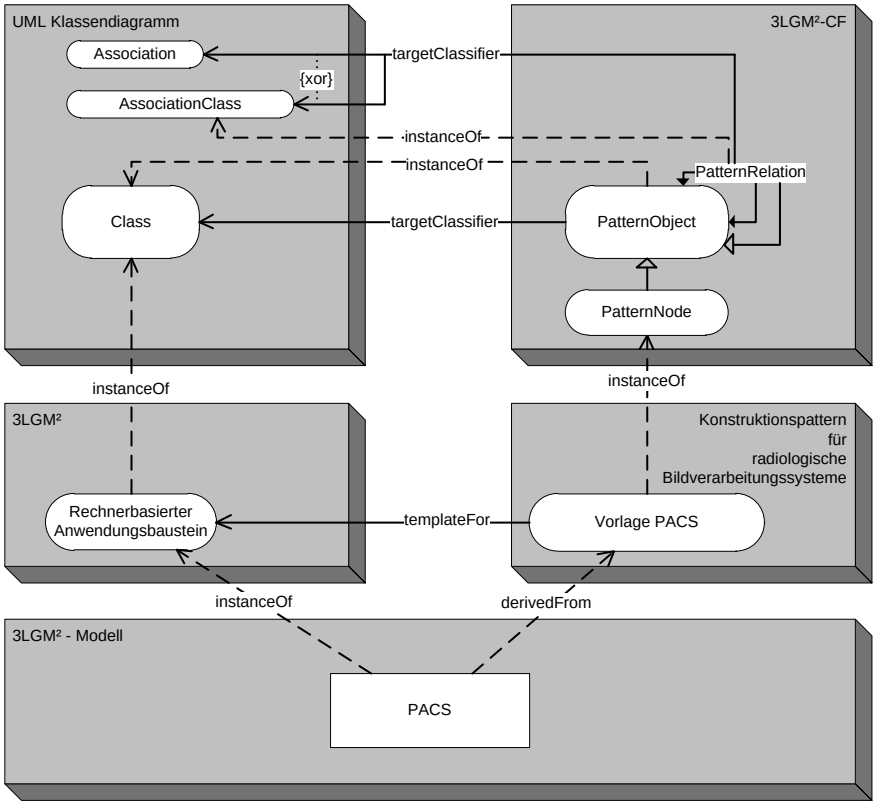

Abbildung 15 – Erweiterung zu den allgemeinen Zusammenhängen aus Abbildung 14 mit Klassen, Vorlagen & Instanzen

Des Weiteren ist es möglich, Vorlagen für Assoziationen oder Assoziationsklassen zu definieren, was über die bereits erwähnten PatternRelations gewährleistet wird. Diese werden in der jeweiligen Form im 3LGM² abgebildet und dienen so als Vorlage für die Beziehungen der Elemente in einem 3LGM²-Modell. Abbildung 16 zeigt hierzu ein Beispiel für eine DICOM-basierte Kommunikation von PACS und Modalität. Das dazugehörige Konstruktionspattern repräsentiert dabei zwei wesentliche Aussagen:

1. Der formalisierte Sachverhalt stellt bewährtes Wissen dar. Hier insbesondere ist DICOM ein geeigneter Kommunikationsstandard für den Austausch radiologischer Bilddaten.
2. Das so formalisierte Wissen lässt sich im 3LGM² abbilden:
    a. Die PatterNodes „Vorlage Modalität-System" und „Vorlage PACS" werden auf die 3LGM²-Klasse „Rechnerbasierter Anwendungsbaustein" abgebildet und dienen damit als Vorlage für Instanzen dieser Klasse innerhalb beliebig vieler 3LGM²-Modelle.
    b. Die PatternRelation „Vorlage DICOM Kommunikation" wird auf die Assoziationsklasse „Kommunikationsbeziehung" im 3LGM² abgebildet, deren Instanzen in 3LGM²-Modellen ausschließlich Rechnerbasierte Anwendungsbausteine verbinden können, die von den beiden PatternNodes abgeleitet worden sind.

Insbesondere an Punkt 2.b wird hierbei deutlich, wie Konstruktionspatterns dem Modellierer helfen, wohlstrukturierte Informationssysteme zu konstruieren, da er durch die strukturellen Restriktionen bei der Modellierung entsprechend dem enthaltenen Entwurfswissen hin zu einer adäquaten Architektur des Informationssystems gelenkt wird. Darüber hinaus können etwa auch Beschreibungstexte oder

Anwendungshinweise sowohl für die PatternNodes, die PatternRelations als auch für das Konstruktionspattern selbst angefügt werden. Dies eröffnet dem Patternmodellierer die Möglichkeit, umfangreiche Informationen im Pattern zu hinterlegen, welche dann wiederum den Modelliereren der Informationssysteme Aufschluss über Funktionsweise und Anwendung der PatternObjects und des Patterns selbst geben. Um zum Beispiel der Kommunikation der radiologischen Systeme zurückzukehren, ließen sich dort etwa bei der PatternRelation „Vorlage DICOM Kommunikation“ wichtige Angaben über Nachrichtenformate erfassen und darüber hinaus auch Hinweise für die Anwendung dieser PatternRelation innerhalb von 3LGM²-Modellen geben. Bei der Erstellung von Informationssystemmodellen profitiert man so durch einen besseren Einblick in das zugrundeliegende Entwurfswissen und erhält zugleich die notwendigen Hintergründe über die Verwendungsweise im 3LGM². Wie etwa beim eben aufgeführten Beispiel, kann dort die DICOM Kommunikation, welche als bewährtes Konzept innerhalb des Konstruktionspatterns erfasst wurde, in das 3LGM²-Modell eines Informationssystems übertragen werden.

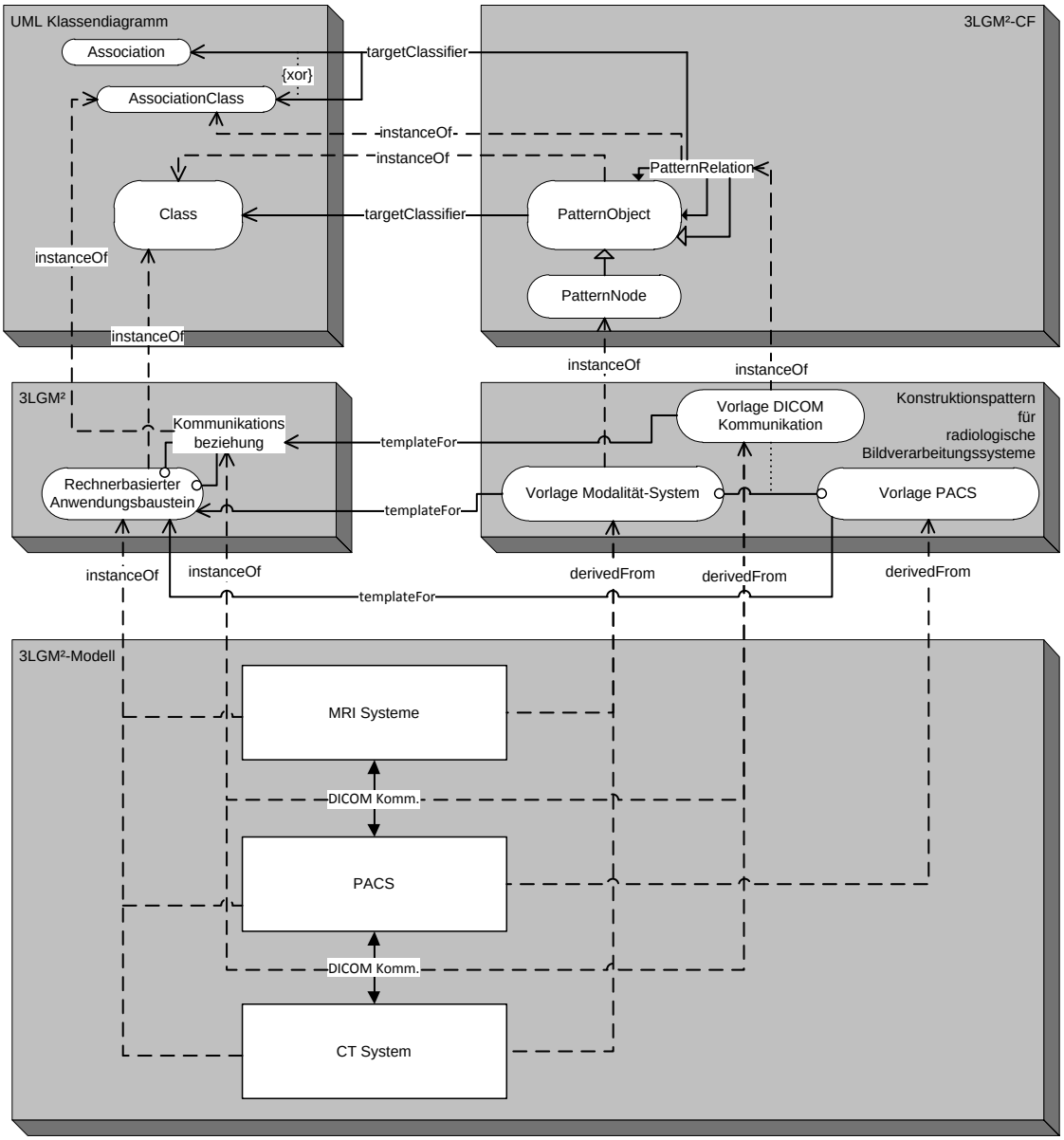

Abbildung 16 – Erweiterung zu Abbildung 15 mit Beziehungen in Pattern, Modell und Metamodell; *Zusammenhänge im 3LGM² sowie im Konstruktionspattern sind aus Gründen der Übersichtlichkeit nur schematisch dargestellt (vgl. 2.1.2))*

Allgemein ergibt sich durch die Strukturierung von Entwurfswissen in Konstruktionspatterns ein weiterer wesentlicher Vorteil: die optimale Wiederverwendbarkeit bei der Modellierung. Wissen über bestimmte Zusammenhänge und Sachverhalte als Konstruktionspattern formalisiert, kann beliebig oft wiederverwendet werden und ermöglicht so nicht nur die Entwicklung besserer Informationssysteme, sondern darüber hinaus auch eine deutliche Beschleunigung des Modellierungsprozesses. Dies liegt insbesondere darin begründet, dass ein Modellierer bei der Erstellung von Informationssystemmodellen zum einen nicht mehr darauf angewiesen ist, das notwendige Wissen selbst zu erarbeiten und zum anderen sich auch nicht mehr mit der Notwendigkeit konfrontiert sieht, die Übertragung dieses Wissens in die Modellierungssprache vornehmen zu müssen.

Damit allerdings die Anwendung derartiger Konstruktionsmuster gelingen kann und diese darüber hinaus die notwendige Effizienz besitzt, sind folgende Grundlagen bzw. Grundannahmen zu schaffen:

1. Das Metamodell für die Modellierung der Informationssysteme muss softwareseitig implementiert sein.
2. Die Formalisierung von Entwurfswissen muss computerunterstützt ablaufen.
3. Die Software für die Modellierung muss das formalisierte Entwurfswissen interpretieren und dem Nutzer zugänglich machen können.

Da im Rahmen dieser Arbeit die Zielsetzung gegeben ist, Entwurfswissen für das 3LGM² bzw. für 3LGM²-Modelle nutzbar zu machen, kann Punkt 1 durch das Tool3LGM² als gegeben angenommen werden. Zu gewährleisten sind nun noch die Punkte 2 und 3. Hierzu werden im folgenden Abschnitt die technische Spezifikation zum 3LGM² Construction Framework sowie die erforderlichen Änderungen am Tool3LGM² für die Interpretation der darüber beschreibbaren Konstruktionspatterns vorgestellt.

4.2 Use-Case

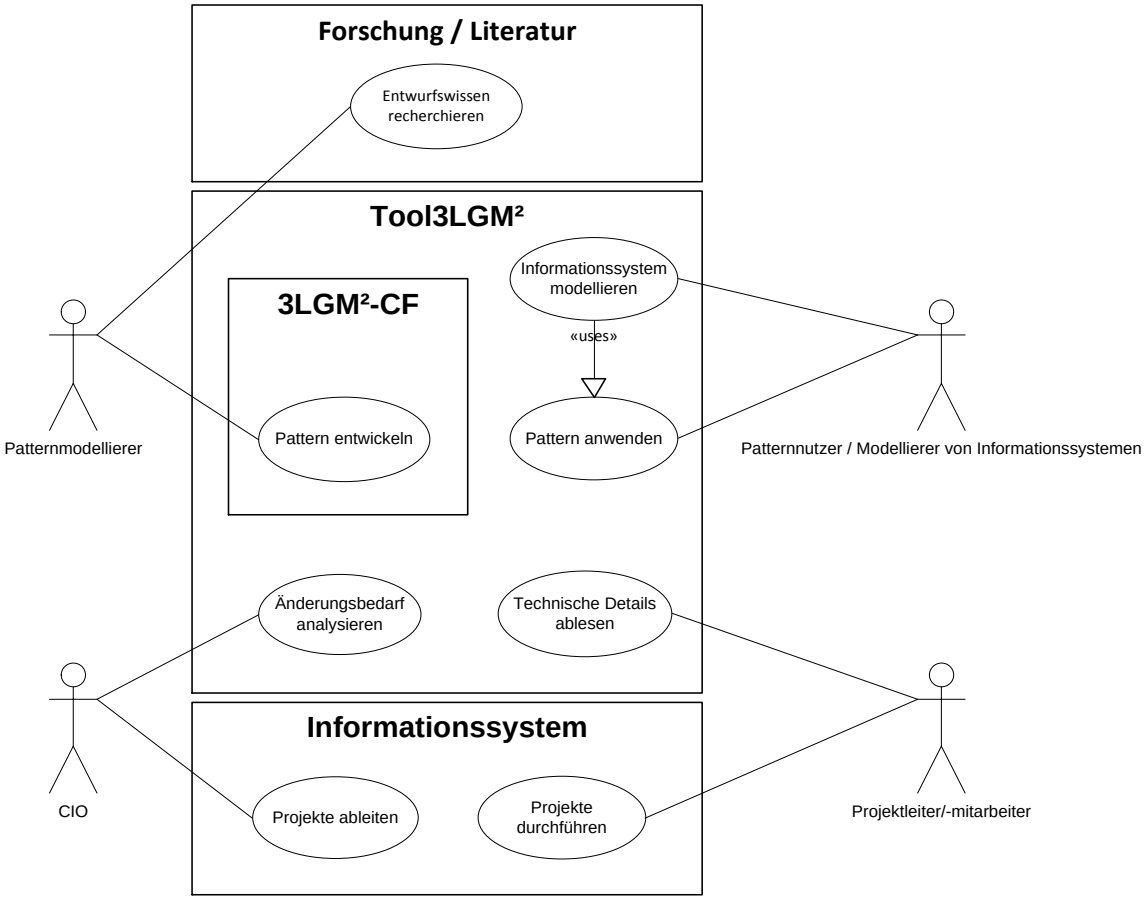

Abbildung 17 – Rollen und ihre Aufgaben im Bezug auf Konstruktionspatterns

Das Use-Case Diagramm zeigt die verschiedenen Akteure im Zusammenhang mit dem Einsatz von Entwurfswissen für die Modellierung. Initial wird ein Patternmodellierer relevantes Entwurfswissen identifizieren, welches in Form von Konstruktionspatterns umgesetzt werden soll. Im Bereich der Informationssysteme im Gesundheitswesen könnte es sich dabei etwa um Technical Frameworks der IHE handeln. Grundsätzlich obliegt es aber dem Patternmodellierer selbst, die Relevanz derartigen Wissens einzuschätzen, etwa gestützt auf Eigenschaften, wie Aktualität, Einsatz von Standards und bewährten Verfahren, Reputation der Autoren etc. Er wird dann die Erstellung von Konstruktionspatterns vornehmen und so das entsprechende Entwurfswissen formalisieren.

Daraufhin können Modellierer von Informationssystemen unter einer gegebenen Zielsetzung passende Konstruktionspatterns auswählen. Das darin enthaltene Wissen kann dann konstruktiv bei der Modellierung eingesetzt werden mit Unterstützung durch das Tool3LGM², welches die Patterns interpretiert und in Modellen abrufbar macht. Als Ergebnis entsteht ein Modell, welches den gewünschten Soll-Zustand des Informationssystems beschreibt. Dieses kann dann etwa vom CIO verwendet werden, um die notwendigen Änderungen am Informationssystem zu identifizieren und entsprechende Projekte abzuleiten. Dabei ist es möglich, auf Beschreibungen und weitere Attribute, welche aus dem Konstruktionspattern ins Modell übernommen wurden, zuzugreifen. Über sie werden beispielsweise technische Details bereitgestellt, die vom Projektteam direkt bei der Umsetzung der Änderungen angewendet werden können.

Es sollen hier allerdings keine Angaben gemacht werden bezüglich des Austausches zwischen Patternmodellierer und Patternanwender. Denkbar wäre etwa ein Bereitstellen über Repositories, in denen Institute, Forschungsabteilugen usw. Konstruktionspatterns zur Verfügung stellen, auf die dann durch Patternanwender zugegriffen werden können.

4.3 Aktivitäten

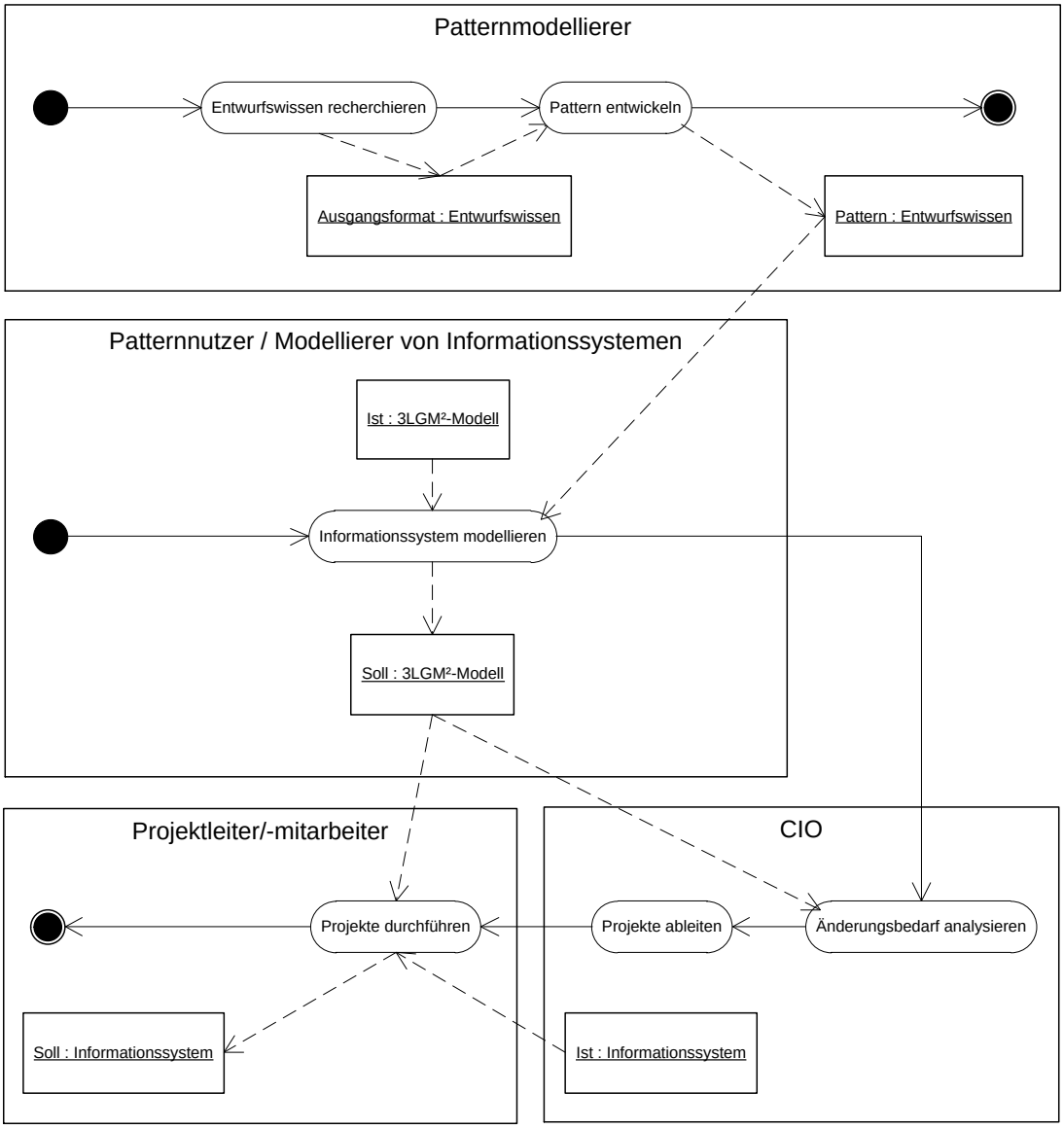

Abbildung 18 – Abläufe und Aktivitäten bei der Erstellung und Anwendung von Konstruktionspatterns

Abbildung 18 zeigt die Aktivitäten im Zusammenhang mit der Nutzbarmachung und Nutzung von Entwurfswissen über Konstruktionspatterns. Diese Aktivitäten werden in zwei parallele Prozessketten separiert:

1. Der Prozess der Erstellung von Konstruktionspatterns
2. Der Prozess der Anwendung von Konstruktionspatterns.

Hierdurch wird insbesondere gezeigt, dass das Formalisieren von Entwurfswissen im Allgemeinen unabhängig von der Anwendung geschehen sollte. Ein wesentliches Ziel der Definition von Konstruktionspatterns ist deren Wiederverwendbarkeit, womit eben diese Aufteilung begründet ist. Dies schließt grundsätzlich zwar nicht aus, dass das Entwickeln eines Konstruktionspatterns im Rahmen der Modellierung und Weiterentwicklung eines speziellen Informationssystems geschehen kann, soll aber zeigen, dass zum Zwecke der Wiederverwendbarkeit ein davon unabhängiges Erstellen dieser Patterns sinnvoller ist. Wie bereits in 4.2 geschildert, kann dann von dem so formalisierten Entwurfswissen bei der Anwendung in beliebig vielen Informationssystemmodellen profitiert werden.

4.4 Datentypen und Klassen

4.4.1 Datentypen

Das 3LGM² ist, wie in 2.1 gezeigt, ein UML Klassendiagramm und lässt sich für die Zwecke des 3LGM²-CF auf die Konzepte Klasse, Assoziation und Assoziationsklasse reduzieren. Da es, wie bereits in 3.3 geschildert wurde, das Ziel ist, eine Interpreterfunktion zu finden, welche von einem Konstruktionspattern auf das Zielmetamodell (hier das 3LGM²) abbildet, muss auf der hier betrachteten Meta-Metaebene die Grundlage für die Definition einer solchen Funktion in einem konkreten Pattern geschaffen werden. Dazu soll das Construction Framework die Möglichkeit bekommen, auf die hinter dem 3LGM² stehenden Konzepte abzubilden, das heißt auf Klassen, Assoziationen und Assoziationsklassen, sowie deren Attribute. Demnach werden die folgenden Datentypen benötigt, welche in der Unified Modeling Language spezifiziert sind (siehe [UML IS], [UML SU]):

Datentypbezeichnung	*Quellklasse*
Class	Uml::Classes::Kernel::Class
Association	Uml::Classes::Kernel::Association
AssociationClass	Uml::Classes::AssociationClasses::AssociationClass
Classifier[1]	Uml::Core::Abstractions::Classifiers:Classifier
string	Uml::Core::PrimitiveTypes::String
int	Uml::Core::PrimitiveTypes::Integer
unat	Uml::Core::PrimitiveTypes::UnlimitedNatural

4.4.2 Bezeichnungskonventionen

In den folgenden Abschnitten werden diverse Begriffe im Zusammenhang mit Modellierung und Metamodellierung verwendet werden. Diesen sollen zunächst kurz vorgestellt und erläutert werden, bevor auf weitere Details eingegangen wird.

Element: Instanz einer Klasse. Da sich das 3LGM²-CF im Kontext der Modellierung bewegt, wird dieser Terminus statt des Begriffes „Objekt“ verwendet, um anzuzeigen, dass es sich hierbei um Bestandteile eines Modelles handelt.

Beziehung/Verknüpfung: Instanz einer *Association*. Diese dienen der Verbindung von Elementen.

Klasse: Entspricht dem UML Konzept *Class*.

Assoziation: Alternativbegriff zur UML *Association*.

Assoziationsklasse: Wird wahlweise zum Begriff der UML *AssociationClass* verwendet.

Instanz: Oberbegriff für Element und Verknüpfung. Dieser bezieht sich auch auf Instanzen von Assoziationsklassen und vereinigt damit sowohl deren Element- als auch Verknüpfungseigenschaften.

Klassifizierung: Entspricht dem UML Konzept *Classifier* und abstrahiert damit von den Begriffen Klasse und Assoziation.

[1] Classifier wurde hier als abstrakte Oberklasse für Class und Association eingeführt, um eine geeignete Generalisierung für PatternNode und PatternRelation in Form des PatternObject definieren zu können.

4.4.3 Klassen

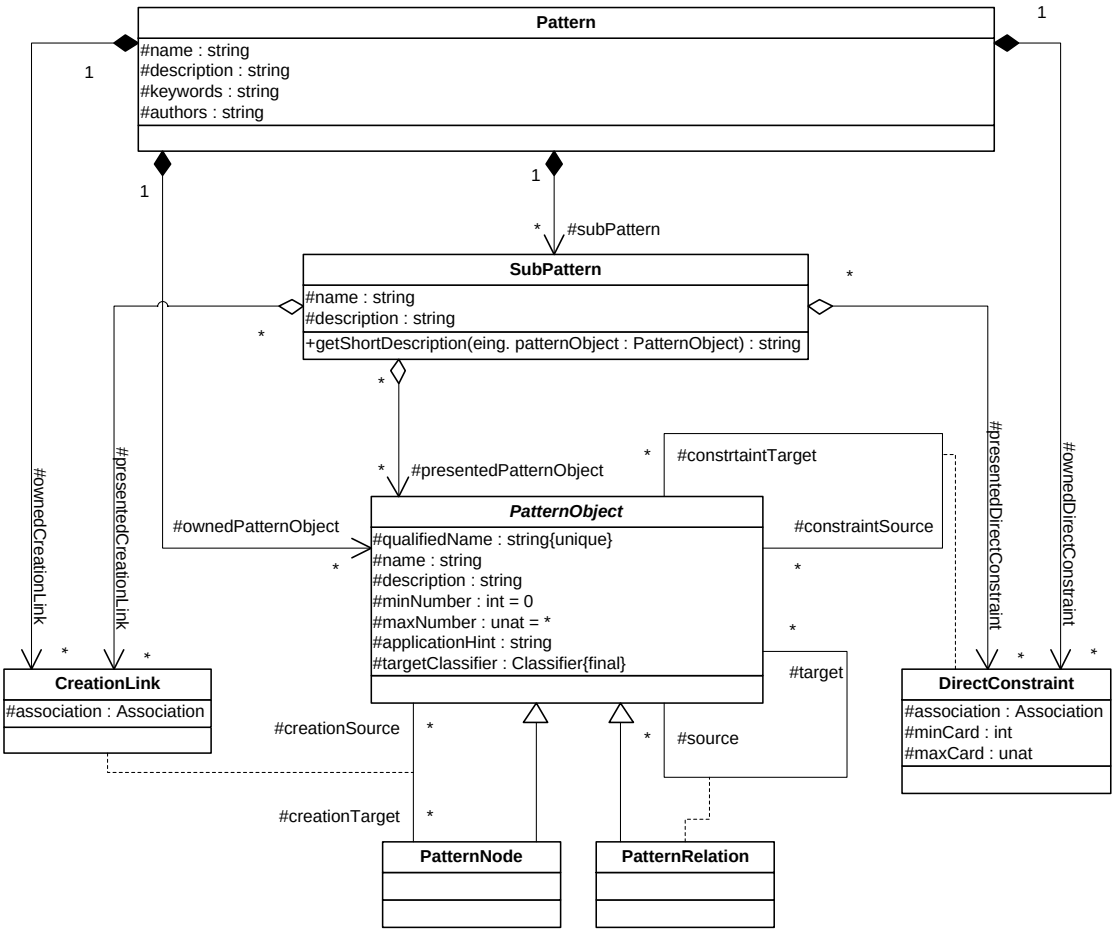

Abbildung 19 – Klassendiagramm des 3LGM² - Construction Framework

Abbildung 19 zeigt die Klassen und Beziehungen im 3LGM²-CF. Zu sehen ist die zentrale Klasse PatternObject, deren Spezialisierungen PatternNode bzw. PatternRelation als Vorlage für jeweils eine Klasse bzw. Assoziation im 3LGM² dienen. Alle PatternObjects erlauben die Definition von Beschreibungen, welche für die Instanzen der jeweiligen Klasse bzw. Assoziation übernommen werden können. Hierdurch lassen sich etwa wichtige Hintergrundinformationen aus dem Entwurfswissen verfügbar machen. Des Weiteren können Häufigkeiten darüber angegeben werden, wie oft von einem PatternObject abgeleitete Instanzen in einem Modell vorkommen dürfen.

Jedes PatternObject besitzt darüber hinaus eine beliebige Menge sogenannter CreationLinks. Diese werden dazu verwendet, automatisch neue Elemente und Verknüpfungen anzulegen. Dadurch ist es beispielsweise möglich, einen Anwendungsbaustein bei seiner Instanziierung mit speziellen Bausteinschnittstellen auszustatten, ohne dass der Modellierer diese händisch anlegen muss.

Außerdem können DirectConstraints definiert werden, welche angeben, ob und wie oft Instanzen, die von einem PatternObject abgeleitet sind, mit Instanzen, welche von einem anderen PatternObject abgeleitet wurden, über eine bestimmte Assoziation verbunden sein dürfen. So wäre es wiederum möglich, Einschränkungen diesbezüglich zu treffen, welche Bausteinschnittstellen ein Anwendungsbaustein besitzen darf bzw. von welchem PatternObject diese abgeleitet sein müssen, um für den Anwendungsbaustein angelegt werden zu können. PatternObjects werden in Patterns zusammengefasst, welche wiederum in SubPatterns untergliedert werden können. Dabei stellt jedes der SubPatterns einen Ausschnitt vom Gesamtpattern dar zum Zweck der logischen Gruppierung und Abgrenzung für die Darstellung spezieller Zusammenhänge.

Im Folgenden werden nun die einzelnen Klassen des 3LGM²-CF genauer vorgestellt.

4.4.3.1 PatternObject

Das PatternObject ist die zentrale Klasse im Construction Framework, deren Instanzen jeweils die Vorlage für eine Klasse oder Assoziation im 3LGM² darstellen. Elemente und Beziehungen in einem 3LGM²-Modell wiederum können als Ableitung eines PatternObject instanziiert werden.

Attribute

- qualifiedName : string {unique}
  Der eindeutige Bezeichner, der dieses PatternObject Pattern-übergreifend identifiziert.

- targetClassifier : Classifier {final}
  Gibt die Zielklasse bzw. Zielassoziation im 3LGM² wieder, auf die das PatternObject abgebildet wird und für deren Instanzen es als Vorlage dienen soll.

- name : string
  Der (anzeigbare) Name des PatternObject.

- description : string
  Die Beschreibung des PatternObject.

- minNumber : int = 0
  Die minimale Anzahl von Instanzen innerhalb eines Modelles, welche von diesem PatternObject abgeleitet sein sollen.
  Der Default-Wert für dieses Attribut ist 0.

- maxNumber : unat = *
  Die maximale Anzahl von Instanzen innerhalb eines Modelles, welche von diesem PatternObject abgeleitet werden dürfen.
  Der Default-Wert für dieses Attribut ist *.

- applicationHint : string
  Hinweise zur Anwendung des PatternObject bei der Modellierung. Dazu zählen insbesondere Angaben zu Art und Weise der Integration der abgeleiteten Instanzen in ein Modell, wie etwa die Beziehungen zu anderen Elementen, welche nicht als Teil dieses Patterns erfasst wurden. Jedes Konstruktionspattern kann durch seine PatternObjects nur bestimmte Teile oder Ausschnitte eines Informationssystems adressieren. Dieses Attribut soll daher Modellierern Aufschluss darüber geben, wie an den „Patterngrenzen“ zu verfahren ist, um eine geeignete Einbindung des Wissens in die Informationssystemmodelle zu erreichen.

Einschränkungen

[1] Der Wert des Attributes *qualifiedName* soll in Übereinstimmung mit den W3C Konventionen des XML Namespace für qualifizierte Namen (`QName`, siehe Abschnitt „4 Qualified Names [7]“ in [XML NS]) sein.

[2] Für alle PatternObjects soll gelten: $minNumber \leq maxNumber$.

[3] Für das Attribut *minNumber* soll gelten: $minNumber \geq 0$.

[4] Das *targetClassifier*-Attribut eines jeden PatternObject ist unveränderlich.

Grafische Repräsentation

PatternObjects im eigentlichen Sinne besitzen keine grafische Repräsentation, da es sich dabei um eine abstrakte Klasse handelt. Dennoch bestehen zwischen den von ihr abgeleiteten PatternNodes und PatternRelations hinreichend viele Ähnlichkeiten in ihrer Darstellung, dass hier eine entsprechende abstrakte grafische Repräsentation als Vorlage für beide gegeben werden kann.

PatternObjects werden zu Darstellungszwecken in sogenannte SubPatterns zusammengefasst, die jeweils einen Ausschnitt aus einem Pattern widerspiegeln. Jedes PatternObject sollte pro SubPattern durch ein Rechteck repräsentiert werden, das sich in die drei untereinander liegenden Bereiche

- Name,
- Abbildungsregeln und
- Kurzbeschreibung

untergliedert. Jeder der Bereiche sollte dabei durch eine horizontale Linie von den anderen abgegrenzt werden, wie es in Abbildung 20 zu sehen ist.

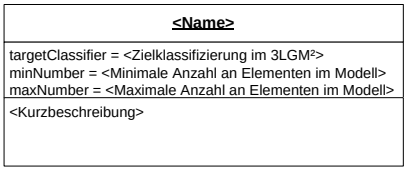

Abbildung 20 – Abstrakte grafische Darstellung für PatternObjects

Der Name ist immer im oberen Bereich des PatternObjects dargestellt. Darunter werden die Abbildungsregeln angegeben, welche die Werte von

- PatternObject.*targetClassifier*,
- PatternObject.*minNumber* sowie
- PatternObject.*maxNumber*

enthalten. Der *targetClassifier* sollte dabei eindeutig die Klassifizierung im 3LGM² erkenntlich machen, auf welche das PatternObject abgebildet wird. Der untere Bereich dient abschließend der Anzeige einer Kurzbeschreibung, welche pro SubPattern und pro PatternObject definiert wird. Der Wert hierfür wird durch den Aufruf der Operation SubPattern.*getShortDescription(PatternObject)* für das jeweilige SubPattern und das jeweilige darin enthaltene PatternObject erhalten.

Grundsätzlich sollte der Name immer sowohl horizontal als auch vertikal zentriert dargestellt werden. Die Verankerung für den Text der Abbildungsinformationen sowie Kurzbeschreibung ist sprachsensitiv zu wählen, beispielsweise links-oben für lateinische Sprachen, rechts-oben für Arabisch oder Japanisch.

Anzumerken ist noch, dass hier keine Forderungen hinsichtlich einer eventuellen Farbgebung für Hintergrund oder Text gemachten werden, da dies als unbedenklich gilt. Potentiell könnten so etwa bestimmte Aspekte betont und spezielle Zusammenhänge zusätzlich unterstrichen werden. Es obliegt allerdings jeweils der konkreten Implementierung des 3LGM²-CF, dies zuzulassen oder zu unterbinden.

Anmerkungen

1. Das *description*-Attribut wird in 3LGM²-Modelle übertragen:

   Dazu wird der Beschreibungstext eines jeden Elements bzw. einer jeden Verknüpfung, die von diesem PatternObject abgeleitet ist, auf den Wert dieses Attributes gesetzt. Hierüber können umfangreiche vorgefertigte Informationen zur Verfügung gestellt werden, die später dann direkt im Modell abrufbar sind. Diese Fähigkeit ist ein wesentlicher Aspekt, der den Vorlagencharakter eines PatternObject kennzeichnet.

4.4.3.2 PatternNode

Diese Klasse stellt eine Spezialisierung des PatternObject dar, welche auf Klassen im 3LGM² abgebildet wird.

Attribute

Keine weiteren Attribute

Einschränkungen

[1] Das *targetClassifier*-Attribut sollte eine Instanz vom Datentyp Class sein, welche die Zielklasse im 3LGM² wiedergibt, auf die der PatternNode abgebildet wird.

Grafische Repräsentation

Die grafische Darstellung für PatternNodes ist identisch mit den Vorgaben der allgemeinen grafischen Repräsentation von PatternObjects. Der *targetClassifier* stellt dabei eine Klasse im 3LGM² dar. Zusätzliche Veränderungen und Erweiterungen der Darstellung sind nicht erforderlich.

4.4.3.3 PatternRelation

Diese Klasse stellt ebenfalls eine Spezialisierung des PatternObject dar. PatternRelations dienen allerdings der Verknüpfung von PatternObjects zur Darstellung von Zusammenhängen im Pattern. Diese werden im 3LGM² auf binäre Assoziationen oder Assoziationsklassen abgebildet.

Attribute

- source : PatternObject [*]
  Das erste Assoziationsende.

- target : PatternObject [*]
  Das zweite Assoziationsende.

Einschränkungen

[1] Das *targetClassifier*-Attribut sollte eine Instanz vom Datentyp Association sein, welche die Zielassoziation im 3LGM² repräsentiert, auf die die PatternRelation abbildet.

[2] Eine PatternRelation darf nur auf eine Assoziation abgebildet werden, welche die *targetClassifiers* der durch sie verbundenen PatternObjects im 3LGM² verbindet.

[3] Die *source*- bzw. *target*-Attribute sind Teilmengen von Pattern.*ownedPatternObject*.

Grafische Repräsentation

Eine PatternRelation erweitert darstellungstechnisch das Konzept des PatternObject, welche in Anlehnung an die AssociationClass der UML durch eine mit dem Rechteck verknüpften Linie repräsentiert wird.

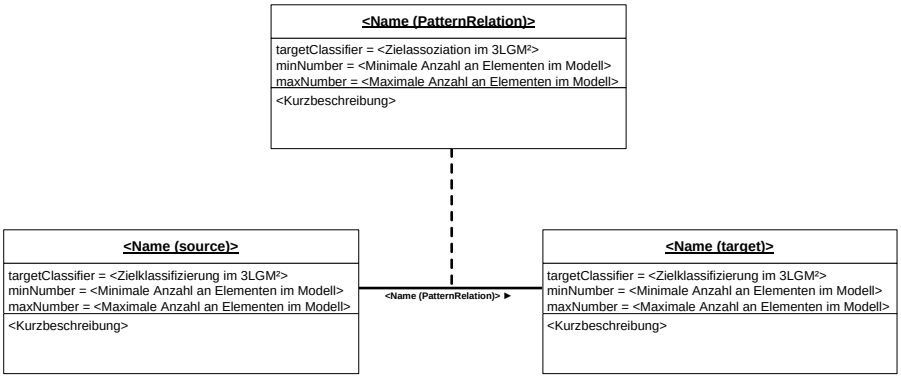

Abbildung 21 - Grafische Darstellung einer PatternRelation

In Abbildung 21 wird hierzu eine PatternRelation gezeigt, welche die PatternObjects „source“ und „target“ verbindet. Im Rechteck sollten dabei analog zum PatternObject der Name, die Abbildungsinformationen und die Kurzbeschreibung dargestellt werden. Die Linie, welche die PatternObjects verbindet, sollte durchgängig sein und ebenfalls mit dem Namen der PatternRelation versehen werden. Darüber hinaus ist diesem ein Dreieckssymbol (`U+25BA`, `U+25C4`, `U+25B2` oder `U+25BC` im Unicode) anzufügen, welches die Richtung vom *source*- zum *target*-Ende der PatternRelation angibt. Rechteck und Verbindungslinie sind abschließend durch eine gestrichelte Linie zu verknüpfen.

Anmerkungen

2. PatternRelations sind für alle Elemente verbindlich:

   Eine Assoziation, welche von einer PatternRelation abgeleitet werden soll, darf ausschließlich zwischen Elementen angelegt werden, die von *source* bzw. *target* abgeleitet sind. Darüber hinaus ist auch ein Verbinden von Elementen, die von keinem PatternObject abgeleitet wurden, nicht möglich.

4.4.3.4 CreationLink

CreationLinks ermöglichen das automatische Anlegen von Elementen bei der Instanziierung von Klassen oder Assoziationsklassen als Ableitung eines PatternObject. Das jeweils neu angelegte Element ist wiederum eine Ableitung eines PatternNode und wird über festgelegte Verknüpfungen mit der anfangs erzeugten Instanz verbunden.

Attribute

- creationSource : PatternObject [*]
  Das PatternObject, dessen abgeleitete Instanzen das Neuanlegen eines weiteren Elementes auslösen.

- creationTarget : PatternNode [*]
  Der PatternNode, von dem das zu instanziierende Element abgeleitet sein soll.

- association : Association
  Die Assoziation, über die die initial erzeugte Instanz mit dem neu angelegten Element verbunden wird.

Einschränkungen

[1] Das Attribut *association* muss eine Assoziation im 3LGM² sein, welche die *targetClassifiers* von *creationSource* und *creationTarget* verbindet.

[2] Das *creationSource.targetClassifier*-Attribut sollte eine Klasse oder Assoziationsklasse des 3LGM² sein. Für eine reine Assoziation kann ersichtlicherweise kein CreationLink definiert werden, da diese selbst über keine Beziehungen verfügen können.

[3] *creationSource* und *creationTarget* sind Teilmengen von Pattern.*ownedPatternObject*.

Grafische Repräsentation

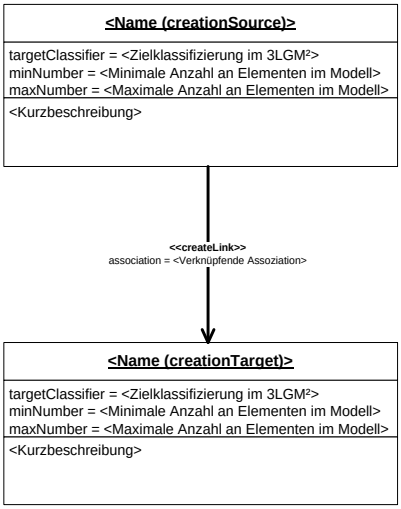

Abbildung 22 – Grafische Darstellung eines CreationLink

Jeder CreationLink ist durch eine durchgängige Linie darzustellen, welche durch den Stereotyp „createLink“ – gefasst innerhalb doppelter spitzer Klammern – gekennzeichnet ist und darüber hinaus in eindeutiger Weise die Assoziation erkennbar macht, welche die *targetClassifiers* von *creationSource* und *creationTarget* verbindet.

Wie in Abbildung 22 dargestellt, sind das *creationSource*- und *creationTarget*-Ende des CreationLink mit den entsprechenden PatternObjects zu verbinden. Das *creationTarget*-Ende ist dabei durch einen Pfeil am entsprechenden Ende der Linie darzustellen.

Bei Erzeugen einer von *creationSource* abgeleiteten Instanz wird daraufhin ein Element als Ableitung des *creationTarget* angelegt und über die gegebene Assoziation mit ihm verknüpft.

Anmerkungen

1. Das Neuanlegen des zu verknüpfenden Elementes sollte unter bestimmten Umständen entfallen:

   Handelt es sich beim *creationTarget* um einen PatternNode für den *maxCard = 1* gilt, so korrespondiert dieser zu höchstens einem Element im Modell und ist damit nicht mehr Vorlage für eine komplette Klasse im 3LGM², sondern nur für ein Element. Zeigt ein CreationLink auf solch einen PatternNode, sollte daher nur dann eine Neuinstanziierung erfolgen, wenn das Element noch nicht im Modell vorhanden ist. Ist dieses allerdings schon angelegt worden, genügt es, die Verknüpfung mit eben diesem einen über die spezifizierte Assoziation herzustellen.

4.4.3.5 DirectConstraint

DirectConstraints erlauben das Einschränken der Beziehungen, die Instanzen innerhalb eines Modelles miteinander eingehen können. Hierzu kann festgelegt werden, dass eine Instanz, welche von einem PatternObject abgeleitet wurde, über eine bestimmte Assoziation nur mit Instanzen verknüpft werden kann, die von einem anderen bestimmten PatternObject abgeleitet wurden. Zusätzlich können noch Aussagen darüber getroffen werden, wie oft solche Verknüpfungen jeweils existieren dürfen.

Attribute

- constraintSource : PatternObject [*]
  Der Ausgangspunkt des Constraint. Instanzen, die von diesem PatternObject abgeleitet sind, dürfen nur eingeschränkt Beziehungen zu anderen Instanzen besitzen.

- constraintTarget : PatternObject [*]
  Der Endpunkt des Constraint, dessen von ihm abgeleiteten Instanzen die möglichen Verknüpfungen der Instanzen des *constraintSource* beschränken.

- association : Association
  Die Assoziation, über die die Einschränkung angewendet werden soll.
  Für eine Instanz, die von *constraintSource* abgeleitet ist, wird beim Anlegen einer davon ausgehenden Verknüpfung dieses Typs überprüft, ob die Instanz am anderen Ende vom *constraintTarget* abgeleitet wurde.

- minCard : int
  Die minimale Anzahl an Verknüpfungen, die eine von *constraintSource* abgeleitete Instanz mit Instanzen des *constraintTarget* über die *association* besitzen soll.

- maxCard : unat
  Die maximale Anzahl an Verknüpfungen, die vom *constraintSource* abgeleitete Instanzen mit Instanzen des *constraintTarget* über die *association* besitzen sollten.

Einschränkungen

[1] Das Attribut *association* muss eine Assoziation im 3LGM² sein, welche die *targetClassifiers* von *constraintSource* und *constraintTarget* verbindet.

[2] Die Attribute *minCard* bzw. *maxCard* dürfen die im 3LGM² definierten Kardinalitäten für das zum *constraintTarget* korrespondierende Ende des *association*-Attributes nicht unter- bzw. überschreiten.

[3] *constraintSource* und *constraintTarget* sind Teilmengen von Pattern.*ownedPatternObject*.

[4] Für all DirectConstraints soll gelten: $minCard \leq maxCard$.

[5] Für das Attribut *minCard* soll gelten: $minCard \geq 0$.

Grafische Repräsentation

Ein DirectConstraint ist analog zum CreationLink durch eine durchgängige Linie darzustellen, welche hier die PatternObjects am *constraintSource*- und *constraintTarget*-Ende verbindet. Die Linie ist dabei durch den in doppelten spitzen Klammern gefassten Stereotyp „constrainedBy“ zu versehen. Des Weiteren soll durch das Attribut *association* in eindeutiger Weise die entsprechende Assoziation im 3LGM² dargestellt werden. Die Attribute *minCard* bzw. *maxCard* sind anzufügen, falls diese von der

minimalen bzw. maximalen Kardinalität abweichen, die im 3LGM² für das zum *constraintTarget* korrespondierende Ende der Assoziation im 3LGM² definiert wurde. Sonst dürfen diese Attribute in der Darstellung entfallen. Die Abbildung 23 zeigt hierzu zwei PatternObjects („*constraintSource*“ und „*constraintTarget*“), welche über einen DirectConstraint verbunden sind. Das *constraintTarget*-Ende ist dabei immer mit einem Pfeil zu versehen, wie es in der Darstellung gezeigt wird. Die Interpretation hierbei ist, dass eine Instanz, welche vom *constraintSource* abgeleitet ist, über die gegebene Assoziation in einem 3LGM²-Modell mindestens *minCard*-mal und höchstens *maxCard*-mal mit Instanzen verbunden sein darf, welche vom *constraintTarget* abgeleitet wurden.

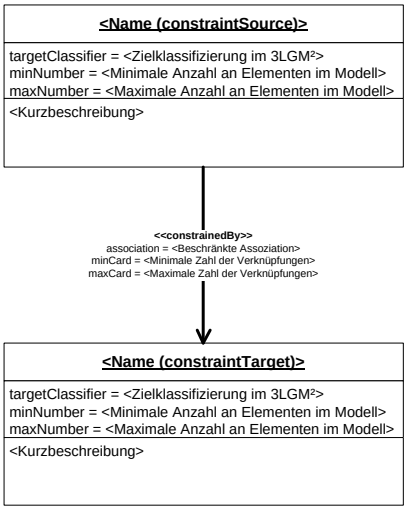

Abbildung 23 – Beispiel eines DirectConstraint

Häufig ist es der Fall, dass Constraints paarweise auftreten: Einmal „vom source zum target“ und einmal „vom target zum source“ über dieselbe Assoziation. Dieser Umstand ermöglicht eine Kompression der Darstellung in Form eines bidirektionalen DirectConstraint, welcher rein grafisch die beiden Constraints kapselt und damit mehr Übersichtlichkeit schafft. Hierzu wird in der Mitte der Linie nur der Stereotyp und die Assoziation angeben und an den Enden jeweils die *minCard*- und *maxCard*-Attribute angetragen. Folgende Darstellungen sind dann äquivalent:

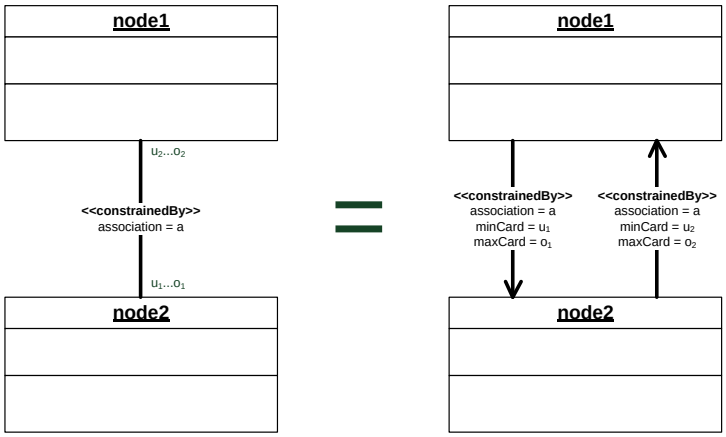

Abbildung 24 – Bidirektionaler DirectConstraint; komprimierte Darstellung (links), separierte Darstellung (rechts)

Anmerkungen

1. DirectConstraints mit demselben *constraintSource*- und *association*-Attribut sind konjunktiv zu interpretieren:

   Eine Instanz, welche von einem PatternObject abgeleitet wurde, das innerhalb mehrerer DirectConstraints, welche dasselbe *association*-Attribut besitzen, als *constraintSource* gesetzt ist, darf mit allen Instanzen verbunden sein, welche von PatternObjects abgeleitet wurden, die als *targetSource* dieser DirectConstraints gesetzt sind.

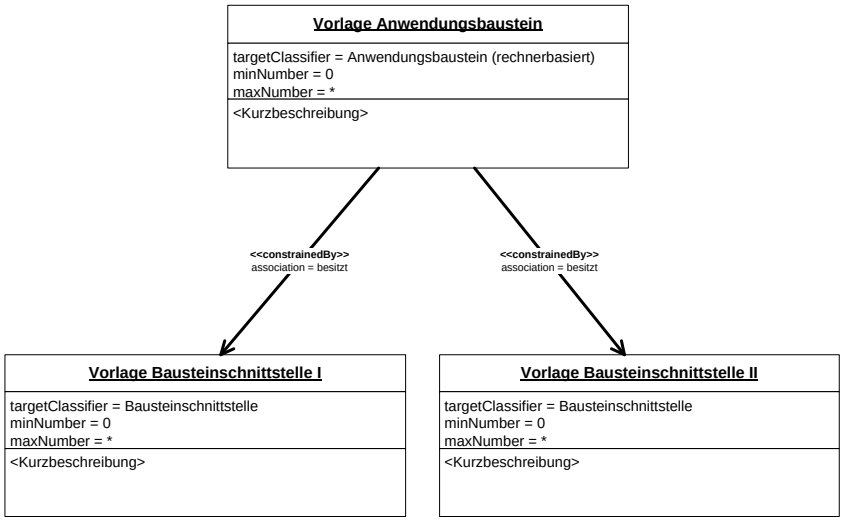

Abbildung 25 – Beispiel eines PatternNode als constraintSource mehrerer DirectConstraints

   Abbildung 25 zeigt dazu ein Beispiel, bei dem Anwendungsbausteine, welche vom PatterNode „Vorlage Anwendungsbaustein“ abgeleitet wurden, mit allen Bausteinschnittstellen, die von „Vorlage Bausteinschnittstelle I“ oder „Vorlage Bausteinschnittstelle II“ abgeleitet sind, über die „besitzt“-Assoziation verbunden werden können.

2. DirectConstraints sind nur für PatternObjects verbindlich:

   Ist eine Instanz von einem PatternObject abgeleitet, das als *constraintSource* innerhalb eines DirectConstraint gesetzt ist, darf sie, wie bereits geschildert, über die zum Constraint gehörige Assoziation nur mit Instanzen verbunden sein, die vom *constraintTarget* abgeleitet sind. Es gilt allerdings: Die Instanz darf über diese Assoziation auch mit all denen verbunden sein, die von keinem PatternObject abgeleitet wurden.

   Betrachtet man etwa das Beispiel aus Abbildung 25, dürfen hier Anwendungsbausteine, die von „Vorlage Anwendungsbaustein“ abgeleitet sind, mit allen Bausteinschnittstellen über die besitzt-Assoziation verbunden sein, die von „Vorlage Bausteinschnittstelle I“, „Vorlage Bausteinschnittstelle II“ oder von keinem PatternObject abgeleitet wurden.

   Dieser Punkt ist von zentraler Bedeutung, da nur so das gemischte Arbeiten mit und ohne Patterns möglich ist.

3. Minimale und maximale Kardinalität sind optional:

   Wurden keine Werte für *minCard* bzw. *maxCard* gesetzt, werden stattdessen die für die Assoziation im 3LGM² gegeben Kardinalitäten verwendet.

4. PatternObjects ohne Constraints erlauben beliebiges verknüpfen:

   Ist für ein PatternObject kein DirectConstraint bezüglich einer bestimmten Assoziation definiert, dürfen davon abgeleitete Elemente uneingeschränkt über sie Verknüpfungen eingehen.

4.4.3.6 Pattern

Ein Pattern ist eine Menge von PatternObjects inklusive der für sie definierten CreationLinks und DirectConstraints, welche zum Zweck der Formalisierung bestimmten Entwurfswissens in 3LGM²-konformer Weise im Pattern zusammengefasst werden. Instanzen dieser Klasse sind dann Konstruktionspatterns für Informationssystemmodelle zum 3LGM².

Attribute

- name : string
  Die Bezeichnung des Patterns.

- description : string
  Die Beschreibung des Patterns. Diese sollte vor allem Inhalt, Anwendungsgebiet und Zweck beinhalten, um dem Patternnutzer einen allgemeinen Einblick in die Thematik und in die Einsatzmöglichkeiten zu geben.

- keywords : string
  Die Schlagworte zum Pattern. Diese sollten kommasepariert in diesem Attribut erfasst werden und jeweils repräsentativ für das Pattern stehen.

- authors : string
  Dieses Attribut gibt den oder die Autoren des Patterns wieder. Diese sollte ebenfalls kommasepariert aufgelistet werden.

- ownedPatternObject : PatternObject [*]
  Die PatternObjects des Patterns.

- ownedCreationLinks : CreationLink [*]
  Die CreationLinks, die für die *patternObjects* definiert wurden.

- ownedDirectConstraints : DirectConstraint [*]
  Die DirectConstraints, die für die *patternObjects* definiert wurden.

Einschränkungen

[1] Ein Pattern enthält keine zwei DirectConstraints, die in den Attributen *creationSource*, *creationTarget* und *association* übereinstimmen. Damit wird das Definieren uneindeutiger Kardinalitäten verhindert.

[2] Die Struktur des Attributes *keywords* sollte unter Verwendung des Unicodes und der Spezifikation `Name` des XML Namespace (siehe „2.3 Common Syntactic Constructs [5]“ in [XML]) dem regulären Ausdruck `((Name · U+002C · U+0020?)* · Name)` entsprechen, wobei `Name` jeweils ein Keyword repräsentiert (der Punktoperator beschreibt hierbei die Konkatenation). Damit erhält man eine kommaseparierte Aneinanderreihung der einzelnen Schlagworte.

Bsp.: „IHE, XDS, Gesundheitswesen, transinstitutionell, Dokumentenaustausch, Informationssysteme“

Weitere Einschränkungen für dieses Attribut sind möglich.

[3] Die Struktur des Attributes *authors* sollte unter Verwendung des Unicodes und der Spezifikation `Names` des XML Namespace (siehe „2.3 Common Syntactic Constructs [6]“ in [XML]) dem regulären Ausdruck `((Names · U+002C · U+0020?)* · Names)` entsprechen, wobei `Names` jeweils einen Autor repräsentiert. Damit ergibt sich ein kommaseparierter String der einzelnen Autoren.

Bsp.: „Jane Doe, Jan Novák“

Weitere Einschränkungen für dieses Attribut sind möglich.

Grafische Repräsentation

Für diese Klasse wird keine explizite grafische Repräsentation definiert, stattdessen sollte sie als Sammlung von PatternObjects inklusive der für sie definierten Beziehungen und Constraints verstanden werden. Die Darstellung eines Patterns wird vielmehr durch eine konkrete Implementierung des 3LGM²-Construction Framework gegeben. Derartige softwareseitige Umsetzungen sollten dabei die Vorgaben zu den in den vorangegangenen Abschnitten gegebenen grafischen Repräsentationen der einzelnen Bestandteile eines Patterns erfüllen. Insbesondere sollten dabei auch die Attribute, die in diesen Vorgaben nicht dargestellt wurden, wie zum Beispiel PatternObject.*qualifiedName*, auf anderem Wege zugänglich gemacht werden, etwa über Eigenschaftendialoge oder vergleichbare GUI-Komponenten. In gleicher Weise können Attribute des Patterns selbst, wie etwa *description* oder *authors* visualisiert werden.

Die Darstellung von PatternObjects, DirectConstraints und CreationLinks erfolgt dann in sogenannten SubPatterns, welche diese zum Zwecke der grafischen Repräsentation vereinen.

4.4.3.7 SubPattern

SubPatterns sind logische Zerlegungen eines Patterns, welche PatternObjects inklusive deren Beziehungen und Constraints zur Herausstellung bestimmter Zusammenhänge und Teilaspekte zusammenfassen. Sie bilden dabei die Grundlage für deren grafische Repräsentation, genauer gesagt sollen PatternObjects, DirectConstraints und CreationLinks immer im Verbund als SubPattern dargestellt werden. Die SubPatterns eines Patterns können sich hierbei überschneiden.

Attribute

- name : string
  Die Bezeichnung des SubPatterns.

- description : string
  Die Beschreibung des SubPatterns. Diese sollte vor allem die dargestellten Zusammenhänge erläutern und wichtige Zusatzinformationen bieten.

- presentedPatternObject : PatternObject [*]
  Die dargestellten PatternObjects.

- presentedDirectConstraint : DirectConstraint [*]
  Die dargestellten DirectConstraints.

- presentedCreationLink : CreationLink [*]
  Die dargestellten CreationLinks.

Operationen

- getShortDescription(patternObject : PatternObject) : string
  Diese Operation soll für jedes PatternObject des SubPatterns die Kurzbeschreibung zurückgeben (diese ist im unteren Bereich seiner grafischen Darstellung angesiedelt). Ein PatternObject kann damit in verschiedenen SubPatterns unterschiedliche Kurzbeschreibungen besitzen, dabei sollte diese jeweils einen knappen Vermerk zur dessen Bedeutung im Kontext des SubPatterns wiedergeben.

Einschränkungen

[1] *presentedPatternObject* ist eine Teilmenge von Pattern.*ownedPatternObject*.

[2] *presentedDirectConstraint* ist eine Teilmenge von Pattern.*ownedDirectConstraint*.

[3] *presentedCreationLink* ist eine Teilmenge von Pattern.*ownedCreationLink*.

Grafische Repräsentation

Für SubPatterns selbst gibt es analog zum Pattern keine definierte grafische Darstellung. Ebenso repräsentieren diese eine Sammlung von Patternbestandteilen, wobei es Aufgabe einer Implementierung des 3LGM²-CF ist, diese entsprechend der für sie gesetzten Anforderungen darzustellen. Für ein SubPattern selbst sollte es dabei ebenfalls möglich sein, Zugriff auf dessen Attribute, wie *name* oder *description* zu erhalten, etwa über Eigenschaftendialoge in Analogie zum Pattern.

Anmerkungen

1. SubPatterns dürfen sich überschneiden:

   Dies bedeutet konkret, dass bei zwei SubPatterns jeweils der mengentheoretische Schnitt ihrer *presentedPatternObject*-, *presentedCreationLink*- bzw. *presentedDirectConstraint*-Attribute ungleich der Leeren Mengen sein darf.

4.4.4 Interpreterfunktion des 3LGM²-CF

In den gerade gegebenen Definitionen der Klassen des 3LGM²-CF wurden bereits häufiger Punkte im Bezug auf die Abbildung im 3LGM² angesprochen. Die Interpreterfunktion wird hierbei über Attribute dieser Klassen ausgedrückt. Um dies zu verdeutlichen, wird hier nun gezeigt, wie sich diese zur Interpreterfunktion zusammensetzen.

(Zur Erinnerung: Diese war gegeben durch i: $\mathcal{P} \longrightarrow \mathcal{M}$, wobei jedem Element und jeder Beziehung im Pattern ein Äquivalent im Metamodell zugeordnet wird. Diese Elemente bzw. Beziehungen sind in diesem Fall Klassen bzw. Assoziationen).

Hierzu ist nun zunächst zu klären, wann – im Sinne der Mengenlehre – von einem Element eines Patterns gesprochen werden kann. Die diesbezüglich relevanten Teilmengen sind dabei die der PatternObjects, der DirectConstraints und der CreationLinks. Für ein Element *e* gilt dann:

$$e \in \mathcal{P} \equiv$$

$$e \in \mathcal{P}.ownedPatternObject \vee e \in \mathcal{P}.ownedDirectConstraint \vee e \in \mathcal{P}.ownedCreationLink$$

Das heißt, das Element muss über eine dieser drei für das Pattern definierten Kompositionen in ihm enthalten sein.

Unter dieser Voraussetzung ergibt sich nun die Interpreterfunktion wie folgt:

$$\mathrm{i}(e) = \begin{cases} e.targetClassifier, \text{ falls } e \text{ ein PatternObject ist} \\ e.association, \text{ falls } e \text{ ein DirectConstraint oder CreationLink ist} \end{cases}$$

Alle weiteren Attribute werden nicht durch diese Funktion erfasst. Deren Interpretation hängt stattdessen von einer konkreten Implementierung des 3LGM²-CF unter den jeweils für sie definierten Vorgaben und Einschränkungen ab. Auch werden hierdurch nicht die SubPatterns abgedeckt, da diese nur eine logische Untergliederung des Patterns darstellen und daher nicht zwingend abgebildet werden müssen, um ein Pattern zielmetamodellkonform zu gestalten.

Anmerkung: Es obliegt dabei dem Modellierer des Patterns, die für die Abbildung notwendigen Attribute zu spezifizieren. Dies geht allerdings einher mit einer Interpretation der Konzepte des Entwurfswissens im Zielmetamodell, sprich im 3LGM². Demnach ist ein Konstruktionspattern immer eine Auslegung des Patternmodellierers, der das Wissen auf diese Weise formalisiert.

Im Folgenden soll nun ein Beispiel eines 3LGM²-konformen Konstruktionspatterns gegeben werden.

4.4.5 Beispiel

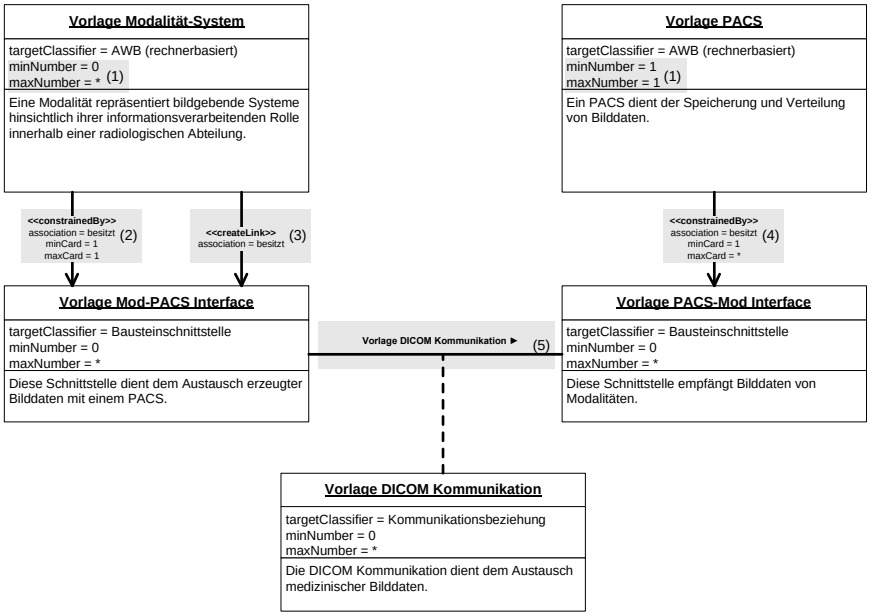

Abbildung 26 – Beispiel eines Patterns zum DICOM-Nachrichtenaustausch; dargestellt sind dabei dessen PatternObjects, CreationLinks und DirectConstraints entsprechend der grafischen Konventionen.

Abbildung 26 greift das Beispiel aus Abbildung 16 auf und zeigt die DICOM-basierte Kommunikation von Modalitäten mit einem PACS als Konstruktionspattern. Zu sehen sind die PatternNodes „Vorlage Modalität-System“, „Vorlage PACS“, „Vorlage Mod-PACS Interface“ und „Vorlage PACS-Mod Interface“ sowie die PatternRelation „Vorlage DICOM Kommunikation“ und einige DirectConstraints und CreationLinks. Es sind hierbei folgende strukturelle Einschränkungen definiert:

(1) Innerhalb eines Modelles darf es beliebig viele Modalitäten geben, aber nur genau ein PACS. Dieser Sachverhalt wird durch die Attribute *minNumber* und *maxNumber* ausgedrückt.

(2) Jede Modalität besitzt dabei genau ein Mod-PACS Interface. Schnittstellen, die von anderen PatternNodes abgeleitet wurden, dürfen nicht angelegt werden.

(3) Bei der Instanziierung eines Anwendungsbausteines als Ableitung vom PatternNode Modalität-System wird dieser mit einer neuen von Mod-PACS Interface abgeleiteten Bausteinschnittstelle ausgestattet.

(4) Der PACS soll mindestens ein PACS-Mod Interfaces als Bausteinschnittstelle besitzen. Das Anlegen von Schnittstellen, die von anderen PatternNodes abgeleitet sind, sollte unterbunden werden.

(5) Kommunikationsbeziehungen, die von der Vorlage DICOM Kommunikation abgeleitet sind, dürfen nur Mod-PACS- und PACS-Mod-Schnittstellen verbinden. Auch das Verbinden von Schnittstellen, die von keinem PatternNode abgeleitet wurden, ist unzulässig.

Hieran ist nun zu sehen, wie es Konstruktionspatterns ermöglichen, Entwurfswissen zu formalisieren. Neben den eben beschriebenen strukturellen Vorgaben, sind darüber hinaus noch kurze Beschreibungen gegeben, welche zunächst ein allgemeines Verständnis des dargestellten Sachverhaltes vermitteln. Des Weiteren können noch (hier nicht gezeigte) detaillierte Beschreibungen im *description*-Attribut oder auch Modellierungshinweise im *applicationHint*-Attribut hinterlegt werden. Der Zugriff auf diese sollte in einer konkreten Implementierung des 3LGM²-CF etwa über Eigenschaftendialoge, Quickinfos etc. ermöglicht werden. Im Folgenden soll dazu gezeigt werden, wie eine Umsetzung im Tool3LGM² realisiert werden kann.

4.5 Umsetzung im Tool3LGM²

Das Tool3LGM² als Modellierungswerkzeug für Informationssysteme basiert auf dem 3LGM² und bietet damit die Grundvoraussetzungen für eine softwareseitige Umsetzung des 3LGM² - Construction Framework. Für die Nutzbarmachung von Entwurfswissen für die Modellierung sollte das Tool3LGM² dazu das Tool3LGM²-CF entsprechend der eben gegebenen Spezifikation implementieren. Hierbei gilt es insbesondere die beiden folgenden Funktionen bereitzustellen:

1. Das Erzeugen von Konstruktionspatterns selbst
2. Der Zugriff auf diese Konstruktionspatterns bzw. deren Inhalt während der Modellierung.

Für beide Punkte soll nun gezeigt werden, wie sich eine mögliche Realisierung im Tool3LGM² gestalten könnte. Dabei wird großes Augenmerk auf die Wiederverwendbarkeit der bereits vorhandenen Komponenten gelegt.

4.5.1 Patternerstellung

Der Prozess des Erstellens von Konstruktionspatterns ist der erste Schritt auf dem Weg hin zum Einsatz von Entwurfswissen bei der Modellierung mit dem Tool3LGM². Ziel ist es hier, die Erstellung wohldefinierter Konstruktionspatterns zu unterstützen, wozu nun Möglichkeiten für die Umsetzung im Tool3LGM² aufgezeigt werden sollen. Es handelt sich dabei im Folgenden um Vorschläge, die vor allem zeigen sollen, welche Punkte bei der Implementierung des 3LGM²-CF zu beachten sind.

4.5.1.1 Menü- und Werkzeugleiste

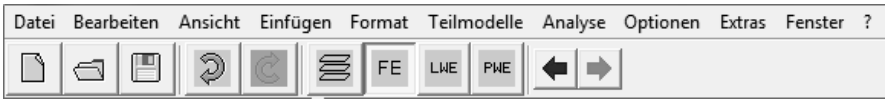

Abbildung 27 – Menü- und Werkzeugleiste im Tool3LGM²

Datei

Das Dateimenü diente in der üblichen Anwendung des Tool3LGM², vor allem dem Erstellen, Laden und Speichern von Modellen. Patterns selbst sollten vergleichbar gehandhabt werden können, sodass auch für sie diese Funktionen zur Verfügung stehen. Dafür ist zunächst diesem Menü der Punkt Neues Konstruktionspattern hinzuzufügen. Wird so ein neues Pattern angelegt, soll sich daraufhin das Tool3LGM² dem Kontext der Konstruktionspatterns anpassen, einschließlich aller weiteren Menüs und Menüpunkte, wie es im Folgenden beschrieben wird:

Das Datei-Menü bot bereits für 3LGM²-Modelle deren Speichern und Laden an. Dies erfolgte über Dateien des Typs „.3lgm“ bzw. „.z3lgm“, was allerdings für Konstruktionspatterns nicht anwendbar ist, da diese auf dem 3LGM²-CF basieren und potentiell eine andere Dateistruktur erfordern. Dennoch müssen auch sie gespeichert und geladen werden können, was ein eigenes Speicherformat erforderlich macht. Es ist hier allerdings nicht das Ziel, ein solches konkret zu spezifizieren. Vielmehr soll dieses bei der Implementierung selbst festgelegt werden, um gegebenenfalls auf bereits intern vorhandene Routinen zurückgreifen zu können:

3LGM²-Dateien werden im XML-Format abgelegt. Es wäre daher denkbar, dieses Format auch für Patterns einzusetzen. Darüber hinaus bestehen eine Vielzahl an Analogien zwischen 3LGM²-CF und 3LGM²: Sowohl PatternObjects als auch Elemente und Beziehungen besitzen Namen und Beschreibung als Attribut. Des Weiteren haben die Beziehungen im 3LGM² analog zu PatternRelations, DirectConstraints und CreationLinks Start- und End-Attribute, welche die verbundenen Elemente bzw. PatternObjects kennzeichnen. Daher wäre es möglich, für das Speichern von Patterns auf schon existierende Methoden aufzubauen. Grundsätzlich sollte dabei ein Pattern durch genau eine Datei

repräsentiert werden. Diese erfasst alle Attribute des Patterns selbst sowie alle SubPatterns, PatternObjects, DirectConstraints und CreationLinks einschließlich deren Attribute. Die Menüpunkte Speichern und Laden können so auf den Kontext der Konstruktionspatterns übertragen werden.

Des Weiteren war es in diesem Menü über den Punkt Beschreibung möglich, ein Fenster zur Eingabe von Beschreibungen für das Modell und alle Teilmodell zu öffnen. Dies sollte jetzt analog für Patterns funktionieren. Unter dem im Eingabefenster dargestellten Tab <Alle Elemente> ist dabei das Attribut Pattern.*description* abzubilden. Analog zum Prinzip der Teilmodelle soll darüber hinaus für jedes SubPattern ein weiterer Tab erscheinen, unter dem dann jeweils das SubPattern.*description*-Attribut dargestellt wird.

Bearbeiten

In diesem Menü sind standardmäßig Funktionen wie Kopieren, Ausschneiden, Rückgängigmachen etc. enthalten, welche in gleicher Weise für das Pattern anwendbar sein sollen. Die einzigen Ausnahmen sind die beiden Löschfunktionen, welche durch Löschen aus Pattern bzw. Löschen aus Subpattern ersetzt werden müssen. In Analogie zum Prinzip von Modell und Teilmodell in der bekannten Anwendung des Tool3LGM² löschen diese nun PatternObjects, DirectConstraints oder CreationLinks aus einem Pattern bzw. SubPattern.

Ansicht

Das Ansicht-Menü wird auf die Punkte Symbolleisten und Modell Browser reduziert. Eine Ebenendarstellung oder Matrix-Sicht wird nicht benötigt.

Einfügen

Dieses Menü darf komplett entfallen.

Optional wäre es noch denkbar, die (alleinige) Funktion Neuen PatternNode anlegen zur Verfügung zu stellen.

Format

In der üblichen Anwendung des Tool3LGM² diente das Formatmenü dem Ändern der Darstellung von Modellelementen sowie deren Anordnung und Ausrichtung zueinander. Im Zentrum der Darstellung von Konstruktionspatterns hingegen befinden sich PatternObjects, deren Layout weitestgehend vorgegeben ist. Damit sollte dieses Menü nur noch das Untermenü «Element Layout» mit den Punkten Schriftart, Farbe ändern, Farbe zurücksetzen, Schriftart zurücksetzen und Alles zurücksetzen beinhalten sowie das vollständige Untermenü «Elementausrichtung». «Reihenfolge» entfällt komplett, da ein überlappendes Darstellen von PatternObjects nicht sinnvoll ist.

Teilmodelle

Dieses Menü sollte vollständig ersetzt werden durch ein «SubPattern»-Menü, welches in Anlehnung an das ursprüngliche Menü die Funktionen Neues SubPattern, SubPattern löschen und SubPattern umbenennen anbietet. Das heißt, die Möglichkeiten für Teilmodelle werden auf die SubPatterns übertragen. Dabei ist zu beachten, dass jetzt das Eingabefenster für das Umbenennen das Attribut SubPattern.*description* abbildet.

Analyse

Das Analyse-Menü sollte komplett deaktiviert werden, da bisher nichts Derartiges für Patterns vorgesehen ist und im Moment nur im Kontext der Modellierung von 3LGM²-Modellen Anwendung findet.

Optionen

Dieses Menü kann teilweise unverändert bleiben, mit folgenden Ausnahmen:

- «Allgemein»/Beim Suchen übergeordnete Elemente berücksichtigen und
- «Allgemein»/Ist-Teil-von-Beziehungen hierarchisch anzeigen

können entfallen sowie:

- «Modell-Browser»/Benutzerdefinierte Eigenschaften anzeigen

und

- «Grafik»/Unbenutzte Bausteinschnittstellen anzeigen,
- «Grafik»/Automatische Farbzuweisung für Konfigurationslinien,
- «Grafik»/Unterelemente automatisch verschieben,
- «Grafik»/Vergröberung anzeigen.

Extras

Dieses Menü kann vollständig entfallen.

Fenster

Dieses Menü kann unverändert übernommen werden.

?

Dieses Menü kann unverändert übernommen werden.

Werkzeugleiste

Hier sollten die Buttons Einzel/Drei-Ebenen-Ansicht, Fachliche Ebene, Logische Werkzeugebene und Physische Werkzeugebene deaktiviert werden, da bei Konstruktionspatterns keine unterschiedlichen Ebenen für die Darstellung existieren. Alle anderen Buttons können aktiv bleiben.

4.5.1.2 Modell Browser

Der Modell Browser im linken Bereich des Tool3LGM² dient der erleichterten Navigation innerhalb eines Modelles. Dazu werden darin Elemente eines Modelles bzw. Teilmodelles in Baumform dargestellt und ihren Klassen im 3LGM² untergeordnet. Ähnlich sollte nun die Anwendung im Kontext der Konstruktionspatterns erfolgen, allerdings mit dem Unterschied, dass jetzt statt Modellelementen, PatternObjects angezeigt werden und diese nicht nach Klasse, sondern nach *targetClassifier*-Attribut sortiert werden. Problematisch hierbei ist allerdings, dass bisher nur Klassen und keine Assoziationen angezeigt wurden. Der Modell Browser soll daher wie folgt angepasst werden:

- Initial – also bei einem neu angelegten Konstruktionspattern – werden in der bisher bekannten Form die Ebenen und die jeweils dazugehörigen Klassen dargestellt.
- Beim Erstellen eines PatternNode wird dieser entsprechend seinem *targetClassifier*-Attribut der jeweiligen Klasse untergeordnet.
- Eine Assoziation wird erst dann im Modell Browser angezeigt, wenn wenigstens eine PatternRelation existiert, die sie als *targetClassifier*-Attribut besitzt. Alle PatternRelations werden dann auf diese Weise ihrer Assoziation untergeordnet.

Hierdurch ist nun die Navigation durch das Pattern bei bester Übersichtlichkeit gewährleistet.

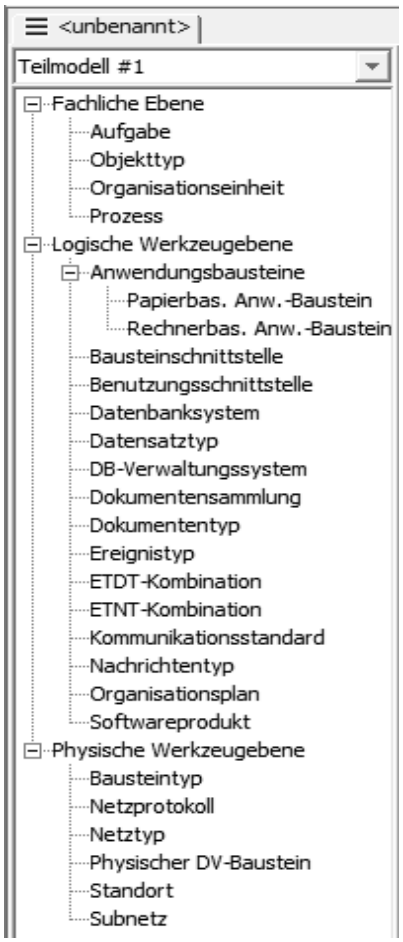

Abbildung 28 – Modell Browser im Tool3LGM²

4.5.1.3 Grafische Ansicht

Die grafische Ansicht diente vor allem der Darstellung der drei Ebenen mit ihren jeweiligen Elementen. Da für das Construction Framework diese Ebenen irrelevant sind, sollte es hier nur eine einzige Zeichenfläche geben, in der nun statt Modellelementen des 3LGM² PatternObjects des 3LGM²-CF angezeigt werden. Jede dieser grafischen Ansichten korrespondiert dabei zu einem SubPattern. Dem Patternmodellierer soll hierbei die Möglichkeit gegeben werden, neue PatternNodes, PatternRelations sowie DirectConstraints und CreationLinks zu erzeugen. Dies soll analog zum Anlegen von Modellelementen geschehen. PatternNodes sollten dabei wie 3LGM²-Klassen instanziiert werden können, das heißt, etwa über einen Menüpunkt Neuer PatternNode innerhalb des Kontextmenüs, welches bei einem Rechtsklick auf die Zeichenfläche erscheint (Abbildung 29).

Abbildung 29 – Kontextmenü für das Anlegen eines neuen PatternNode

Bei Auswahl dieses Menüpunktes sollte sich dann ein Dialogfenster öffnen, in dem Name und Zielklasse des PatternNodes auszuwählen sind (Abbildung 30).

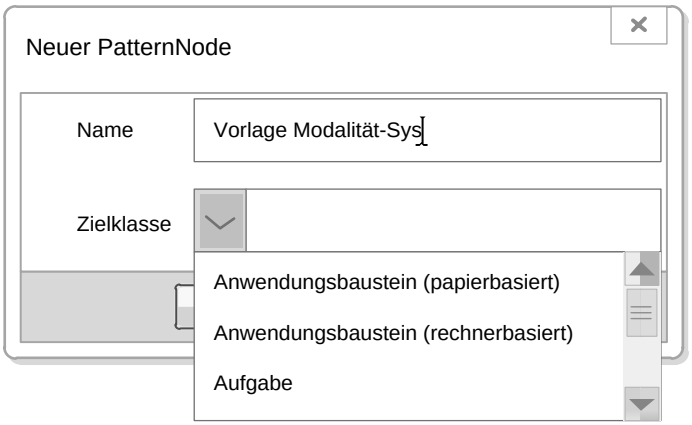

Abbildung 30 – Dialog für das Anlegen eines neuen PatternNode

Diese werden dann wie folgt als Werte der Klassenattribute für den PatternNode gesetzt:

Bezeichnung	*Klassenattribut*
Name	PatternObject.*name*
Zielklasse	PatternObject.*targetClassifier*

Nachdem die Eingaben akzeptiert wurden, soll auf der Zeichenfläche ein entsprechender neuer PatternNode erscheinen, welcher mit der in 4.4.3.2 geforderten grafischen Repräsentation übereinstimmt (Ein Beispiel hierzu wird im Folgenden noch gezeigt). Für ihn kann nun eine Kurzbeschreibung angelegt werden. Dazu sollte beim Rechtsklick auf den PatternNode im Kontextmenü die Funktion Kurzbeschreibung ändern angeboten werden (Abbildung 31), welche ein entsprechendes Eingabefenster erscheinen lässt (Abbildung 32). Die dort gesetzte Kurzbeschreibung korrespondiert dabei zum Rückgabewert der SubPattern.*getShortDescription(PatternObject)*-Methode für den betreffenden PatternNode innerhalb des SubPattern. Außerdem sollte in diesem Menü die Funktion Bearbeiten existieren, welche den in Abbildung 33 zu sehenden Eigenschaftendialog öffnet.

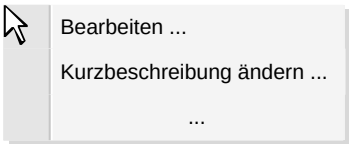

Abbildung 31 – Kontextmenü für das Editieren von PatternObjects

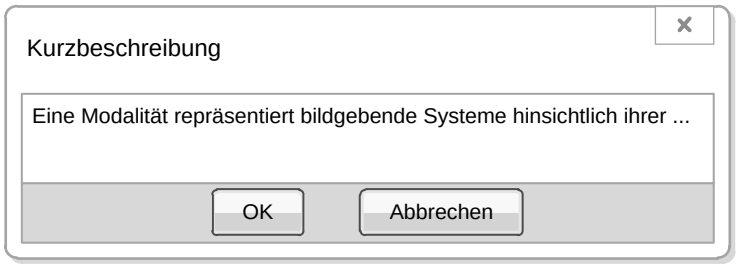

Abbildung 32 – Dialogfenster für die Eingabe von Kurzbeschreibungen für PatternObjects

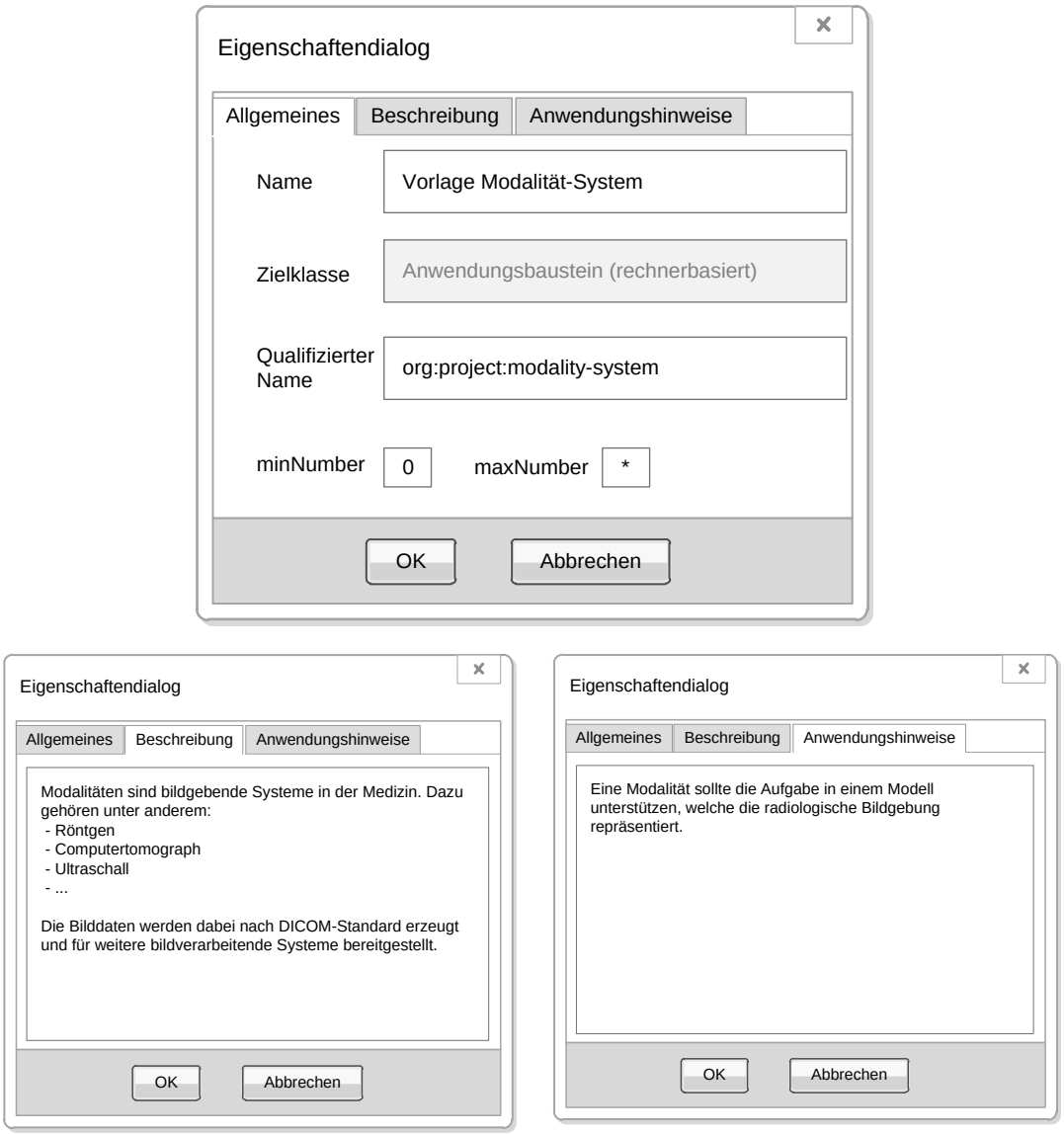

Abbildung 33 – Eigenschaftendialog für PatternObjects

In diesem Dialog, können nun weitere Attribute des PatternNode editiert werden. Diese korrespondieren dabei jeweils zu einem Klassenattribut:

Bezeichnung	*Klassenattribut*
Qualifizierter Name	PatternObject.*qualifiedName*
minNumber	PatternObject.*minNumber*
maxNumber	PatternObject.*maxNumber*
Beschreibung	PatternObject.*description*
Anwendungshinweise	PatternObject.*applicationHint*

Anmerkung: PatternRelations besitzen dieselben Attribute wie PatternNodes, welche sie gemeinsam von der Klasse PatternObject erben. Darüber hinaus können für sie auch die SubPattern-abhängigen Kurzbeschreibungen angelegt werden, sodass die eben beschriebenen Möglichkeiten für das Editieren der PatternNode-Eigenschaften ebenfalls für sie Anwendung finden können. Auf Besonderheiten im Umgang mit PatternRelations wird später noch näher eingegangen.

Nach dem Anlegen einiger solcher PatternNodes, könnte nun unter Verwendung des Beispiels aus Abbildung 26 die Zeichenfläche etwa wie folgt aussehen:

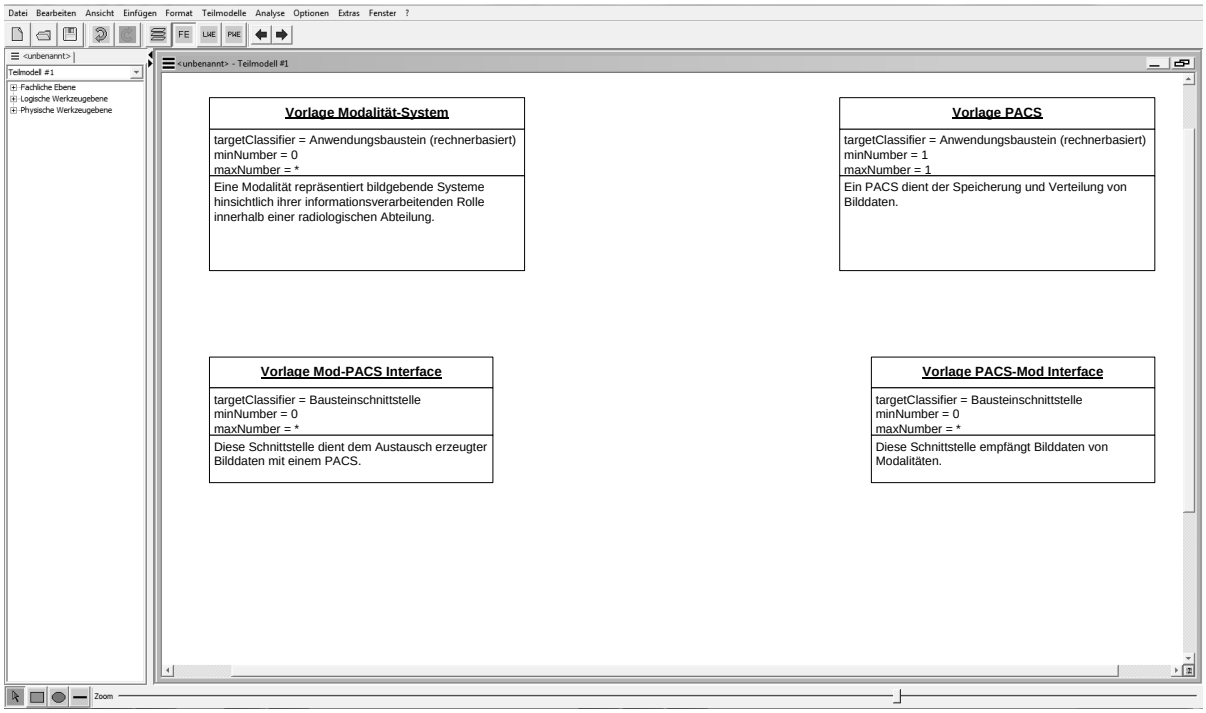

Abbildung 34 – Beispieldarstellung eines Subpatterns im Tool3LGM²

Nun ist es möglich, PatternNodes, wie sie in Abbildung 34 dargestellt sind, durch PatternRelations zu verbinden. Dies erfolgt, indem zwei PatterNodes selektiert und dann das Kontextmenü erneut geöffnet wird. Darin sollten nun die Punkte Neue PatternRelation, Neuer DirectConstraint und Neuer Creation-Link erscheinen (Abbildung 35).

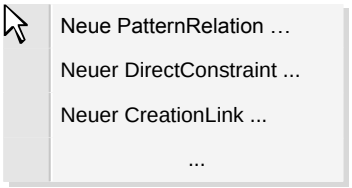

Abbildung 35 – Kontextmenü für das Anlegen von Beziehungen zwischen PatternObjects

Beim Anlegen einer neuen PatternRelation sollte sich dazu ein Dialog öffnen, in dem Name und Zielassoziation festgelegt werden können (Abbildung 36). „Source“ und „Target“ entsprechen dabei den ausgewählten PatternNodes.

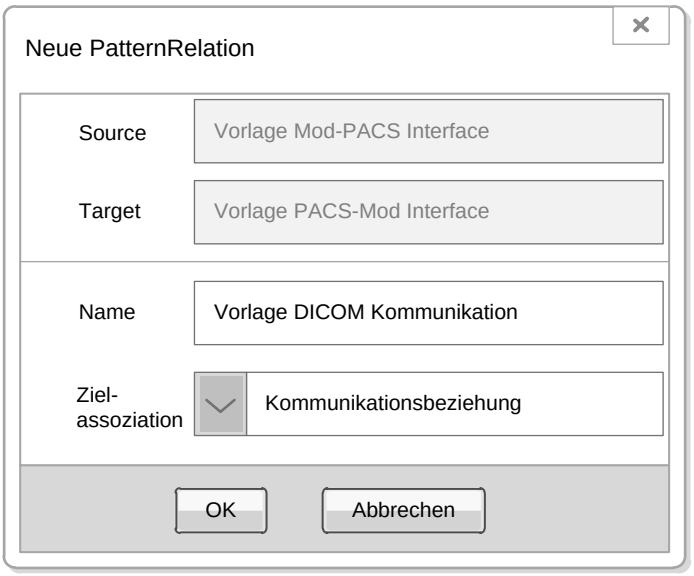

Abbildung 36 – Dialog für das Anlegen einer neuen PatternRelation

Die hier angegebenen Werte werden wie folgt auf die Klassenattribute der PatternRelation abgebildet:

Bezeichnung	*Klassenattribut*
Name	PatternObject.*name*
Zielassoziation	PatternObject.*targetClassifier*
Source	PatternRelation.*source*
Target	PatternRelation.*target*

Soll stattdessen ein DirectConstraint eingefügt werden, öffnet sich ein Dialog, der die Eingabe von Assoziation und Kardinalitäten erlaubt (Abbildung 37).

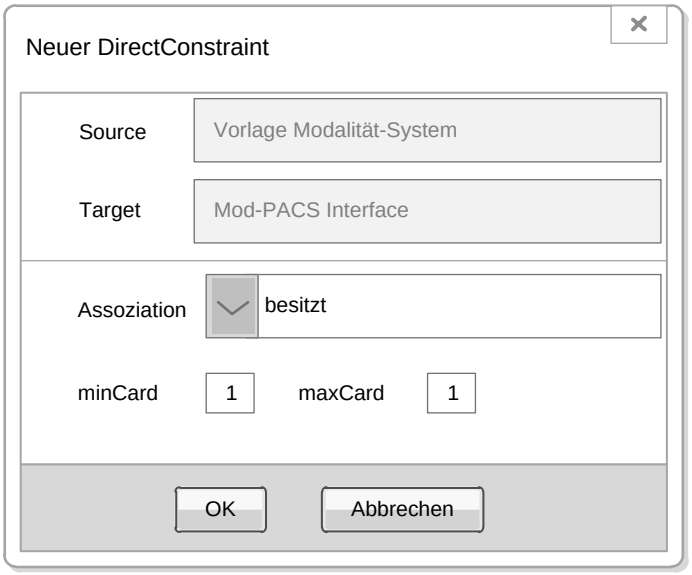

Abbildung 37 – Dialog für das Erzeugen eines DirectConstraint

Die Werte im Dialogfenster werden dabei wie folgt abgebildet:

Bezeichnung	*Klassenattribut*
Assoziation	DirectConstraint.*association*
minCard	DirectConstraint.*minCard*
maxCard	DirectConstraint.*maxCard*
Source	DirectConstraint.*constraintSource*
Target	DirectConstraint.*constraintTarget*

Im Fall, dass ein CreationLink angelegt werden soll, muss der Dialog nur die Auswahl der Assoziation anbieten (Abbildung 38).

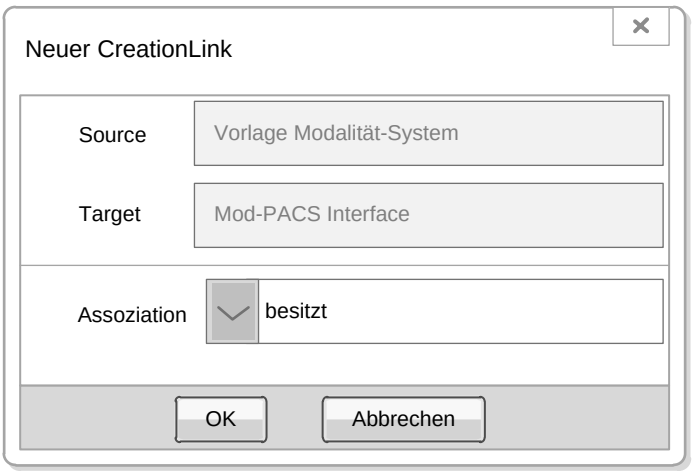

Abbildung 38 – Dialog für das Anlegen eines CreationLink

Die hier angegebenen Werte werden wie folgt den Klassenattributen zugeordnet:

Bezeichnung	*Klassenattribut*
Assoziation	CreationLink.*association*
Source	CreationLink.*creationSource*
Target	CreationLink.*creationTarget*

Um erneut zum Beispiel aus Abbildung 26 zurückzukehren, würde sich nach dem Anlegen entsprechender PatternRelations, DirectConstraints und CreationLinks die Anzeige in der Zeichenfläche, wie etwa in Abbildung 39 zu sehen, gestalten.

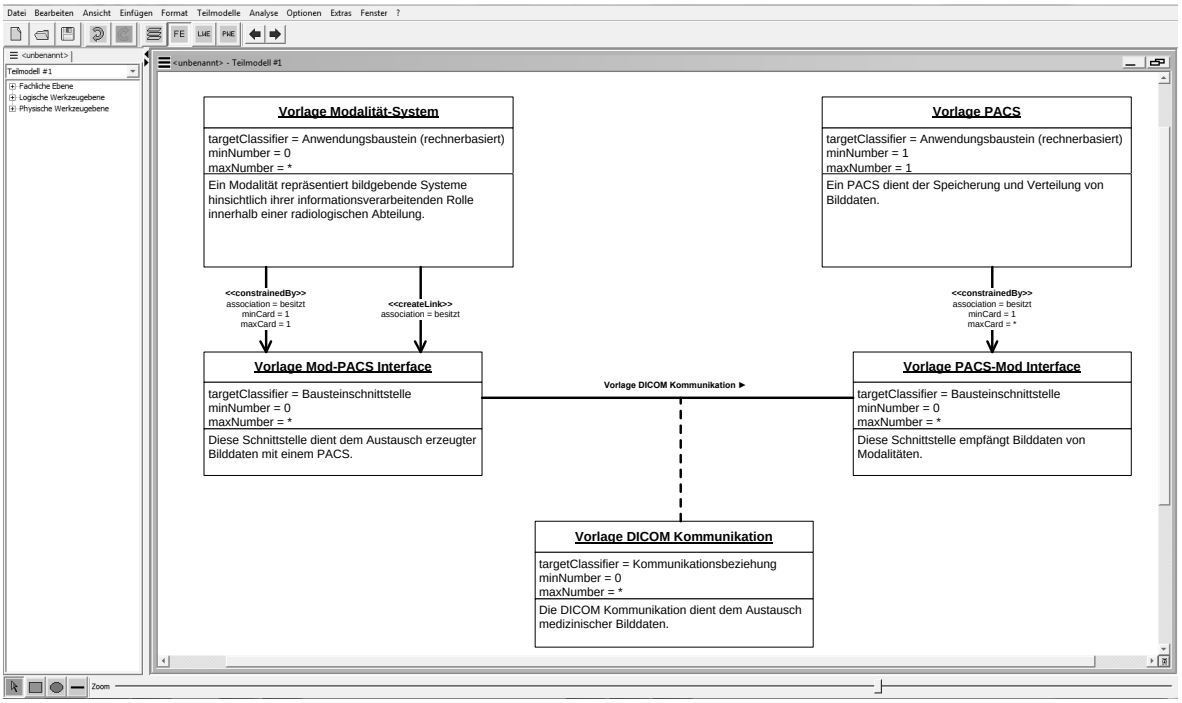

Abbildung 39 – Beispieldarstellung eines Subpatterns mit Beziehungen der PatternObjects im Tool3LGM²

Analog zu den PatternNodes ist es auch für PatternRelations möglich, eine Beschreibung anzulegen. Hier können dann etwa wichtige Details bezüglich der DICOM-Kommunikation hinterlegt werden, wie es in Abbildung 40 dargestellt ist.

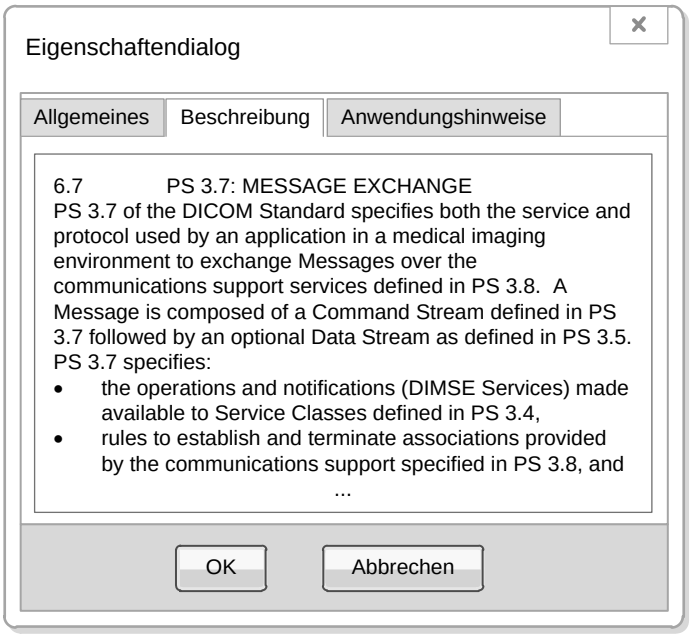

Abbildung 40 – Eigenschaftendialog zur „Vorlage DICOM Kommunikation“

Dieses Pattern kann nun, wie oben beschrieben, abgespeichert werden und ist somit bereit für den Einsatz in 3LGM²-Modellen. Bevor darauf näher eingegangen wird, sollen abschließend noch die Modifikationen an der unteren Werkezugleiste im Tool3LGM² vorgestellt werden.

4.5.1.4 Untere Werkzeugleiste

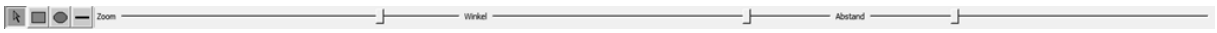

Abbildung 41 – Untere Werkzeugleiste im Tool3LGM²

Die Werkzeugleiste im unteren Bereich bietet diverse Mauswerkzeuge und Darstellungsparameter für die grafische Ansicht. Im Kontext der Konstruktionspatterns genügt es hier, folgende Funktionen anzubieten:


Auswahlwerkzeug


Neuer PatternNode


Neue PatternRelation

Zoom

Abbildung 42 – Funktionen der unteren Werkzeugleiste für das 3LGM²-CF

4.5.2 Patternanwendung

Ist ein Pattern einmal angelegt, kann dieses in beliebig vielen Modellen zur Konstruktion der Informationssysteme eingesetzt werden. Als Ausgangssituation ist jetzt ein leeres oder auch nichtleeres 3LGM²-Modell im Tool3LGM² geöffnet, in dem ein oder mehrere Patterns zur Anwendung kommen sollen.

4.5.2.1 Patterns laden

Das Laden eines Konstruktionspatterns kann beispielsweise als Funktion Pattern laden in der Menüleiste angeboten werden, etwa im Menü «Extras». Hierbei sollte sich ein Auswahldialog öffnen, in dem die Datei des Patterns ausgewählt werden kann. Das Tool3LGM² liest daraufhin, entsprechend der konkreten Umsetzung des Speicherformats, die Datei aus und legt das Pattern als Laufzeitinstanz des 3LGM²-CF an.

4.5.2.2 Elemente als Ableitungen eines PatternNode instanziieren

Neben dem bekannten Anlegen von Elementen und Beziehungen muss es nun auch möglich sein, derartige Instanzen als Ableitungen von PatternNodes und PatternRelations aus diesem Pattern zu erstellen. Um dies zu gewährleisten, sollten zunächst jeder Klasse die PatternNodes zugeordnet werden, deren *targetClassifier*-Attribut mit ihr übereinstimmt. Um zum Beispiel der DICOM-Kommunikation aus Abbildung 39 zurückzukehren, wären es hier für die Klasse Anwendungsbaustein (rechnerbasiert) die PatternNodes „Vorlage Modalität-System“ sowie „Vorlage PACS“. Nun sollte nicht nur das Instanziieren eines einfachen Anwendungsbausteines möglich sein, sondern darüber hinaus auch das Anlegen dieser PatternNodes. Etwa im Kontextmenü in der grafischen Ansicht könnte sich dies wie folgt gestalten:

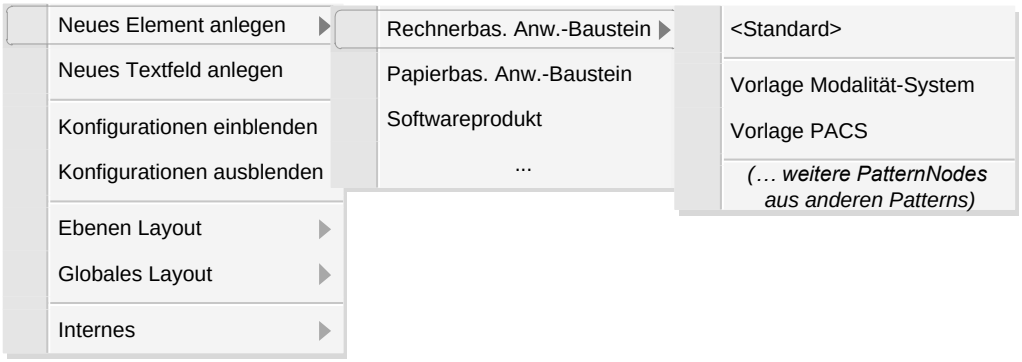

Abbildung 43 – Kontextmenü für das Anlegen von rechnerbasierten Anwendungsbausteinen als Ableitungen von PatternNodes

In Rückblick auf das Beispielmodell der radiologischen Bildgebung und –speicherung, wie es in 2.2 zu sehen war, soll nun die Unterstützung der Konstruktion dessen Logischer Werkzeugebene durch das oben genannte Pattern der DICOM-Kommunikation dargestellt werden.

Zunächst werden ein MRI System und ein CT System als Ableitung von „Vorlage Modalität-System“ erzeugt, durch Auswahl des PatternNode im obigen Kontextmenü. Ebenso erfolgt dann die Instanziierung eines PACS. Entsprechend der im Pattern definierten Abbildungsregeln werden diese jeweils als rechnerbasierter Anwendungsbaustein angelegt, was sich im Tool3LGM², wie in Abbildung 44 zu sehen, gestalten könnte.

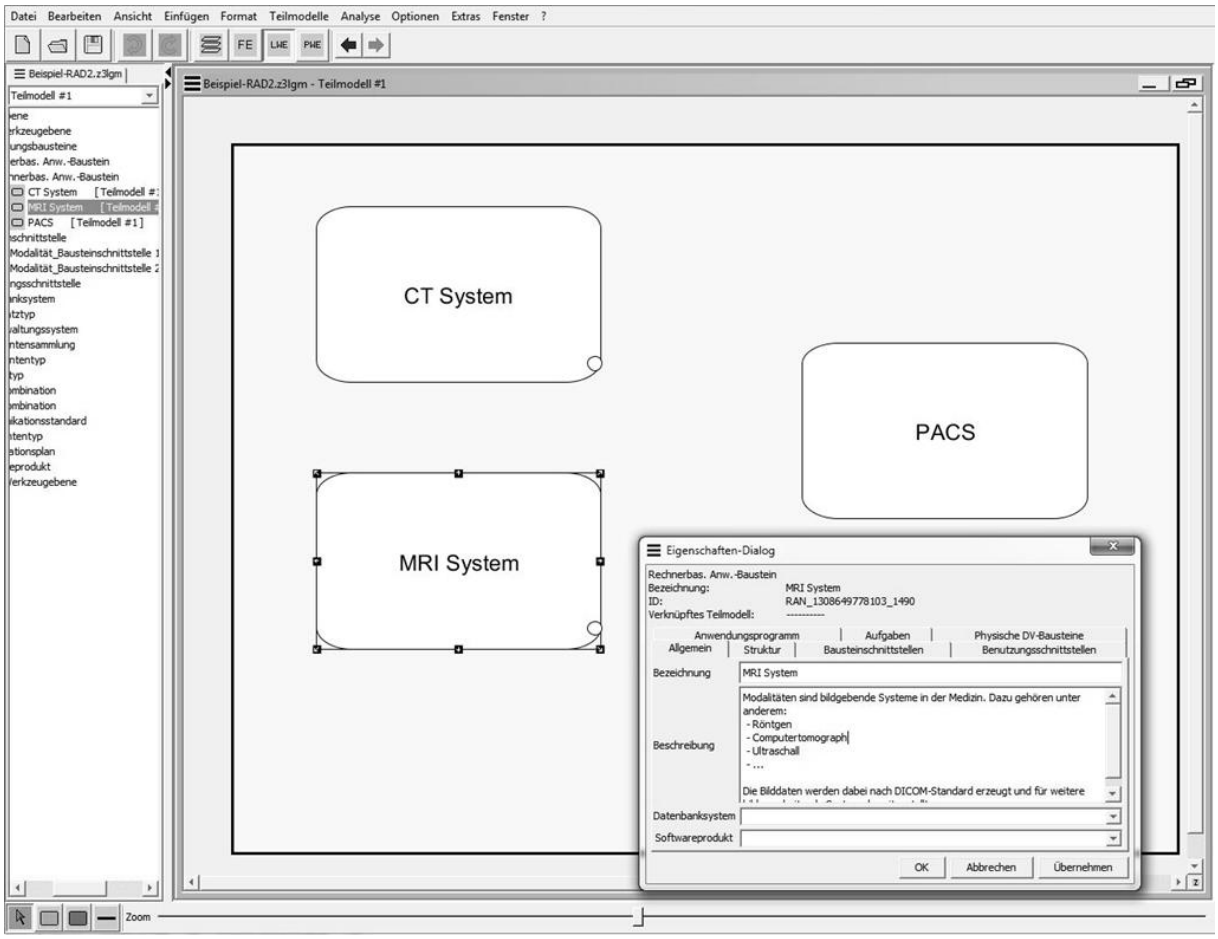

Abbildung 44 – Logische Werkzeugebene mit Anwendungsbausteinen der Radiologie als Ableitungen aus einem Pattern

Es ist dabei Aufgabe des Tool3LGM², für jedes Element und jede Beziehung in einem Modell zu hinterlegen, von welchem PatternObject sie abgeleitet wurden. Intern könnte dies etwa über ein Klassenattribut erfolgen, welches auf das PatternObject im geladenen Pattern referenziert. Später wird dies von Bedeutung, wenn es gilt, die im Pattern definierten Constraints zu überprüfen.

Da CT und MRI System, wie eben erwähnt, von „Vorlage Modalität-System“ abgeleitet sind, wurde für sie bereits der vorgefertigte Beschreibungstext aus dem *description*-Attribut dieses PatternNode übernommen, wie er im Eigenschaftendialog in Abbildung 33 definiert wurde. Dieser ist nun wiederum in den Eigenschaftendialogen der davon abgeleiteten Elemente (hier: CT und MRI System) verfügbar und gibt nun sowohl Modellierern als auch Modellnutzern bereits allgemeine Informationen zu den Modalitäten und kann nach Bedarf noch ergänzt werden (Abbildung 44). Des Weiteren sind für sowohl CT als auch MRI – entsprechend dem für das PatternNode definierten CreationLink – automatisch Bausteinschnittstellen angelegt worden, welche von „Vorlage Mod-PACS“ Interface abgeleitet sind und nun für die Kommunikation mit dem PACS bereitstehen. Zuvor muss dieses allerdings selbst noch mit geeigneten Schnittstellen versehen werden. Dazu ist jetzt im Kontextmenü dieses Elementes das Anlegen von „Vorlage PACS-Mod Interfaces“ möglich:

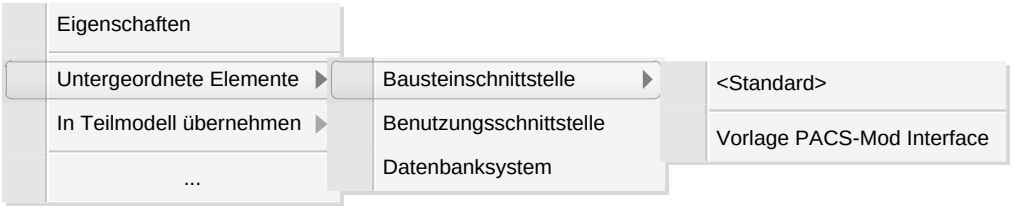

Abbildung 45 – Anlegen untergeordneter Elemente als Ableitungen aus einem Pattern

Aufgrund des DirectConstraint, der für „Vorlage PACS“ definiert wurde, ist hier nur die Auswahl zwischen Standard-Bausteinschnittstelle und „Vorlage PACS-Mod Interface“ möglich. Nach dem Anlegen des Interface erscheint dieses nun als Bausteinschnittstelle für das PACS in der grafischen Ansicht.

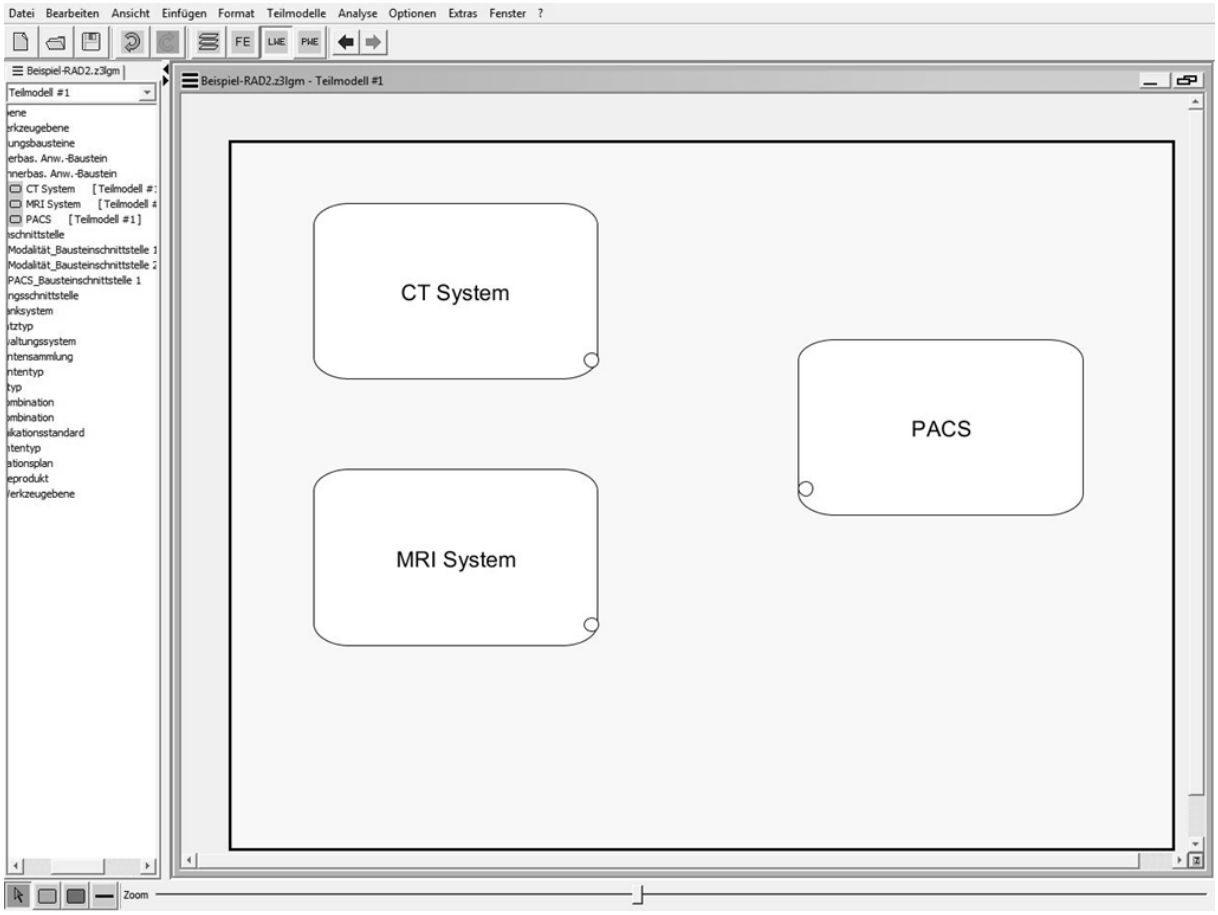

Abbildung 46 – PACS mit abgeleiteter Bausteinschnittstelle

4.5.2.3 Verknüpfungen als Ableitungen einer PatternRelation instanziieren

Möchte man nun wissen, wie CT und MRT mit dem PACS Bilddaten austauschen sollen, kann man hierzu wie im üblichen Fall beginnen, eine Kommunikationsbeziehung zwischen ihnen anzulegen. Dies geschieht durch Auswahl der zu verbindenden Schnittstellen und dem Öffnen des Kontextmenüs. Hier wird nun allerdings neben dem Anlegen einer standardmäßigen Kommunikationsbeziehung bereits eine vorgefertigte DICOM Kommunikation aus dem Pattern angeboten:

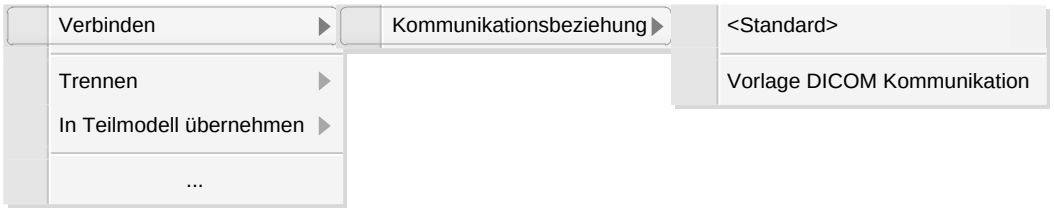

Abbildung 47 – Kontextmenü für das Anlegen abgeleiteter Kommunikationsbeziehungen

Diese darf nur zwischen diesen speziellen Bausteinschnittstellen angelegt werden, womit dem Modellierer der richtige Weg hin zu einem Pattern-konformen Modell gewiesen wird.

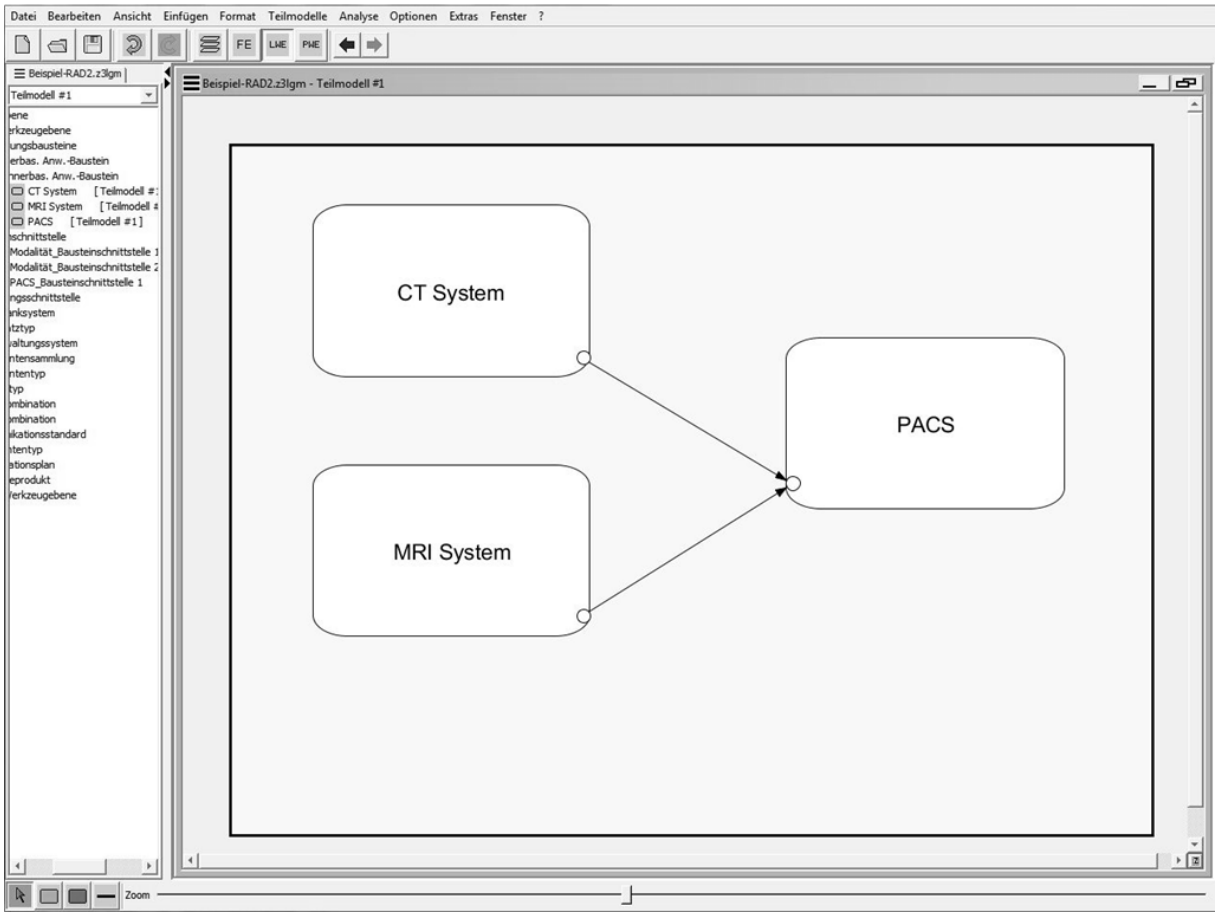

Abbildung 48 – Vollständige Logische Werkzeugebene zum Beispiel der Radiologie aus 2.2 nach Einsatz des DICOM-Konstruktionspatterns aus Abbildung 39

Die DICOM Kommunikation wird dabei als normale Kommunikationsbeziehungen dargestellt (Abbildung 48). Diese übernehmen analog zu Ableitungen von PatternNodes, die Beschreibung der jeweiligen PatternRelation, wie es im Eigenschaftendialog in Abbildung 49 gezeigt ist.

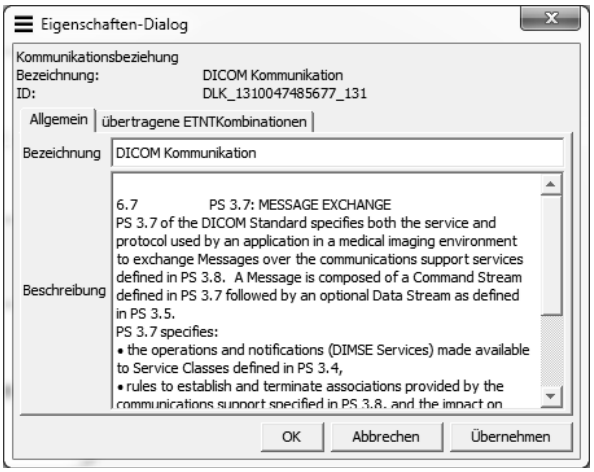

Abbildung 49 – Eigenschaftendialog einer Kommunikationsbeziehung im Tool3LGM²

Für den Modellierer entfällt damit der Aufwand, eine geeignete Kommunikation identifizieren und deren Grundlagen erarbeiten zu müssen.

4.5.2.4 Anwendungshinweise anzeigen

Über Anwendungshinweise kann nun der Modellierer noch dahingehend unterstützt werden, wie Instanzen, die von einem PatternObject abgeleitet wurden, mit anderen Instanzen in einem Modell in Beziehung treten sollen, welche nicht innerhalb des Patterns repräsentiert sind. Derartige Hinweise können für jedes PatternObject im *applicationHint*-Attribut hinterlegt werden.

Eine Möglichkeit, den Zugriff auf diese zu erlangen, ergibt sich über das Kontextmenü von Elementen und Verknüpfungen, welches um den Punkt Anwendungshinweise… ergänzt werden kann:

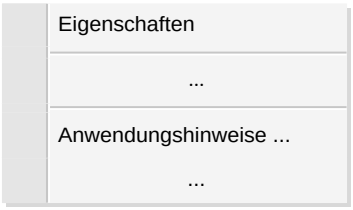

Abbildung 50 – Auswahl der Anwendungshinweise im Kontextmenü

Daraufhin öffnet sich ein Fenster, welches den Anwendungshinweis des PatternObject wiedergibt, von dem das aktuell ausgewählte Element bzw. die aktuell ausgewählte Verknüpfung abgeleitet ist, wie es in Abbildung 51 dargestellt wird.

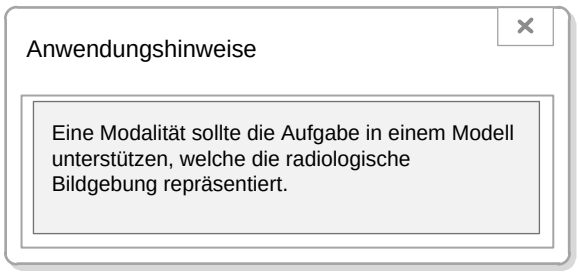

Abbildung 51 – Anzeige von Anwendungshinweisen über ein Fenster

Hier zu sehen ist der Anwendungshinweis von CT oder auch MRI System, wie er für den dazugehörigen PatternNode „Vorlage Modalität-System“ definiert wurde (siehe Abbildung 33).

Eine weitere Möglichkeit für den Zugriff auf *applicationHint*-Attribute ergäbe sich im Pattern selbst. Dazu könnte zum Beispiel das «Extras»-Menü in der Menüleiste um den Punkt Pattern anzeigen ergänzt werden, woraufhin eine (nicht editierbare) Ansicht des Patterns während der Modellierung geöffnet wird. Der Modellierer erreicht dann durch das Bewegen des Cursors über ein PatternObject, dass ein Tooltip angezeigt wird, der den Anwendungshinweis für ihn wiedergibt, wie es in Abbildung 52 am Beispiel der „Vorlage Modalität-System“ zu sehen ist.

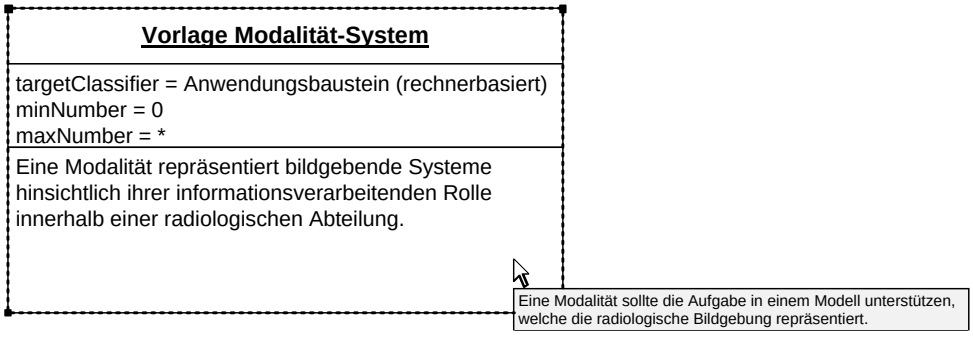

Abbildung 52 – Darstellung von Anwendungsweisen über Tooltips

Die Verwendung von Anwendungshinweisen birgt zwei wesentliche Vorteile: Zum einen ist der Patternmodellierer so in der Lage, strukturell nicht erfassbare Zusammenhänge zumindest textuell abbilden zu können und zum anderen kann der Modellierer darüber sehr leicht ermitteln, welche Elemente und Verknüpfungen noch anzulegen sind, um dem zugrundeliegenden Entwurfswissen zu entsprechen.

4.5.2.5 Ableiten bestehender Elemente und Verknüpfungen

Häufig wird es der Fall sein, dass ein Pattern in ein bereits bestehendes – also nichtleeres – Informationssystemmodell integriert werden muss. Dazu sollte auch alternativ zum Neuanlegen von Elementen und Verknüpfungen, bestehenden Instanzen ein PatternObject zugewiesen werden können. Hierzu bedarf es erneut einer geeigneten Modifikation des Kontextmenüs, wie es in Abbildung 53 dargestellt ist. Dieses sollte sich in diesem Fall bei Auswahl einer Kommunikationsbeziehung öffnen.

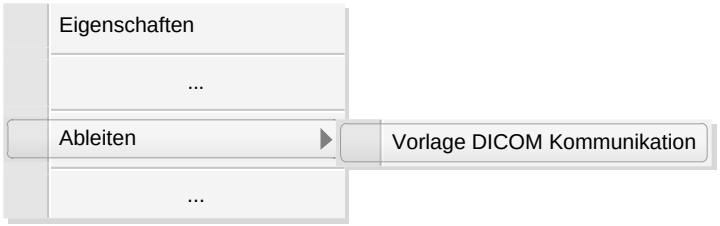

Abbildung 53 – Ableiten einer bestehenden Verknüpfung von einer PatternRelation

Nachdem das gewünschte PatternObject ausgewählt wurde, sind nun zunächst noch die Beschreibungstexte abzugleichen – der bereits bestehende von Element bzw. Verknüpfung mit dem des PatternObject – wie es in Abbildung 54 beispielhaft dargestellt ist. Dies ist durch den Modellierer selbst durchzuführen.

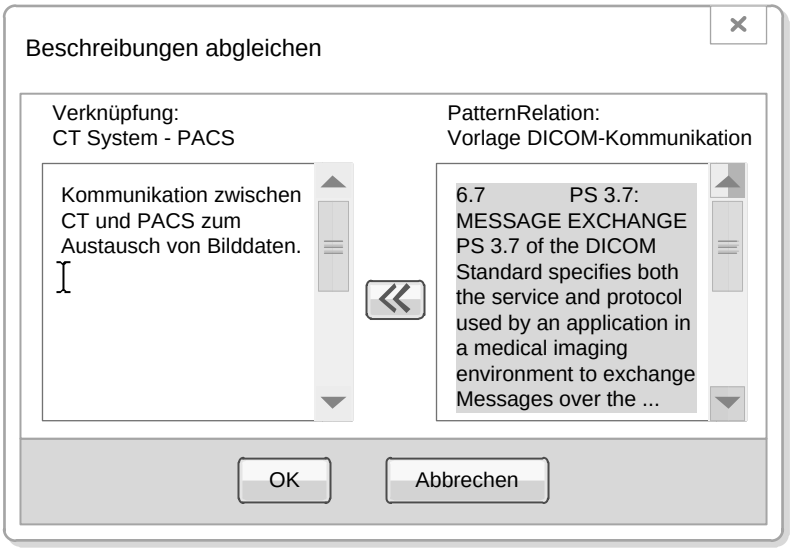

Abbildung 54 – Abgleich von Beschreibungen beim Ableiten einer existierenden Kommunikationsbeziehung von „Vorlage DICOM Kommunikation“

Die Kommunikationsbeziehung wird daraufhin den gewählten Text als Beschreibung erhalten, wodurch jetzt in diesem Beispiel nähere Informationen zu DICOM ins Modell einfließen und von nun ab

jederzeit abrufbar sind. Sinnvoll ist dies, wenn etwa im Modell schon PACS und Modalitäten angelegt und mit Standard-Kommunikationsbeziehungen verbunden worden sind, im Nachhinein allerdings die Kommunikation durch DICOM konkretisiert werden soll.

Analog sollte dieses Verfahren natürlich auch für PatternNodes ausführbar sein.

4.5.3 Patternkonformität eines Modelles

Die eben geschilderten Möglichkeiten für das Anwenden eines Patterns können nun noch erweitert werden durch das Prüfen der Einhaltung von Einschränkungen und Vorgaben innerhalb eines Modelles.

Definition (Patternkonformität):

Ein Modell ist in Konformität mit einem Pattern, wenn gilt:

1. *Alle Elemente und Verknüpfungen im Modell, welche von einem PatternObject abgeleitet wurden, sind Instanzen des jeweiligen PatternObject.targetClassifier-Attributes.*

2. *Das erste bzw. zweite Ende jeder Verknüpfung, die von einer PatternRelation abgeleitet wurde, ist jeweils wiederum vom source- bzw. target-Ende dieser PatternRelation abgeleitet.*

3. *Für jedes PatternObject gibt es im Modell wenigstens PatternObject.minNumber- und höchstens PatternObject.maxNumber-viele Instanzen, die von ihm abgeleitet sind.*

4. *Jede Instanz im Modell, welche von einem PatternObject abgeleitet wurde, ist für jeden DirectConstraint, für den dieses PatternObject als DirectConstraint.constraintSource gesetzt ist, über die jeweilige DirectConstraint.association nur mit den Instanzen verknüpft, für die gilt:*
   - *Sie sind von DirectConstraint.constraintTarget abgeleitet und es bestehen wenigstens DirectConstraint.minCard- und höchstens DirectConstraint.maxCard-viele solcher Verknüpfungen zu ihnen oder*
   - *Die Instanzen sind von keinem PatternObject abgeleitet (Damit sind auch alle PatternObjects gemeint, die Teil eines anderen Patterns sind).*

Aus den Kriterien für die Konformität eines Modelles mit einem Pattern ergibt sich nun die Notwendigkeit, diese im Tool3LGM² zu überprüfen und sicherzustellen:

Die Einhaltung von Kriterium 1 kann bereits durch die oben geschilderte Anwendung eines Konstruktionspatterns gewährleistet werden, indem das Ableiten von einem PatternObject nur dann möglich ist, wenn der *targetClassifier* mit der jeweiligen Klasse oder Assoziation im 3LGM² übereinstimmt. Analog lässt sich Kriterium 2 sicherstellen, indem das Anlegen einer Verknüpfung als Ableitung einer PatternRelation nur dann angeboten wird, wenn beide zu verbindenden Elemente Ableitungen vom *source*- bzw. *target*-Ende der PatternRelation sind und die Verknüpfung als eine Instanz ihres *association*-Attributes erstellt werden soll.

Beide vorangegangen Kriterien stellten Invarianten dar, was bedeutet, dass sie zu jedem Zeitpunkt gelten müssen. Die beschriebenen Möglichkeiten für die Anwendung eines Konstruktionspatterns garantieren dabei deren permanente Einhaltung. Im Gegensatz dazu stellen Punkt 3 und Punkt 4 optionale Kriterien dar. Betrachtet man etwa den Fall eines leeren Modelles, für das ein Pattern geladen wird, welches PatternObjects beinhaltet, für die PatternObject.*minNumber* > 0 gilt, ist dieses pauschal inkonsistent. Ebenso sind alle DirectConstraints verletzt für die DirectConstraint.*minCard* > 0 gilt. Hieraus resultieren zwei Aufgaben für das Tool3LGM²:

1.) Zum einen soll dem Modellierer jederzeit eine Übersicht darüber zugänglich sein, welche Inkonsistenzen momentan im Modell vorliegen.

Nummer	Fehler	Elementart/ PatternObject	Element	Verbindungstyp	Beschreibung
1	PatternObject - MIN	Vorlage PACS	-	-	Die minimale Anzahl wurde unterschritten (min = 1)
2	DirectConstraint - MAX	Vorlage Modalität-System	CT System	Anwendungsbaustein besitzt Bausteinschnittstelle	Die maximale Anzahl wurde überschritten (max = 1)

Abbildung 55 – Anzeige von Inkonsistenzen in einem Modell

Abbildung 55 zeigt eine mögliche Darstellung hierfür, in Anlehnung an die bereits im Tool3LGM² existierende Konsistenzprüfung (siehe Abbildung 13 in 2.2). Für sie würde es sich anbieten, eine solche Erweiterung durchzuführen, welche auch Abweichungen bezüglich eines Patterns anzeigen kann. Im hier dargestellten Fall handelt sich um ein fiktives Szenario, aufbauend auf der oben beschriebenen DICOM-Kommunikation, bei dem jetzt das Fehlen eines PACS sowie das zu häufige Vorhandensein eines Mod-PACS-Interface beim CT System unterstellt werden. In der ersten Zeile wird dabei eine Verletzung von Kriterium 3 angezeigt, bei dem das *minNumber*-Attribut von „Vorlage PACS“ unterschritten ist. Zeile zwei zeigt eine Inkonsistenz mit Kriterium 4 an, wobei hier der DirectConstraint von „Vorlage Modalität-System“ zu „Vorlage Mod-PACS-Interface“, wie er in Abbildung 26 bzw. Abbildung 39 zu sehen ist, verletzt wurde durch das Überschreiten seines *maxCard*-Attributes.

2.) Zum anderen obliegt es dem Tool3LGM², a priori eine Warnmeldung in den Fällen anzuzeigen, wo das Anlegen bzw. Entfernen eines Elementes oder einer Verknüpfung zu einer Über- bzw. Unterschreitung des *maxNumber*- bzw. *minNumber*-Attributes eines PatternObject führt. Möglichkeiten hierfür bestehen etwa durch das Hinzufügen einer Funktion Vor Inkonsistenzen warnen im Optionen-Menü in der Menüleiste.

Auf diese Weise ermöglicht es das Tool3LGM², ein Modell konform zu einem Pattern zu konzipieren, ohne den Modellierer im Umgang mit dem Tool selbst zu stark einzuschränken.

4.5.4 Automatisches Anlegen und Verknüpfen von Elementen

Neben der Definition von Einschränkungen bieten Konstruktionspatterns noch weitere Möglichkeiten, um den Modellierer zu unterstützen, etwa das automatische Aufbauen von Strukturen im Modell. In vielen Fällen werden bestimmte Beziehungen zwischen den Elementen eines Modelles zwingend erforderlich sein, um die Konformität mit dem Pattern zu gewährleisten. Dies ist etwa dann gegeben, wenn *minCard*-Attribute von DirectConstraints einen Wert ≥ 1 besitzen. Hier wäre es nur plausibel, wenn derartige, in jedem Fall benötigte Elemente und Verknüpfungen, automatisch angelegt werden könnten. Im Rückblick auf 4.5.2 wäre es dort beispielsweise wünschenswert, wenn das vom CT- bzw. MRI-System benötigte Mod-PACS-Interface direkt bei ihrer Instanziierung für sie angelegt werden würde. Hierbei kommen nun CreationLinks zum Einsatz, welche genau das bewerkstelligen. Diese können, wie in Abbildung 26 bzw. Abbildung 39 zu sehen, zwischen PatternObjects und PatternNodes angelegt werden und signalisieren, dass beim Ableiten eines Elementes oder einer Verknüpfung vom PatternObject am *creationSource*-Ende des CreationLink, dieses mit einem von *creationTarget* abgeleiteten Element über die CreationLink.*association* verknüpft werden soll. Dazu wird zunächst ein solches Element neu instanziiert und daraufhin die Verknüpfung angelegt. Dabei ist zu beachten: In dem Fall, wo *creationTarget.maxNumber* = 1 gilt, gibt es nur höchstens ein zum *creationTarget* gehöriges Element im Modell, mit dem alle Instanzen, die vom *creationSource* abgeleitet wurden, verknüpft werden sollen. Das heißt: Da dieses vom *creationTarget* abgeleitete Element nur einmal im gesamten Modell vorkommen darf, können ersichtlicherweise nicht für jeden CreationLink neue Instanzen angelegt werden. Stattdessen sollen alle vom *creationSource* abgeleiteten Instanzen mit genau diesem einen Element verknüpft werden (weitere Erläuterungen hierzu sind in Abschnitt 4.4.3.4 unter „Anmerkungen – Punkt 1“ gegeben).

Durch ein geeignet designtes Konstruktionspattern kann somit dem Modellierer viel Arbeit erspart werden, indem benötigte Strukturen automatisch im Modell angelegt werden, was darüber hinaus auch zu dessen Konformität mit dem Pattern beiträgt.

Anhand des eben gegebenen Vorschlages für die Implementierung des 3LGM²-CF im Tool3LGM² kann auf Basis der technischen Spezifikation in den vorangegangenen Abschnitten die Patternmodellierung und Patternanwendung umgesetzt werden. Wie bereits eingangs erwähnt wurde, sind insbesondere für Informationssysteme im Gesundheitswesen die Technical Frameworks bzw. Integration Profiles der IHE von großer Bedeutung, da diese für verschiedenste Problemstellungen umfangreiches Entwurfswissen zur Verfügung stellen. Im Folgenden soll dazu gezeigt werden, wie das 3LGM² - Construction Framework zur Formalisierung der Konzepte der IHE in 3LGM²-konformen Konstruktionspatterns angewendet wird.

5 IHE – Technical Frameworks

Die IHE ist eine Initiative, welche die Verbesserung der Interaktion von Computersystemen im Gesundheitswesen verfolgt. Im Fokus steht dabei vor allem der Einsatz bewährter Methoden und Standards, beispielsweise HL7, DICOM, SOAP etc., hin zu einer effizienteren Informationsverarbeitung und verbesserten Interoperabilität der beteiligten Systeme. Hierzu formuliert die IHE Technical Frameworks, welche verschiedene Problembereiche hinsichtlich deren Integrationsbedürfnisse adressieren, unter anderem:

- Cardiology,
- Patient Care Coordination,
- IT Infrastructure,
- …

Letzteres etwa befasst sich mit der Integration der Computersysteme selbst und zeigt dazu Wege auf, wie eine effiziente Zusammenarbeit und Kommunikation erreicht werden kann. Dabei untergliedert sich jedes der Frameworks in verschiedene Integration Profiles. So beinhaltet beispielsweise das IT Infrastructure Technical Framework die Integration Profiles:

- Patient Identifier Cross-referencing (PIX),
- Consistent Time (CT),
- Cross-Enterprise Document Sharing (XDS),

und weitere mehr. Jedes dieser Integration Profiles gibt dabei eine Lösungsmöglichkeit für einen abgegrenzten Bereich innerhalb der Thematik des jeweiligen Technical Frameworks vor. So beschäftigt sich etwa das PIX Integration Profile mit der Referenzierung verschiedener IDs für denselben Patienten, das CT Integration Profile mit der Problematik der zeitlichen Synchronität der Computersysteme, und das XDS Integration Profile mit dem einrichtungsübergreifenden Austausch patientenbezogener Dokumente, jeweils im Kontext der Integration der beteiligten Computersysteme.

Die IHE nutzt dabei für alle Integration Profiles und für alle Technical Frameworks dasselbe universelle Konzept, welches im Folgenden vorgestellt werden soll.

5.1 Konzept der Technical Frameworks

Die IHE stellt umfangreiches Wissen hinsichtlich der Architektur von Informationssystemen im Gesundheitswesen über Technical Frameworks bereit. Darin werden zum einen allgemeine Grundlagen und Prinzipien erläutert und zum anderen technische Details für eine konkrete Implementierung gegeben. Die IHE vermeidet es dabei, von speziellen Produktkategorien auszugehen und beschränkt sich stattdessen auf die Beschreibung von Komponenten inklusive der für die Integration benötigten Funktionen. Die Komponenten werden dabei als *Actors* und die zu unterstützenden Funktionen als *Transactions* bezeichnet. Die verwendete Terminologie macht bereits deutlich, dass es sich bei den Funktionen tatsächlich um Interaktionen handelt, die zwar im Sinne einer Funktionalität von einem Actor implementiert werden müssen, allerdings immer eine Reaktion bei einem anderen Actor hervorrufen.

Diese Interaktion der Actors geht dabei einher mit einem Nachrichtenaustausch (Abbildung 56), welcher auf Standards, wie HL7, SOAP und weiteren mehr, sowie speziellen Anpassungen und Kombinationen derer durch die IHE, basiert. Dabei handelt es sich im Allgemeinen um eine ereignisgesteuerte Kommunikation, welche über die Typen von Nachrichten hinaus noch mit Beschreibungen zu Umständen und Gegebenheiten, die den Nachrichtenaustausch auslösen, versehen ist. Derartige Detailinformationen werden von der IHE üblicherweise in textueller Form den Diagrammen beigefügt.

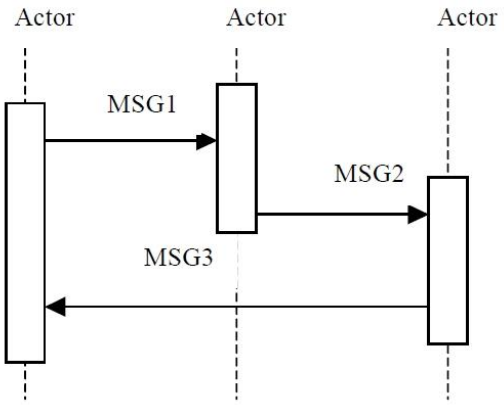

Abbildung 56 – Nachrichtenaustausch zwischen Actors (Quelle: [ITI-TF])

Im konkreten Fall – hier etwa am Beispiel des CT Integration Profile – wird die Interaktion wie in Abbildung 57 dargestellt. Zu sehen sind dabei die Actors *Time Server* und *Time Client*, sowie die Transaction *Maintain Time [ITI-1]*. Jedes Integration Profile, jeder Actor und jede Transaction wird dabei mit umfangreichen Beschreibungen zu Anwendungsgebiet, Funktionsweise, Kommunikationsstandards etc. ausgestattet. Des Weiteren werden technische Spezifikationen hinsichtlich der Implementierung des jeweiligen Integration Profile gegeben. Über dieses Konzept wird so das Zusammenarbeiten funktionaler Komponenten in Informationssystemen formalisiert und damit die Lösung zu den jeweiligen Integrationsbedürfnissen gegeben.

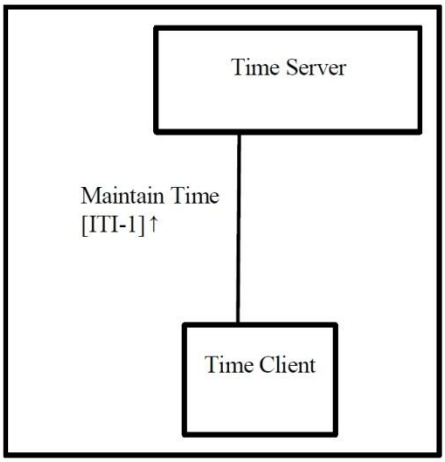

Abbildung 57 – Actors und Transaction des CT Integration Profile (Quelle: [ITI-TF])

IHE Integration Profiles basieren damit auf Entwurfswissen aus erprobten Ansätzen und liefern unter anderem eine Anleitung für die Konstruktion integrierter Informationssysteme im Gesundheitswesen. Somit dienen sie potentiell als Basis für Patterns entsprechend Malich's Patternbegriff aus 3.2.

5.2 Abbildung im 3LGM²

Häufig weisen IHE Profile eine hohe Komplexität auf, bedingt durch die Tatsache, dass sie zum einen die allgemeinen Grundlagen schildern und zum anderen diese durch technische Spezifikationen fundieren. Integration Profiles haben damit den Vorteil, dass durch sie umfassende Lösungen geboten werden, bergen aber insbesondere den Nachteil einer schlechten Überschaubarkeit und mangelnder Transparenz der Zusammenhänge zwischen allgemeinen Grundlagen und technischen Details, deren Nachvollziehbarkeit durch die vorwiegend textuelle Form noch zusätzlich erschwert wird. Damit sind sie nur bedingt für die Planung und Weiterentwicklung von Informationssystemen einsetzbar. Geeigneter wäre daher ein Ansatz zum Einbeziehen von dem in den Profiles enthaltenen Entwurfswissen über die Modellierung. Modelle – etwa auf Basis des 3LGM² – bieten, wie eingangs bereits erwähnt, die Möglichkeit, Sachverhalte grafisch übersichtlich abzubilden und könnten damit helfen, dieses Wissen besser für die Planungsaktivitäten nutzbar zu machen. Da sich IHE Profile, wie oben bereits aufgeführt, nach 3.2 als Grundlage für Patterns eignen, bleibt nun noch zu prüfen, ob für sie darüber hinaus die Eignung zur Definition von Konstruktionspatterns (nach 3.3) besteht, insbesondere in 3LGM²-konformer Weise, um die enthaltenen Konzepte für die Modellierung mit dem 3LGM² bzw. Tool3LGM² zugänglich machen zu können.

Unter Betrachtung der Techncial Frameworks und Integration Profiles können im 3LGM² nun analoge Ansätze zur Repräsentation von Informationssystemen gefunden werden. Dort sind etwa die funktionalen Komponenten als Anwendungsbausteine dargestellt und der Nachrichtenaustausch als Kommunikationsbeziehungen. Des Weiteren gibt es Möglichkeiten zur Modellierung von Austauschformaten und Standards, was über Nachrichtentypen und Kommunikationsstandards geschieht. Man erkennt so, dass Konzepte der IHE im 3LGM² abgebildet werden können. Gesucht ist nun ein abstraktes Konstruktionspattern, welches als Richtlinie für die 3LGM²-konforme Formalisierung von Integration Profiles dient. Hierfür ist es nur naheliegend, das 3LGM² - Construction Framework zu Hilfe zu nehmen, unter dessen Anwendung sich nun folgende Abbildungsregeln ergeben:

Jeder Actor – als funktionale Komponente – wird in Form eines PatternNode modelliert, welcher auf die Klasse Anwendungsbaustein (rechnerbasiert) abbildet:

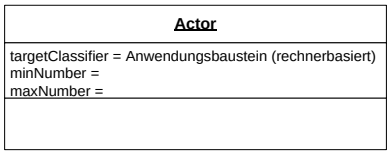

Abbildung 58 – IHE Actor als 3LGM²-CF PatternNode

Da sich die Integration Profiles auf interagierende Computersysteme beziehen, kann hier von rechnerbasierten Anwendungsbausteinen ausgegangen werden, da papierbasierte Anwendungsbausteine nur für den nichtcomputerunterstützten Fall in Frage kommen. Alle weiteren Attribute sowie die Kurzbeschreibung variieren für jedes konkrete Integration Profile und werden hier daher nicht näher spezifiziert. Actors sollen nun miteinander interagieren und müssen deshalb über Transactions verbunden werden. Diese werden als PatternRelations dargestellt und als Kommunikationsbeziehungen im 3LGM² abgebildet (Abbildung 59).

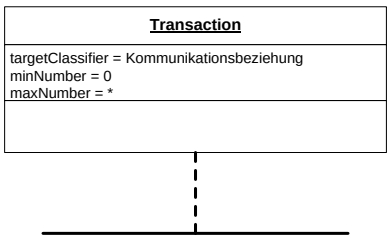

Abbildung 59 – IHE Transaction als 3LGM²-CF PatternRelation

(Anmerkung: Tatsächlich werden pro Transaction zwei Kommunikationsbeziehungen bzw. zwei dieser PatternRelations benötigt. Darauf wird im Folgenden noch näher eingegangen.)

Damit nun Actors über Transactions verbunden werden können, sind diese zunächst mit geeigneten Bausteinschnittstellen auszustatten („Actor-Transaction-Interface“). Dazu kommen DirectConstraints zum Einsatz, welche die für einen Actor anlegbaren Schnittstellen über die besitzt-Assoziation festlegen. Die jeweils erforderlichen Kardinalitäten und Häufigkeiten können dabei von Profile zu Profile unterschiedlich sein und werden daher hier nicht festgelegt.

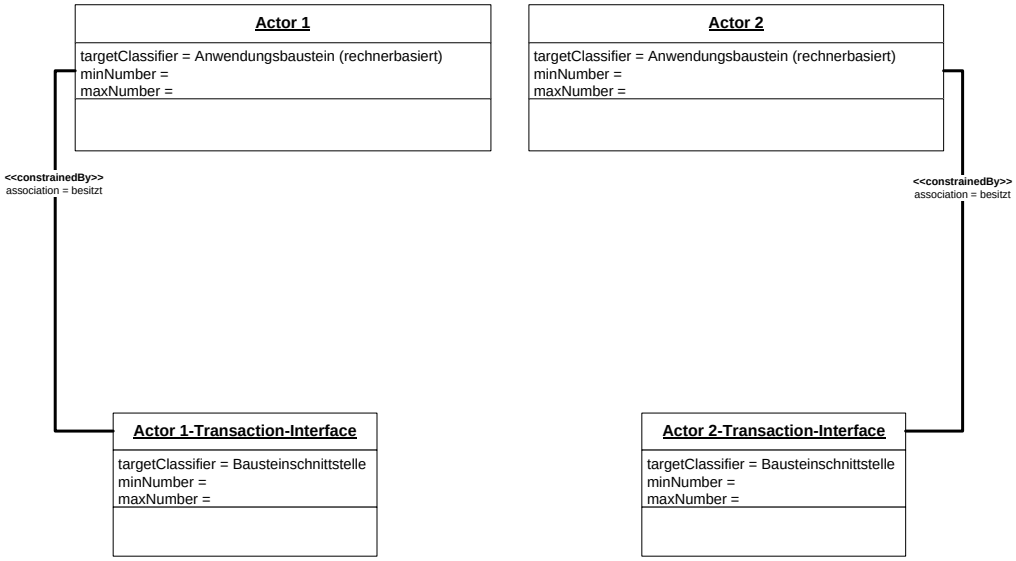

Abbildung 60 – Kopplung von IHE Actors und 3LGM²-Bausteinschnittstellen durch 3LGM²-CF DirectConstraints und CreationLinks

Die Interaktion zweier Actors über eine Transaction wird nun durch Verbinden der Interfaces über die zu ihr korrespondierenden PatternRelations erzielt. Es handelt sich dabei um eine Aufspaltung der Transaction in ihre zwei elementaren Bestandteile:

1. Das Initiieren (durch Actor 1)
2. Die Reaktion (durch Actor 2).

Dies hat dann die Repräsentation der Transaction durch zwei Kommunikationsbeziehungen bzw. zwei PatternRelations als Konsequenz (Abbildung 61).

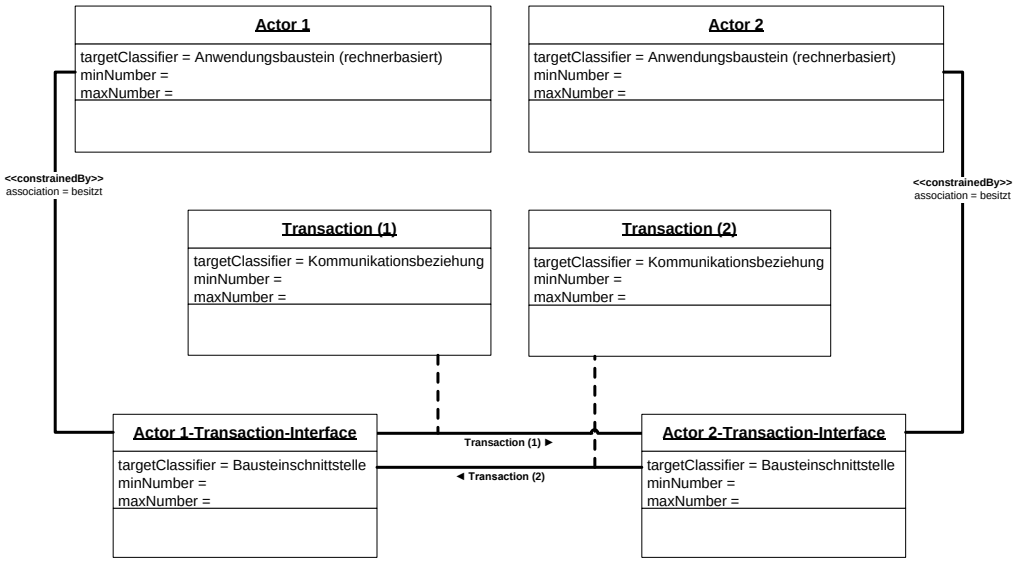

Abbildung 61 – Verbinden von Actors über Transactions im 3LGM²-CF

Die Notwendigkeit dieser Aufgliederung liegt darin begründet, dass die Transaction eigentlich eine Interaktion darstellt und damit aus Aktion und Reaktion besteht, das Pendant zu ihr im 3LGM² – die Kommunikationsbeziehung – allerdings unidirektional ist und damit die Transaction nur aus der Sicht jeweils eines Actors abbilden kann. Daher erfolgt hier die Zerlegung in die PatternRelations „Transaction (1)“ und „Transaction (2)“, welche beide dieselbe Transaction bezeichnen, allerdings einmal im Sinne der Initiierung durch Actor 1 und einmal hinsichtlich der Reaktion durch Actor 2.

Au diese Weise können die grundlegenden Strukturen eines Integration Profile in einem Konstruktionspattern formalisiert werden. Um nun allerdings die vollen Möglichkeiten des 3LGM² auszuschöpfen, kann ein Pattern noch wie folgt erweitert werden:

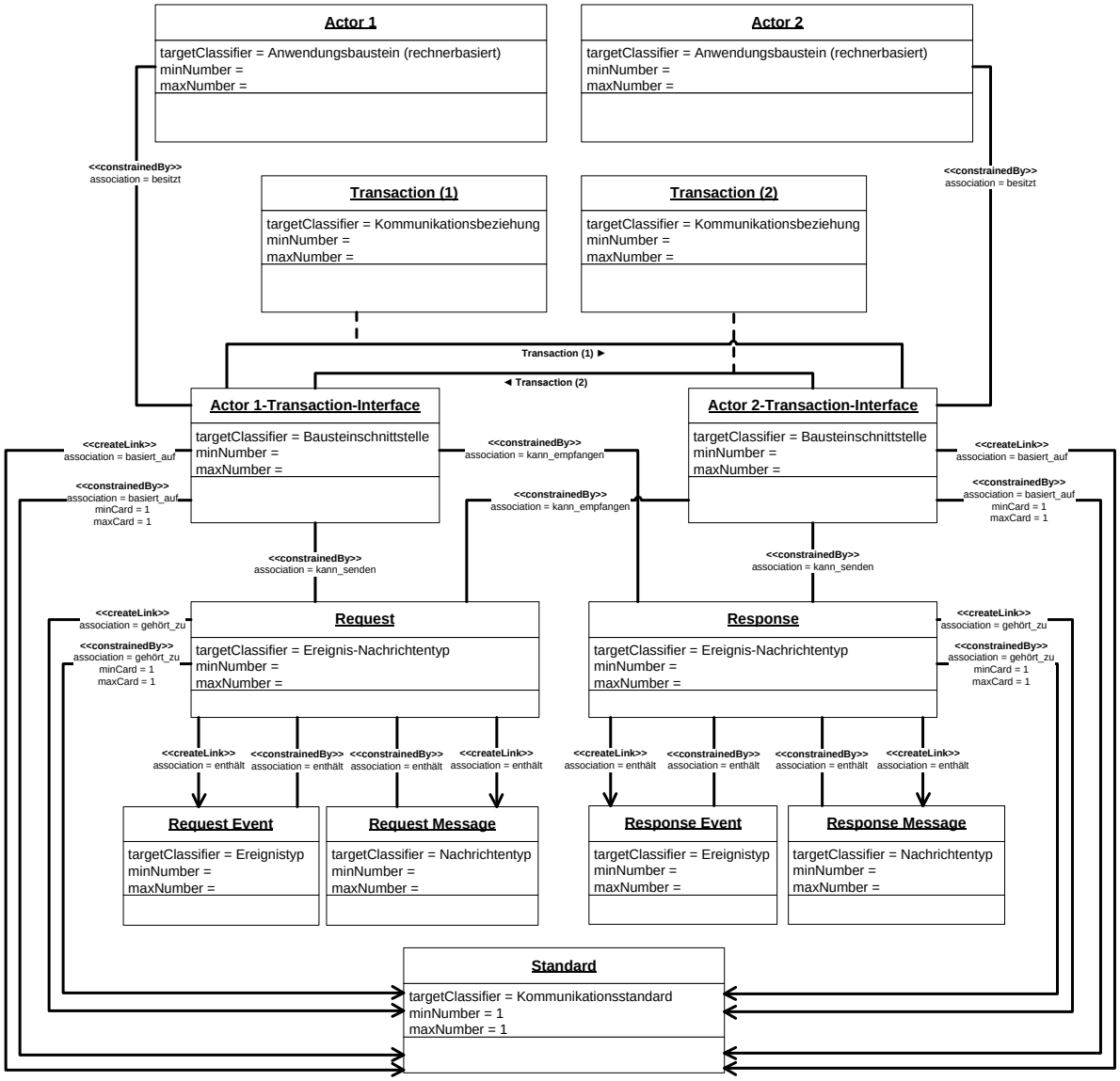

Abbildung 62 – SubPattern-Vorlage einer IHE Transaction

In den IHE Integration Profiles werden für jede Transaction Ereignisse spezifiziert, welche die beteiligten Actors zum Austausch von Nachrichten veranlassen. Diese korrespondieren zu den 3LGM²-Ereignistypen. Dabei muss über sie zum einen die Ursache für das Auslösen der Transaction durch einen der Actors und zum anderen das Reagieren des zweiten Actor auf die eingehende Transaction ausgedrückt werden („Request/Response Event“). Jedes dieser Ereignisse bewirkt dabei den Austausch bestimmter Nachrichten („Request/Response Message“). Diese können im 3LGM² durch die Klasse Nachrichtentyp beschrieben werden. Da sich die Integration Profiles auf interagierende Computersysteme beziehen, kann von Nachrichtentypen ausgegangen werden, da Dokumenttypen nur für

den nichtcomputerunterstützten Fall in Frage kommen. Für sie ist es nun insbesondere vonnöten, die umfangreichen Angaben der IHE hinsichtlich der zu verwendenden Formate zu hinterlegen. Dazu sollten sowohl Nachrichtentypen als auch Ereignistypen als Vorlagen im Pattern erfasst werden. Für sie ist es so durch den Patternmodellierer möglich, diese umfangreichen Hintergrundinformationen und technischen Details im Beschreibungstext wiederzugeben. Typischerweise werden Ereignistypen und Nachrichtentypen über die jeweilige enthält-Assoziation zu sogenannten Ereignis-Nachrichttypen („Request/Response“) zusammengefasst, um auszudrücken, dass ein Austausch bestimmter Nachrichten wiederum durch bestimmte Ereignisse bedingt wird. Dies wird über die für sie definierten DirectConstraints ausgedrückt. Zur Unterstützung eines Modellierers wurden dazu noch CreationLinks angelegt, welche ein automatisches Zuweisen derer zu den Ereignis-Nachrichtentypen bewerkstelligen. Bausteinschnittstellen bzw. Kommunikationsbeziehungen sind dann in der Lage, diese Kombinationen aus Ereignis- und Nachrichtentypen zu übertragen. Dazu werden zwischen Actor-Transaction-Interfaces und Request bzw. Response DirectConstraints über die kann_senden- und kann_empfangen-Assoziation angelegt, worüber ausgedrückt wird, dass die Kommunikation der dazugehörigen Ereignis-Nachrichtentypen über derartige Bausteinschnittstellen erfolgen sollte. Sowohl Ereignis-Nachrichtentypen als auch Bausteinschnittstellen gehören dabei einem bestimmten Kommunikationsstandard („Standard“) an bzw. basieren auf ihm. Dieser Zusammenhang wird erneut über DirectConstraints ausgedrückt, indem darüber Request, Response, Actor 1-Transaction-Interface und Actor 2-Transaction-Interface mit ihm verbunden werden.

Anmerkung: Für die dabei gebrauchten Assoziationen gehört_zu bzw. basiert_auf wird die maximale Kardinalität an deren Kommunikationsstandard-Enden bereits mit dem Wert „1“ implizit durch das 3LGM² gegeben. Im Pattern wird nun darüber hinaus gefordert, dass auch die minimale Kardinalität den Wert „1“ besitzen sollte, im Gegensatz zum 3LGM², welches hierfür auch „0“ zulässt. Daher wurde für diese Constraints, jeweils das Attribut minCard = „1“ gesetzt, wodurch die Zugehörigkeit dieser speziellen Bausteinschnittstellen und Ereignis-Nachrichtentypen zum Kommunikationsstandard erzwungen wird.

Auch hier ist noch zusätzlich für eine Unterstützung des Modellierers gesorgt worden, durch das Anlegen entsprechender CreationLinks, welche in einem Modell das automatische Verknüpfen der Instanzen, die von diesen PatternNodes abgeleitet sind, auslösen. Somit wird etwa für derartige Bausteinschnittstellen (die beispielsweise von Actor 1-Transaction-Interface abgeleitet werden), der Kommunikationsstandard direkt bei ihrer Instanziierung für sie festgelegt. Anzumerken ist noch, dass es innerhalb eines Modelles nur genau eine Instanz dieses Kommunikationsstandards geben sollte, was durch die Attribute Standard.*minNumber* bzw. Standard.*maxNumber* ausgedrückt wird.

Das eben geschilderte Konzept ist die Grundlage für die Formalisierung beliebiger Transactions und Integration Profiles der IHE in 3LGM²-konforme Konstruktionspatterns. Anzumerken ist, dass dabei typischerweise Variationen auftreten, die in den jeweiligen Eigenheiten eines Integration Profile begründet sind, welche sich etwa im Wegfallen bestimmter PatternObjects sowie in der Konkretisierung der Häufigkeiten und Kardinalitäten niederschlägt. Umfassende Beschreibungen bzw. Erklärungen hinsichtlich der Verwendung einzelner PatternObjects können im *description*- bzw. *applicationHint*-Attribut hinterlegt werden, wodurch es etwa möglich wäre, für die Ereignistypen („Request/Response Event“) anzugeben, durch welche Arten von Aufgabe sie innerhalb eines Modelles ausgelöst werden, was dann wiederum dem Modellierer im Tool3LGM² Aufschluss über die Verwendung des Patterns gibt. Außerdem können Kurzbeschreibungen definiert werden, welche im unteren Bereich der PatternObject-Shapes angezeigt werden und damit einen schnellen Einblick in die dargestellte Thematik ermöglichen. Da es sich in den vorangegangenen Abbildungen um ein abstraktes Beispiel handelte, wurden derartige Beschreibungen nicht formuliert. Auch wurden nur allgemeine Bezeichnungen stellvertretend für die eigentlichen Namen der Actors, Transaction usw. verwendet, die dann entsprechend für jedes Integration Profile zu ersetzen sind. Im folgenden Kapitel soll dazu nun eine konkrete Anwendung mit dem XDS Integration Profile gezeigt werden.

6 Das XDS Integration Profile als Konstruktionspattern für 3LGM²-Modelle

Dieses Kapitel beschäftigt sich mit dem XDS Integration Profile der IHE sowie dessen Anwendung bei der Modellierung eines transinstitutionellen Dokumentenaustausches im Gesundheitswesen. Hier sollen nun abschließend die Erkenntnisse aus den vorangegangenen Kapiteln zusammengeführt werden, um dieses Integration Profile in ein 3LGM²-konformes Konstruktionspattern zu überführen. Zunächst wird allerdings noch ein allgemeiner Einblick in die Thematik und das Profile selbst gegeben.

6.1 Transinstitutioneller Dokumentenaustausch im Gesundheitswesen

Die zentrale Anforderung der klinischen Dokumentation an die Informations- und Kommunikationstechnologie ist die geeignete Unterstützung des Sammelns und Bereitstellens aller relevanten Informationen, die im Zusammenhang mit der Versorgung eines Patienten erhoben wurden. Diese müssen in der erforderlichen Form erfasst werden und für alle berechtigten Personen am richtigen Ort und zur richtigen Zeit zur Verfügung stehen [vgl. LEINER et al. 2003]. Typischerweise wird dies mittels medizinischer Dokumentationssysteme realisiert, welche innerhalb einer Einrichtung die Informationen halten. Im Kontext des transinstitutionellen Dokumentenaustausches gilt es, die Verfügbarkeit dieser Informationen über die Einrichtungsgrenzen hinweg zu erweitern, um so das Zusammenführen der einzelnen Dokumentationen zu einer ganzheitlichen Krankenakte – in der Terminologie der IHE als „Longitudinal Health Record“ bezeichnet – zu ermöglichen, wobei nun hier über die einrichtungsinternen Anforderungen hinaus die Diskussion weiterer Fragestellungen erforderlich ist, welche im Folgenden auszugsweise dargestellt sind:

- Wie kann ein Patient innerhalb eines Verbundes mehrerer Versorgungseinrichtungen identifiziert werden?
- Was genau ist in diesem Zusammenhang ein Dokument und welche Informationen enthält es?
- Welche Vorkehrungen sind notwendig, um die bei der Behandlung entstandenen Dokumente dem Patienten zuordnen zu können?
- Wer ist für das zuverlässige Bereitstellen der Dokumente verantwortlich und wie kann der Zugriff auf diese am richtigen Ort und zur richtigen Zeit gewährleistet werden?
- Wie soll die Kommunikation der Dokumente erfolgen?
- Welche Möglichkeiten existieren für deren Suche?

Ein transinstitutionelles Informationssystem muss all diese Aspekte abdecken, sodass ein Wechsel vom einrichtungsspezifischen Fokus hin zu einer patientenzentrischen Perspektive auf die Informationen erfolgen kann, wodurch eine ganzheitliche Sicht auf den Patienten bzw. seine Krankengeschichte gewährleistet wird und damit eine potentiell bessere Versorgung möglich ist.

6.2 Das XDS Integration Profile

Vorbemerkung:

Alle folgenden Abschnitte werden sich auf Version b des XDS Integration Profile beziehen, welche die ursprüngliche Fassung (jetzt Version a) 2009 abgelöst hat. XDS.b bietet durch die Einführung zusätzlicher Restriktionen eine im Vergleich zur Vorgängerversion höhere Sicherheit bei dem täglichen Umgang mit patientenbezogenen Informationen und wird aus diesem Grund vorzugsweise für alle weiteren Schritte als Basis dienen.

Die IHE beschränkt sich in ihren Integration Profiles nicht nur auf intrainstitutionelle Gegebenheiten, sondern fokusiert auch auf eine einrichtungsübergreifende Integration. Eines der Problemfelder hierbei ist der transinstitutionelle Dokumentenaustausch: Dieser soll es erlauben, medizinische Dokumente überall dort verfügbar zu machen, wo die Versorgung eines Patienten erforderlich ist, unabhängig davon, wo diese ursprünglich erstellt wurden.

Als Teil des ITI-Technical Framework stellt die IHE hierzu das XDS Integration Profile bereit, welches für diese Problematik Anforderungen verdeutlicht und Lösungen aufzeigt.

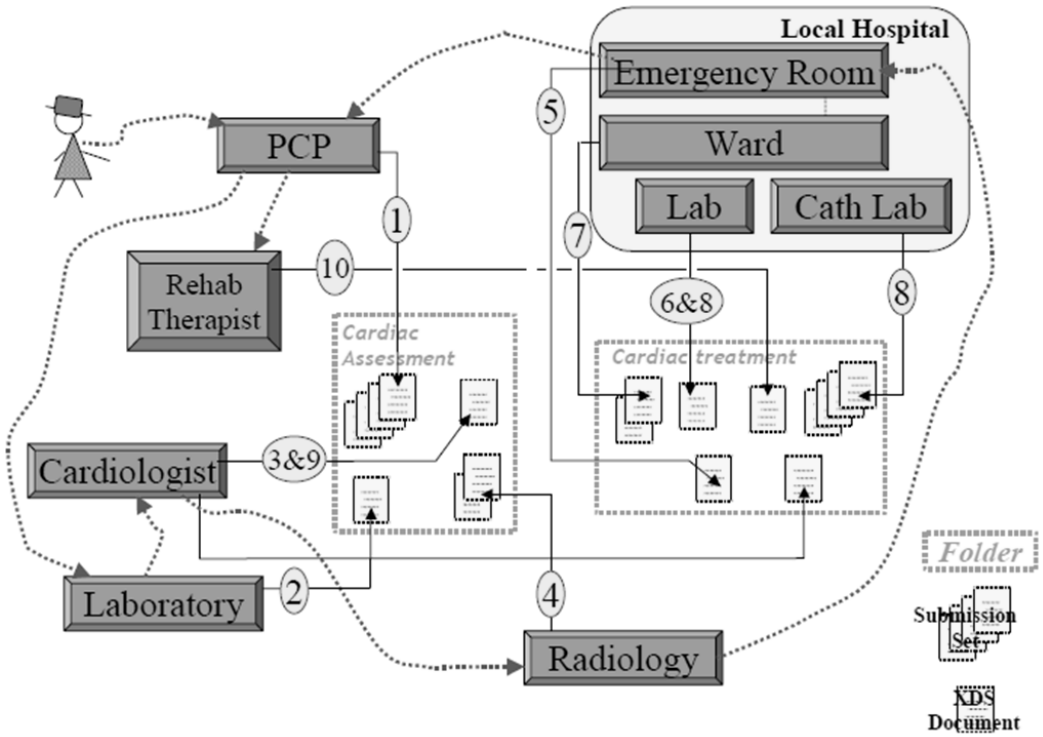

Abbildung 63 – Szenario einer verteilten kardiologischen Behandlung (Quelle: [ITI-TF])

In Abbildung 63 ist hierzu eine potentielle Anwendung des XDS Integration Profile im Kontext des Zusammenschlusses medizinischer Einrichtungen zu einem regionalen Kardiologienetzwerk gegeben. Der Patient durchläuft in diesem Beispiel eine Reihe von Stationen vom „Primary Care Provider“ (PCP) bis hin zum „Local Hospital“, in denen jeweils unterschiedliche Dokumente erzeugt werden. Hierbei ergibt sich die Notwendigkeit, etwa einen Zugriff auf bereits erstellte Befunde zu erlangen. So wäre es beispielsweise für den „Cardiologist“ wünschenswert, Informationen zu den beim PCP gemachten Untersuchungen zu erhalten, um wiederum seine Behandlungsmaßnahmen besser am Patienten ausrichten zu können. Im Fokus des XDS Intergation Profile steht nun das Etablieren eines zentralen Zugriffes auf diese Informationen, indem sowohl ein Bereitstellen als auch Abfragen derer in allen Einrichtungen ermöglicht wird. Wie in Kapitel 5 geschildert, basiert auch dieses Integration Profile auf den Grundbausteinen Actor und Transaction, wobei hier Actors Rollen von Computersystemen im Kontext des einrichtungsübergreifenden Dokumentenaustausches und Transactions die dafür zu implementierenden Funktionalitäten beschreiben. Diese Transactions

werden dabei als Interaktionen verstanden, über die Daten ausgetauscht werden können. Das XDS Integration Profile sieht dazu, wie in Abbildung 64 dargestellt, fünf Actors und Transactions vor.

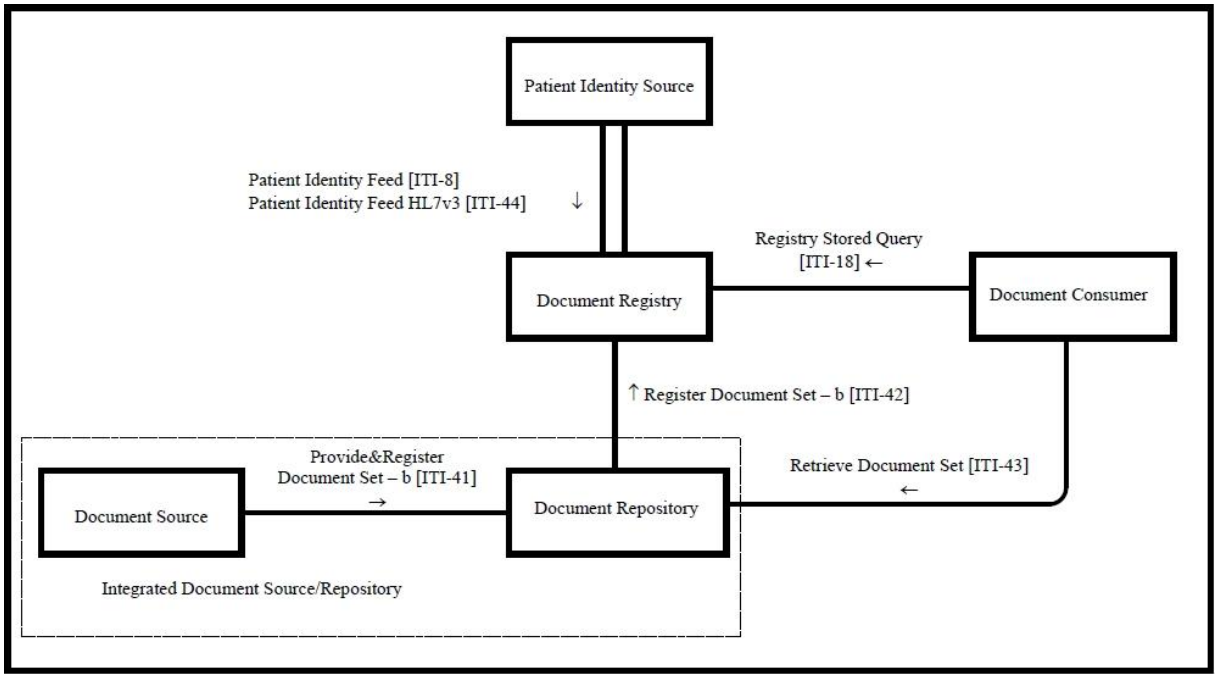

Abbildung 64 – Interaktionsdiagramm des XDS Integration Profile (Quelle: [ITI-TF])

Der Document Source Actor ist der Erzeuger von Dokumenten. In der realen Welt repräsentieren diese die Teile aus dem Informationssystem einer medizinischen Einrichtung, welche patientenbezogene Informationen in Form solcher Dokumente – sogenannte „XDS Documents" – für deren einrichtungsübergreifenden Austausch erzeugen. Die IHE macht dabei keinerlei Einschränkungen hinsichtlich des genauen Aufbaus dieser Dokumente. Ausschließlich die Arten des Inhalts werden begrenzt. So sind etwa folgende Typen patientenbezogener Informationen keine XDS Documents:

- Dynamische Daten (z.B. Allergielisten) sowie
- Befundanforderungen

Beispiele für typsiche XDS Documents wären hingegen:

- Arztbriefe,
- Radiologische Bilddaten,
- Laborbefunde etc.

Es ist dabei offen gehalten, inwiefern diese Inhalte in Dokumenten zusammengefasst werden, unter anderem, ob es sich um ein zusammengesetztes Dokument handelt, das verschiedene Arten von Informationen beinhaltet oder auch, welche Formate jeweils verwendet werden. Grundsätzlich gilt eine wesentliche Regel: Ein XDS Document kapselt jeweils Informationen zu genau einem Patienten. Der Document Source Actor ist neben dem Erzeugen dieser Dokumente verantwortlich für das Bereitstellen von Metadaten – sogenannte XDS Document Entries – welche jeweils zu genau einem XDS Document gehören. Diese umfassen unter anderem Attribute wie Patienten-ID, Author, den Typ des Dokuments sowie Angaben zu den Ereignissen im Zusammenhang mit der Versorgung eines Patienten, die das Anlegen des Dokumentes bedingt haben. Es ist Aufgabe des Document Source Actor, XDS Documents und XDS Document Entries an einen Document Repository Actor zu übergeben, welcher die Dokumente persistent speichert und anderen Einrichtungen zur Verfügung stellt. Diese Fähigkeit zum Übermitteln von Dokumenten und Metadaten wird dabei durch die Provide & Regsiter Document Set-b Transaction beschrieben. Der Document Repository Actor wiederum leitet dann die Metadaten über die Register Document Set-b Transaction an einen zentralen Document

Registry Actor weiter, welcher den XDS Document Entry einschließlich eines Verweises zum Document Repository und dem dazugehörigen XDS Document speichert.

Die Zuordnung eines Dokumentes zu einem Patienten wird dabei durch einen Patient Identity Source Actor ermöglicht, der als Quelle von Patienten-IDs fungiert, welche in den Dokument-Metadaten innerhalb des Document Registry Actor erfasst werden. Hierzu müssen beide Actors die Patient Identity Feed Transaction implementieren, welche den Document Registry Actor mit Patientenidentitäten versorgt. Die Zuweisung von Patienten-IDs zu Dokumenten erfolgt dabei durch den verantwortlichen Document Source Actor.

Nach einer erfolgreichen Speicherung eines XDS Document in einem Document Repository Actor und der Registrierung der Metadaten in Form eines XDS Document Entry im Document Registry Actor können nun weitere medizinische Einrichtungen bzw. Anwendungssysteme auf diese Dokumente zugreifen. Hierbei nehmen sie die Rolle des Document Consumer Actor ein, was ihnen erlaubt, über die Registry Stored Query Transaction zunächst nach den für einen Patienten registrierten Dokumenten suchen. Der Document Registry Actor stellt hierfür eine Reihe vordefinierter Queries bereit, welche über Suchparameter ein Einschränken der zurückgelieferten Ergebnisse erlauben. Durch sie erhält ein Document Consumer Actor Dokument-Matdaten (XDS Document Entries) für einen bestimmten Patienten, deren oben bereits beschriebenen Attribute nun dabei helfen, Dokumente von Interesse zu identifizieren. Über die Retrieve Document Set Transaction können diese dann vom entsprechenden Document Repository Actor abgefragt werden.

Dieser Mechanismus des Dokumentaustausches bringt vor allem den Vorteil mit sich, dass ein Hinzufügen zusätzlicher kommunizierender Document Source und Document Consumer Actors leicht möglich ist, indem sie diese standardisierten Transactions implementieren. Medizinische Einrichtungen bzw. deren Anwendungssysteme in den Rollen Document Source und Document Consumer Actor werden dabei zu sogenannten *XDS Affinity Domains* zusammengefasst. Jede dieser Affinity Domains repräsentiert dabei eine Menge solcher Einrichtungen, die sich zum Zwecke des Dokumentenaustausches zusammengeschlossen haben und einen zentralen Document Registry sowie Patient Identity Source Actor teilen, über die Dokumente registriert und den Patienten zugeordnet werden können. Der Patient Identity Source Actor muss dabei dafür sorgen, dass allen Patienten innerhalb der XDS Affinity Domain eine eindeutige ID zugewiesen wird. Analog dazu ist es Aufgabe des Document Registry Actor, ihnen die erstellten Dokumente zuzuordnen, sodass diese dann in den verschiedenen Einrichtungen abrufbar sind.

6.3 Transformation des XDS Integration Profile in ein Konstruktionspattern

Die eben gegebene Darstellung des XDS Integration Profile umfasste dessen allgemeine Grundlagen, wie sie im ITI-Technical Framework vermittelt werden. Für jede Transaction werden darin zusätzlich diese grundlegenden Konzepte zu detaillierten technischen Spezifikationen aufgegliedert hinsichtlich der Adaption von Kommunikationsstandards. Dies schließt unter anderem den Aufbauen von Austauschformaten ein, wie etwa die Verwendungsweise spezieller HL7-Nachrichtensegmente und weitere Vorgaben, welche durch die Actors umgesetzt werden müssen, um im einrichtungsübergreifenden Dokumentenaustausch partizipieren zu können. Aufgrund des daraus resultierenden großen Umfanges dieses Integration Profile wäre es hier insbesondere angebracht, eine Überführung in ein Konstruktionspattern vorzunehmen – zum einen, um strukturelle Zusammenhänge und deren Abbildung im 3LGM² darzustellen und zum anderen, um eine Verknüpfung derer mit den jeweiligen technischen Details zu erreichen. Hieraus ergeben sich drei wesentliche Chancen:

1. Ein Modellierer, der die Architektur eines entsprechenden transinstitutionellen Informationssystems anlegen möchte, wird durch den Einsatz des Patterns hin zu einem konformen Modell gelenkt und damit bezüglich der Einarbeitung in das Wissen sowie dessen Abbildung in der Modellierungssprache entlastet.
2. Nutzer des dabei resultierenden Soll-Modelles erhalten nun bereits durch die Darstellungsform als Modell selbst eine bessere Übersicht zu den strukturellen Anforderungen des Dokumentenaustausches und haben zugleich die Sicherheit, damit auf bewährtes Wissen zu bauen.
3. Darüber hinaus geben die aus dem Pattern übernommenen technischen Beschreibungen in leicht zugänglicher Weise Aufschluss zur konkreten Implementierung, sodass ein Einsatz des Modelles nicht für die Planung, sondern auch für die Umsetzung ermöglicht wird.

Dementsprechend soll das XDS Integration Profile nun in ein Konstruktionspattern überführt werden. Dazu wird es in SubPatterns zerlegt, welche dann zusammen das vollständige Pattern dieses Integration Profile ergeben. Im Folgenden werden nun diese SubPatterns dargestellt und erläutert, wobei jedes Unterkapitel zu jeweils einem der SubPatterns korrespondiert. Die Überschriftentitel sind dabei direkt als deren Bezeichnung zu interpretieren, sprich, jeweils als das SubPattern.*name*-Attribut. Die genauen Beschreibungen, sprich die *description*-Attribute von SubPatterns, PatternObjects und vom Pattern selbst werden hier nicht aufgeführt und sind stattdessen in den Anlagen (9.1) hinterlegt. Ebenso werden etwa die *applicationHint*-Attribute der PatternObjects ausschließlich den Anlagen beigefügt, da diese nicht in der grafischen Darstellung wiedergegeben werden.

Vorbemerkungen:

- *Die Bezeichnungen und Beschreibungen in den folgenden Abbildungen sind denen im XDS Integration Profile [XDS] entlehnt und werden sowohl für PatternObjects als auch für SubPatterns verwendet. Hierzu wurden wesentliche Teile des Textes direkt übernommen, um möglichst nah am Wortlaut der IHE zu bleiben. Das Pattern beabsichtigt keine „Neuerfindung" des XDS Integration Profile, sondern versucht lediglich, eine möglichst authentische Repräsentation des Entwurfswissens zu erreichen. Die dargestellten Beziehungen, Constraints etc. sind dabei eine entsprechende Interpretation der Konzepte des Integration Profile im 3LGM², welche über den Einsatz des 3LGM²-CF bewerkstelligt wird und auf den Überlegungen zum allgemeinen Abbilden von IHE Konzepten im 3LGM² aus 5.2 aufbaut.*

- *Gemäß der Definition des 3LGM²-CF sind die dargestellten Kurzbeschreibungen der PatternObjects abhängig vom jeweiligen SubPattern und spiegeln den beabsichtigten Rückgabewert der SubPattern.getShortDescription(PatternObject)-Methode für das jeweilige PatternObject wider.*

- *Häufig wird zu den SubPatterns ein Abschnitt ‚Bemerkungen' zu finden sein, in denen Anwendungshinweise gegeben werden. Diese sind ebenfalls für die betroffenen PatternObjects als applicationHint-Attribute in den Anlagen erfasst.*

6.3.1 Aufbau

6.3.1.1 XDS Affinity Domain

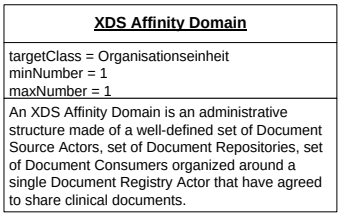

Abbildung 65 – SubPattern der XDS Affinity Domain

Abbildung 65 zeigt das SubPattern zur XDS Affinity Domain, welches nur aus dem gleichnamigen PatternNode besteht, der auf die Klasse Organisationseinheit abbildet. Jede Instanz, die von ihm abgeleitet ist, stellt dabei genau eine Affinity Domain dar, also einen Zusammenschluss von Einrichtungen zum Zwecke des Dokumentenaustausches. Die Organisationseinheiten, die diese Einrichtungen innerhalb eines 3LGM²-Modelles repräsentieren, sollten dazu über die ist_Teil_von-Beziehung jeder Affinity Domain untergeordnet werden, der sie angehören. Innerhalb eines Modelles sollte dabei nur genau eine Affinity Domain existieren.

6.3.1.2 XDS Actors

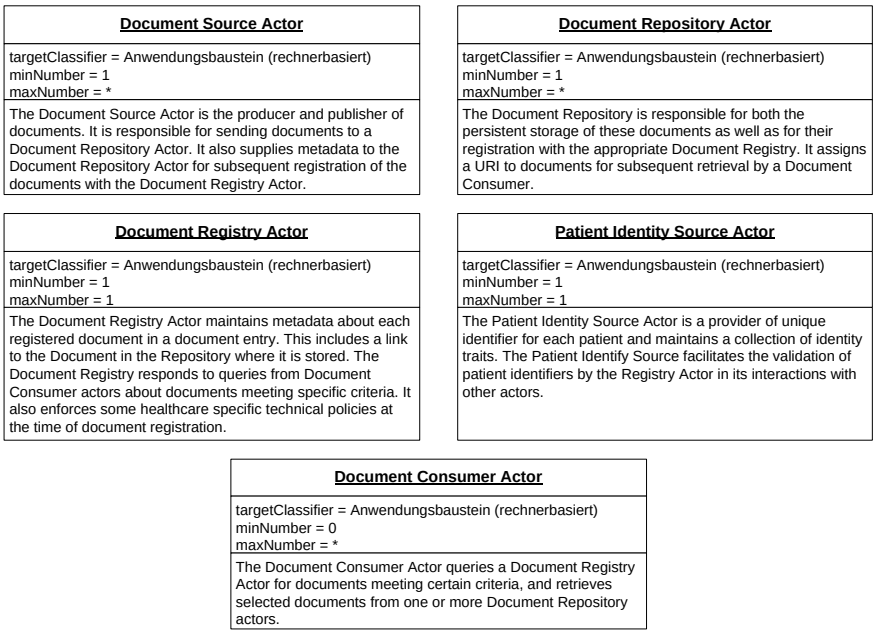

Abbildung 66 – SubPattern der Actors des XDS Integration Profil

Abbildung 66 zeigt eine Übersicht zu den Actors des XDS Integration Profile. Ein Document Source Actor ist Quelle von Dokumenten und Metadaten, wobei die Dokumente von Document Repositories und die Metadaten von einer zentralen Document Registry gespeichert werden. Innerhalb eines Modelles sollte es daher wenigstens einen Document Source, einen Document Repository und genau einen Document Registry Actor geben. Der Patient Identity Source Actor stellt für die gesamte Affinity Domain Patienten-IDs zur Verfügung, die er an die Document Registry übermittelt. Auch von ihm sollte nur genau einer existieren. Document Consumers greifen auf Metadaten in dieser Registry

zu und fragen Dokumente von Document Repositories ab. Von ihnen können beliebig viele innerhalb eines Modelles vorkommen.

6.3.1.3 Provide and Register Document Set-b Transaction

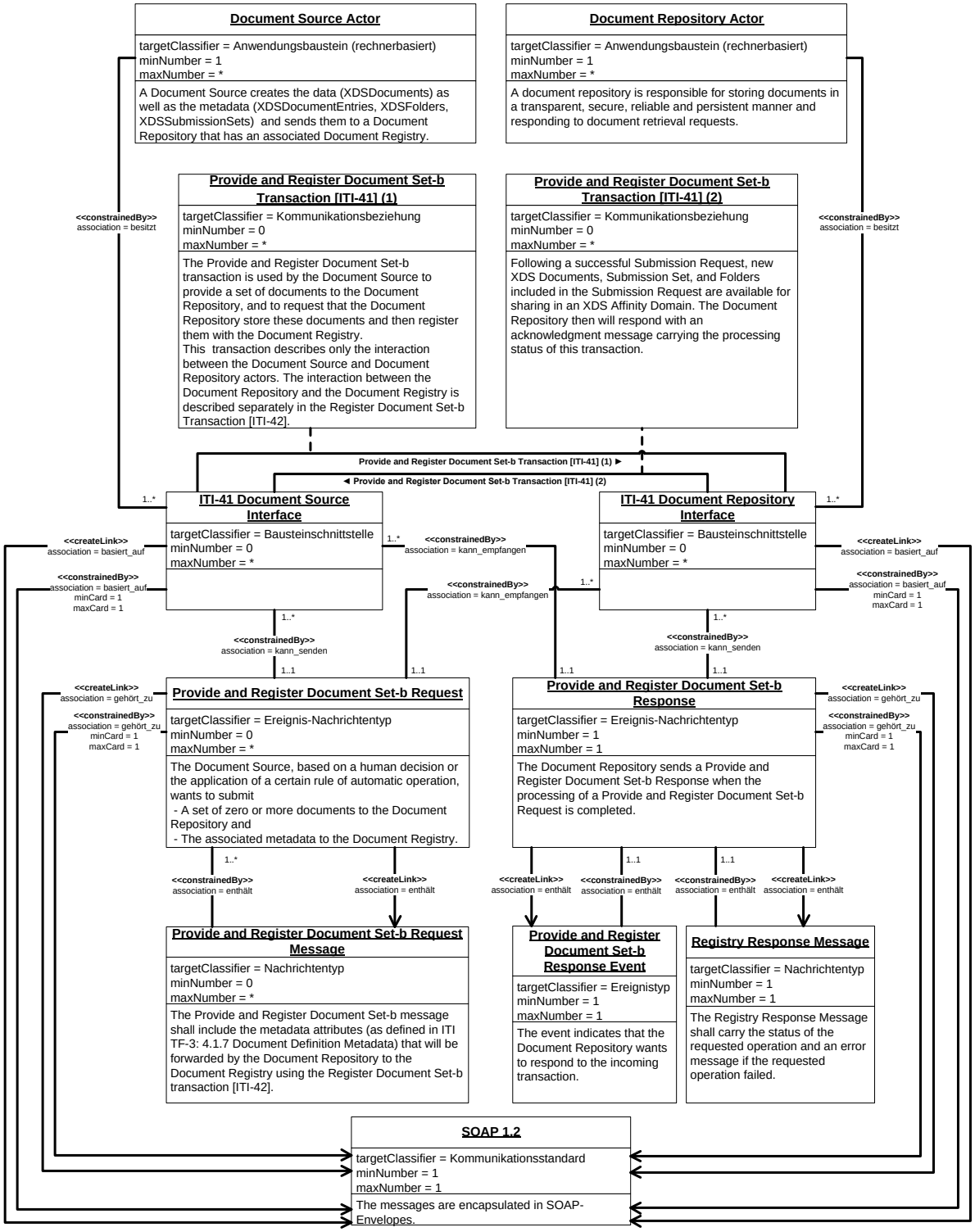

Abbildung 67 – SubPattern der Provide and Register Document Set-b Transaction

In Abbildung 67 ist das SubPattern der Provide and Register Document Set-b Transaction dargstellt, welches auf der abstrakten Vorlage aus Abbildung 62 basiert. Oben zu sehen sind Document Source und Document Repository Actor. Ein Document Source Actor darf Dokumente an beliebig viele solcher Repositories senden, was jeweils durch ein ITI-41 Document Source bzw. ITI-41 Document Repository Interface unter Verwendung einer Provide and Register Document Set-b Transaction [ITI-41] erreicht wird. Document Repository Actors können dabei Dokumente von beliebig vielen Document Source Actors empfangen. Deren Übertragung erfolgt in Form von Provide and Register Document Set-b Requests, welche in SOAP-Envelopes gekapselt werden. Jeder dieser Requests bekommt dabei eine Provide and Register Document Set-b Request Message zugewiesen, welche sowohl die Dokumente als auch die dazugehörige Metadaten enthält. Dabei kann eine solche Message in allen Requests verwendet werden, welche diese bestimmten Dokumente und Metadaten übertragen sollen. Analog kann jeder Request durch alle ITI-41 Interfaces gesendet bzw. empfangen werden, die eben diese Dokumente und Metadaten übertragen möchten. Die Interfaces basieren dazu auf SOAP. Das Document Repository ist dann dafür verantwortlich, die Dokumente persistent zu speichern und die Metadaten über eine Register Document Set-b Transaction an eine Document Registry weiterzuleiten, auf die im folgenden Abschnitt noch näher eingegangen wird. Abschließend erhält der Document Source Actor über einen Provide and Register Document Set-b Response eine Benachrichtigung, ob die Transaction erfolgreich durchgeführt werden konnte. Das Provide and Register Document Set-b Response Event zeigt dabei das Ende des Speicherns bzw. Registrierens der Dokumente bzw. Metadaten an. Der Status hierzu wird dann in der Provide and Register Document Set-b Response Message zurückgegeben, welche wiederum über einen SOAP-Envelope übertragen wird.

Bemerkungen:

Im Rückblick auf das abstrakte Pattern für IHE Transactions aus Abbildung 62 dürfte hier auffallen, dass ein Provide and Register Document Set-b Request Event fehlt. Es wurde hier absichtlich auf dessen Einführung verzichtet. Vielmehr sollte jeder von Provide and Register Document Set-b Request abgeleitete Ereignis-Nachrichtentyp mit einem (möglicherweise schon vorhandenen) Ereignistyp ausgestattet werden, der das Erzeugen des bereitzustellenden Dokumentes anzeigt und beispielsweise auch die einrichtungsinterne Kommunikation des Dokumentes auslöst. Diese Ereignistypen sind erwartungsgemäß nicht auf den Kontext des XDS Integration Profile beschränkt, sodass sie deshalb hier auch nicht als PatternObject erfasst werden. Das Verknüpfen der von Provide and Register Document Set-b Request abgeleiteten Ereignis-Nachrichtentypen mit diesen Ereignistypen sollte dabei über die enthält-Beziehung geschehen.

Das zweite Element des Request ist die Provide and Register Document Set-b Request Message. Von diesem PatternNode abgeleitete Nachrichtentypen müssen – entsprechend den Vorgaben der IHE – sowohl den Patienten selbst als auch die zu übertragenden Dokumente repräsentieren. Hierzu ist (falls noch nicht vorhanden) der Patient als Objekttyp auf der Fachlichen Ebene des 3LGM² anzulegen und über die wird_repräsentiert_durch-Beziehung mit ihm zu verknüpfen. Des Weiteren muss dieser Nachrichtentyp noch mit allen Objekttypen verbunden werden, welche den Inhalt der zu übertragenden Dokumente reflektieren (Es können mehrere Objekttypen sein, da Dokumente im Sinne des XDS Integration Profile potentiell zusammengesetzt sein dürfen). Der Verzicht auf PatternObjects, welche Patient und Dokumente repräsentieren, liegt ebenfalls darin begründet, dass diese keine auf XDS beschränkten Konzepte sind.

Sowohl das Verwenden gemeinsamer Ereignistypen als auch das Verbinden von Requests und Objekttypen gewährleistet die Integration von Pattern und Modell und sorgt damit für das Einbinden des Entwurfswissens in das so repräsentierte Informationssystem. Die Intention der hierbei verwendeten Kardinalitäten für die Constraints ist es, jeden Objekttyp bzw. jede Menge von Objekttypen, welche ein im Sinne von XDS übertragbares Dokument repräsentieren, durch genau eine Provide and Register Document Set-b Request Message darzustellen. Diese werden dann für jedes Ereignis, was deren Kommunikation auslöst, zu jeweils einem Provide and Register Document Set-b Request zusammengefasst, der an die Schnittstellen aller der Document Source Actors angefügt wird, welche bei Eintreten dieses Ereignisses genau dieses Dokument an ein Repository übermitteln. Im Tool3LGM² wird es dann durch die dort bereitgestellten Analysefunktionen ermöglicht, nachzuvollziehen, wo welche Informati-

onen zu einem Patienten für die Affinity Domain bereitgestellt werden. Für den Provide and Registry Document Set-b Response genügt es hierbei, nur eine Instanz im Modell zu haben, da über ihn keine Objekttypen übertragen werden.

6.3.1.4 Register Document Set-b Transaction

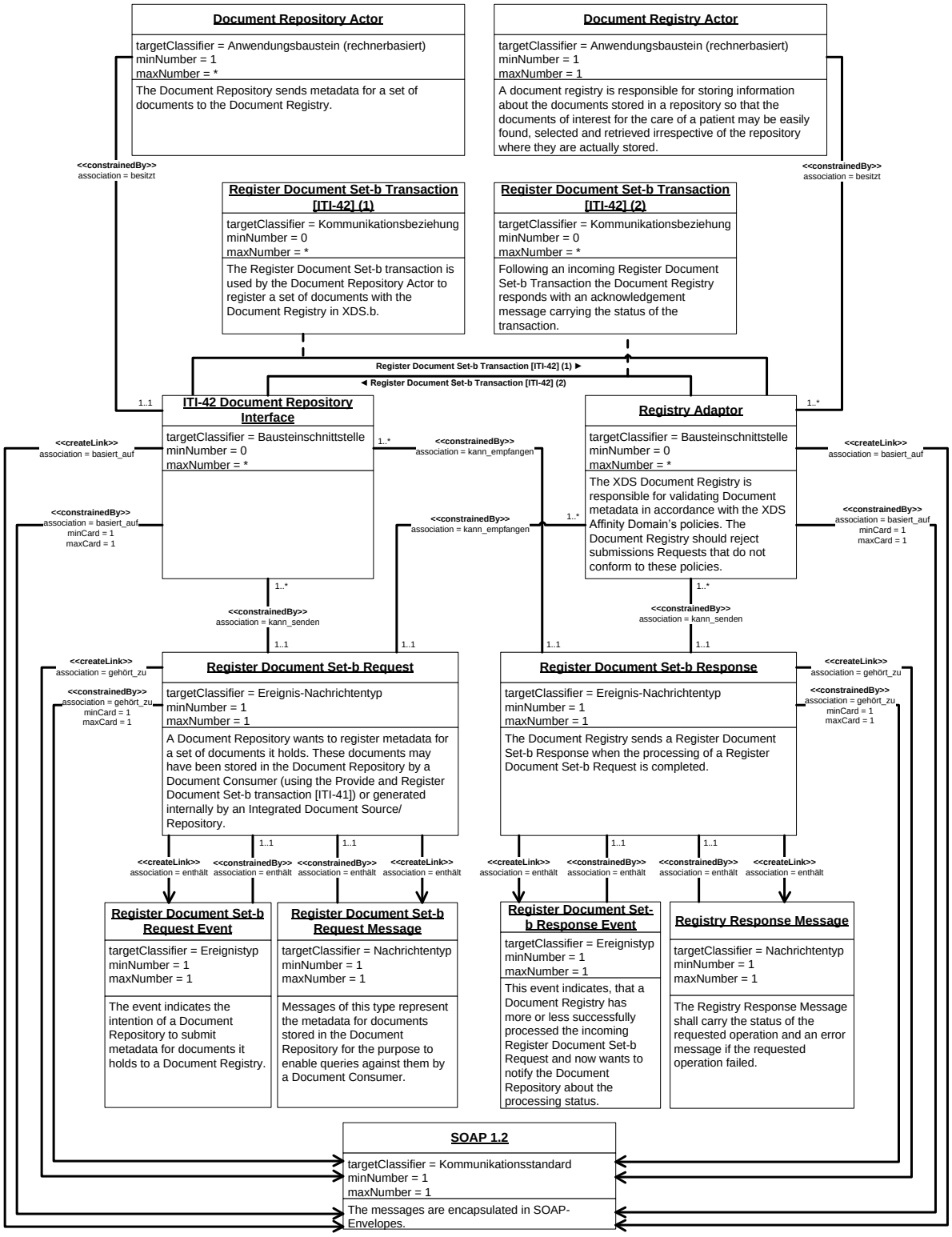

Abbildung 68 – SubPattern der Register Document Set-b Transaction

Hier zu sehen ist das SubPattern für die Registrierung von Dokumentmetadaten (Abbildung 68). Ein Document Repository überträgt dabei – etwa in Folge einer eingegangenen Provide and Register Document Set-b Transaction – Metadaten zu Dokumenten an die zentrale Document Registry. Dies erfolgt unter Einsatz einer Register Document Set-b Transaction [ITI-42] vom ITI-42 Document Repository Interface zum Registry Adaptor. Dieser erweitert die Funktionalität der Document Registry, indem er etwa die Validierung der eingehenden Metadaten vornimmt. Diese werden dabei in Form von Register-Document Set-b Request Messages übermittelt deren Übertragung durch ein Register Document Set-b Request Event ausgelöst wird. Hierzu werden beide in einem Register Document Set-b Request zusammengefasst, welcher über die Bausteinschnittstellen gesendet bzw. empfangen werden kann. Als Reaktion auf die Registrierung der Metadaten, welche durch das Register Document Set-b Response Event ausgedrückt wird, erfolgt eine Rückmeldung von der Document Registry an das sendende Document Repository. Diese ist durch eine Register Document Set-b Response Message repräsentiert, welche den Status der Registrierung trägt. Analog zum Request, werden sowohl der Ereignis- als auch der Nachrichtentyp zum Register Document Set-b Response zusammengefasst, welcher dann ebenfalls durch die Bausteinschnittstellen übertragen wird. Request und Response werden dabei in SOAP-Envelopes gekapselt. Um deren Übertragung zu ermöglichen, basieren daher auch beide ITI-42 Interfaces auf dem SOAP-Standard.

Bemerkungen:

Im Gegensatz zum Provide and Register Document Set-b Request aus Abbildung 67, zu dem es mehrere Instanzen geben durfte, darf ein Register Document Set-b Request nur ein einziges Mal innerhalb eines Modelles vorkommen. Dies liegt darin begründet, dass es mehrere übertragbare Dokumente (in Form von Objekttypen auf der fachlichen Ebene) gibt, für die jeweils ein eigener Provide and Register Document Set-b Request benötigt wird. Im Gegensatz dazu, existiert allerdings nur ein Objekttyp, der die Dokumentmetadaten (sprich den Patienten) repräsentiert. Der Register Document Set-b Request als Überträger dieser Metadaten ist damit auch nur einmal anzulegen. Analog verhält sich dies mit dem dazugehörigen Request Event und der dazugehörigen Request Message. Letztere ist dann über die wird_repräsentiert_durch-Beziehung mit eben diesem Objekttypen zu verbinden. (Im Gegensatz zur Provide and Register Document Set-b Transaction wurde hier ein auslösendes Ereignis – das Register Document Set-b Request Event – spezifiziert, da das Senden von Dokumentmetadaten von einem Document Repository Actor zu einem Document Registry Actor ein XDS-spezifisches Ereignis darstellt.)

Analog zur Provide and Register Document Set-b Transaction genügt es auch hier, den Register Document Set-b Response einmal pro Modell repräsentiert zu haben, da auch er nur eine Bestätigungsnachricht darstellt.

Anzumerken bleibt noch, dass jeder Document Repository Actor nur genau einen assoziierten Document Registry Actor besitzen darf, an den er alle Metadaten sendet. Dies ist der Grund für die ‚1..1'-Kardinalität des Constraints zum ITI-42 Document Repository Interface.

6.3.1.5 Registry Stored Query Transaction

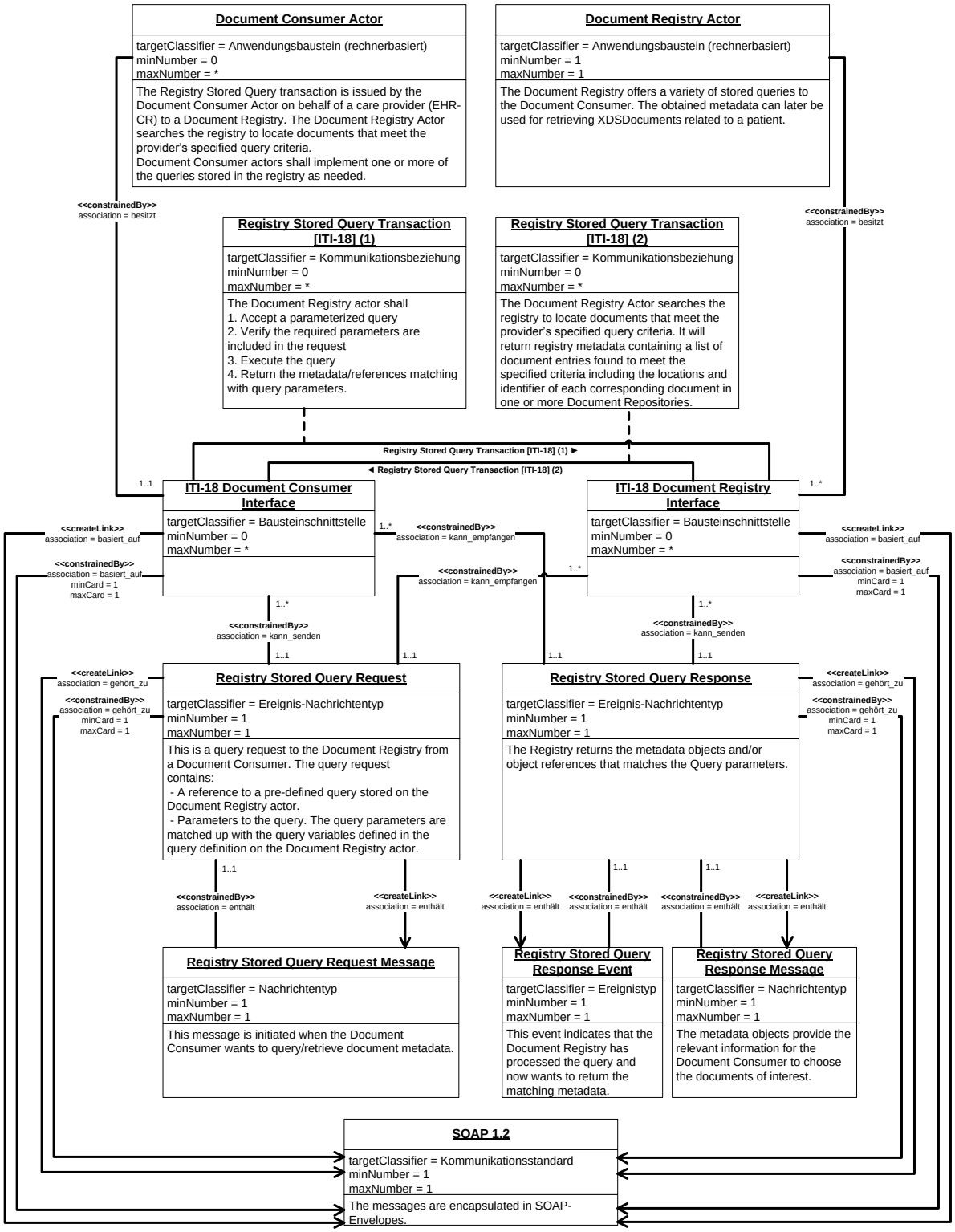

Abbildung 69 – SubPattern der Registry Stored Query Transaction

Nach einer erfolgreichen Bereitstellung und Registrierung von Dokumenten stehen diese nun allen Einrichtungen in der Affinity Domain zur Verfügung, welche dann jeweils in der Rolle eines Document Consumer die Document Registry nach ihnen durchsuchen können. Dies erfolgt durch einen Registry Stored Query Request, der mittels einer Registry Stored Query Transaction [ITI-18] über beide ITI-18 Interfaces vom Document Consumer zur Document Registry übertragen wird (Abbildung

69). Diese beginnt daraufhin, entsprechend der durch den Document Consumer in der Registry Stored Query Message festgelegten Query-Parametern, passende Dokumentmetadaten zu suchen und gibt diese dann über einen Registry Stored Query Response zurück, was durch Auslösen eines Registry Stored Query Response Event angezeigt wird. Die Metadaten werden dabei in einer Registry Stored Query Response Message gekapselt. Als Kommunikationsstandard wird wiederum SOAP verwendet.

Bemerkungen:

Für die Registry Stored Query Transaction stellt die Document Registry einen Satz vordefinierter Queries bereit, deren Parameter vom Document Consumer mit entsprechenden Werten belegt werden. Hierzu gehört vor allem die Patienten-ID. Die Registry Stored Query Request Message bzw. der dazugehörige Nachrichtentyp als Überträger der Query-Parameter muss dazu, analog zu den Request Messages in den vorangegangenen Transactions, über die wird_repräsentiert_durch-Beziehung mit dem Objekttyp verbunden werden, der den Patienten im Informationssystem repräsentiert. Eine derartige Suche bezieht sich dann auf jeweils genau einen Patienten.

Es wurde hier darauf verzichtet, ein Registry Stored Query Request Event zu definieren (vgl. Abbildung 62). Typischerweise entsteht ein Bedarf nach bereits erstellten Dokumenten, wie etwa Vorbefunde, Arztbriefe etc., immer im Zusammenhang mit Ereignissen im Behandlungsverlauf eines Patienten, wie zum Beispiel das Eintreffen in der jeweiligen Einrichtung oder auch der Beginn der Behandlungsplanung. Genau in solchen Fällen wird die Registry Stored Query Transaction ausgelöst. Da es sich allerdings hierbei nicht um XDS-spezifische Ereignisse handelt, werden diese auch nicht als PatternObject repräsentiert. Der Patternnutzer sollte vielmehr die Ereignistypen wählen, die bereits im Modell zur Repräsentation derartiger Ereignisse verwendet werden – gegebenenfalls müssen diese neu angelegt werden. Sie sind dann über die enthält-Beziehung mit den vom Registry Stored Query Request abgeleiteten Ereignis-Nachrichtentypen zu verknüpfen.

6.3.1.6 Retrieve Document Set Transaction

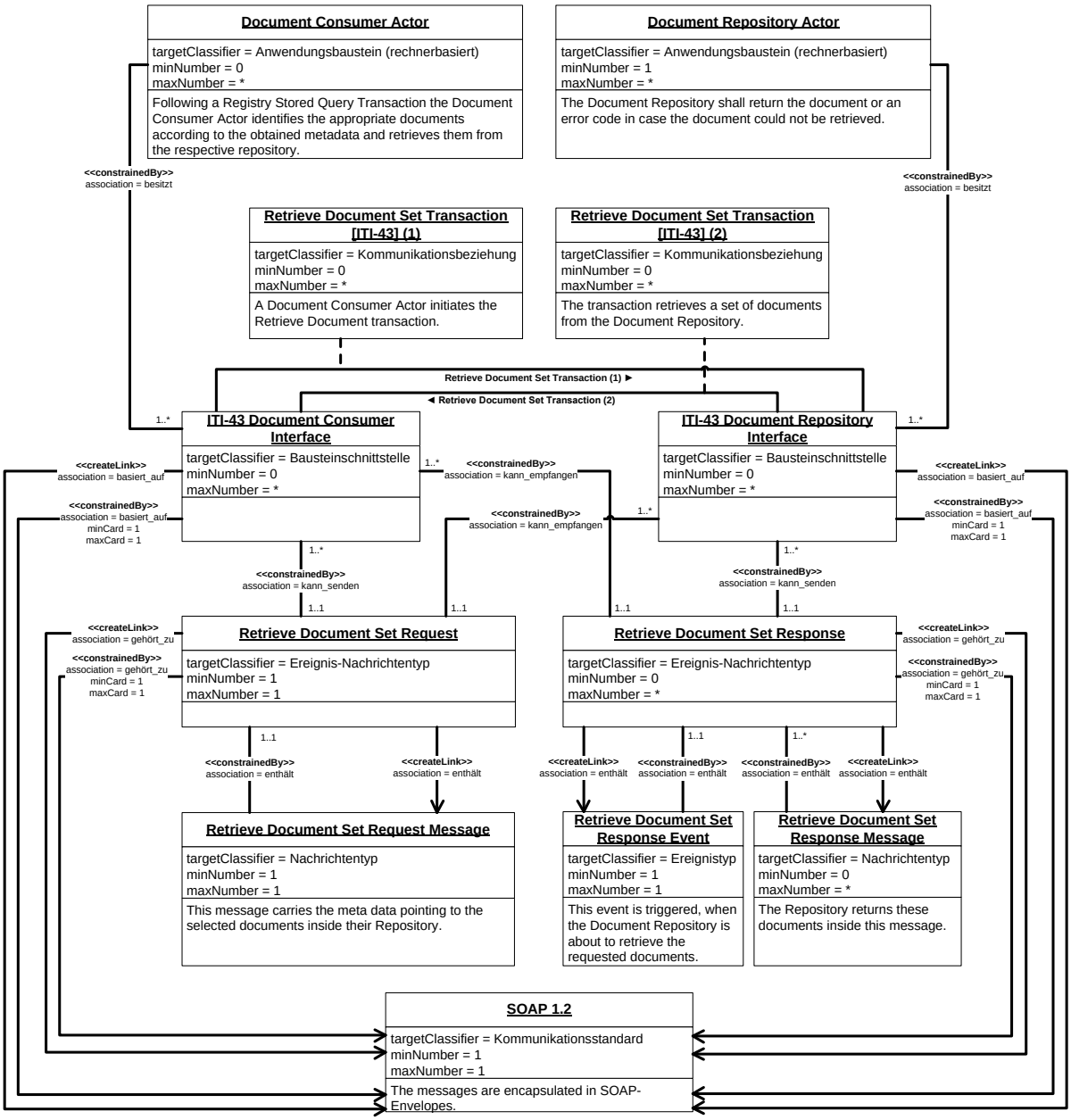

Abbildung 70 – SubPattern der Retrieve Document Set Transaction

Auf Basis der gefundenen Metadaten aus einer Registry Stored Query Transaction ist der Document Consumer nun in der Lage, die gewünschten Dokumente zu identifizieren und wird daraufhin damit beginnen, diese von dem entsprechenden Document Repository abzufragen. Hierzu verwendet er die Angaben zu Repository und Dokument-IDs aus dem vorangegangenen Registry Stored Query Response und überträgt diese in einen Retrieve Document Set Request, welcher über beide ITI-43 Interfaces unter Einsatz der Retrieve Document Set Transaction an das entsprechende Repository übermittelt wird (Abbildung 70). Die Retrieve Document Set Request Message übernimmt dabei die Referenzen zu Repository und Dokumenten. Das betroffene Document Repository reagiert daraufhin mit einem Retrieve Document Set Response, der durch das Retrieve Document Set Event ausgelöst wird und die dem Request entsprechenden Dokumente in Form einer Retrieve Document Set Response Message zurückliefert. Es ist dabei Aufgabe des Document Repositories, die Dokumente unverändert von ihrer Bereitstellung durch den Document Source Actor wiederzugeben. Sowohl Request als auch Response

Message sind dabei wieder in SOAP-Envelopes gekapselt. Ebenfalls basieren die Interfaces auf diesem Kommunikationsstandard.

Bemerkungen:

Analog zum Provide and Register Document Set-b Request, welcher die bereitzustellenden Dokumente repräsentierte, soll auch der Retrieve Document Set Response bzw. die Retrieve Document Set Response Message eben die Dokumente repräsentieren, die vom jeweiligen Document Consumer abgefragt werden können. Hierzu ist zwischen den von ihr abgeleiteten Nachrichtentypen und den zu eben diesen Dokumenten gehörigen Objekttypen jeweils eine Verknüpfung über die wird_repräsentiert_durch-Beziehung anzulegen. Im Modell wird nun damit – etwa durch den Einsatz von Analysen im Tool3LGM² – erkennbar, wo welche Dokumente bereitgestellt und wiederum abgefragt werden.

Die Auswahl von Dokumenten mittels einer Retrieve Document Set Transaction erfolgt dabei analog zu deren Suche über eine Registry Stored Query Transaction im Zusammenhang mit Ereignissen im Verlauf der Behandlung eines Patienten. Damit ist auch hier das Wegfallen eines Retrieve Document Set Request Event begründet. Jeder Ereignis-Nachrichtentyp, welcher von Retrieve Document Set Request abgeleitet ist, sollte daher mit genau dem Ereignistyp versehen werden, welcher auch den vorangegangenen Registry Stored Query Request auslöst. Dies sollte über die enthält-Beziehung geschehen.

6.3.1.7 Patient Identity Feed Transaction – Part 1

Vorbemerkungen:

Die Patient Identity Feed Transaction wurde aus Gründen der Übersichtlichkeit in zwei SubPatterns zerlegt. Part 1 zeigt den allgemeinen Nachrichtenaustausch, Part 2 die Struktur der Nachrichten sowie die auslösenden Ereignisse.

Es wird darüber hinaus noch eine kompaktere Darstellung von DirectConstraints und CreationLinks verwendet. Mehrere DirectConstraints bzw. mehrere CreationLinks teilen sich dabei eine gemeinsame Linie, welche sich am constraintSource- oder constraintTarget-Ende bzw. creationSource oder creationTarget-Ende in die einzelnen DirectConstraints bzw. CreationLinks aufgliedert. Dies kam allerdings nur in den Fällen zur Anwendung, wo die association-Attribute übereinstimmten und keine Widersprüche in den Kardinalitäten am gemeinsamen Ende existierten, um die Eindeutigkeit wahren zu können. Etwa bei der „<<constrainedBy>>“-Verknüpfung zwischen den PatternNodes der Ereignis-Nachrichtentypen („ACK“, „A01_ADT“, ...) mit dem PatternNode „HL7 2.3.1“ handelt es sich um insgesamt sechs DirectConstraints – einer pro Ereignis-Nachrichtentyp – welche für dieselbe Assoziation definiert sind. (In diesem Fall gibt es keine Kardinalitäten an den constraintSource-Enden, da es sich hierbei um unidirektionale DirectConstraints handelt, wodurch damit auch keine Widersprüche auftreten können.)

Beide SubPatterns weichen rein optisch von der Standardvorlage aus Abbildung 62 ab, beinhalten aber prinzipiell dieselben Typen von Elementen: Die PatternNodes A01_ADT, A04_ADT, A05_ADT, A08_ADT sowie A40_ADT sind Requests, welche über die ITI-8 Interfaces vom Patient Identity Feed Actor zum Document Registry Actor übertragen werden. Der PatternNode ACK ist dabei der Response zu diesen Requests. Das zweite SubPattern gliedert dann die Requests und den Response in die jeweiligen PatternNodes für die Ereignis- und Nachrichtentypen auf.

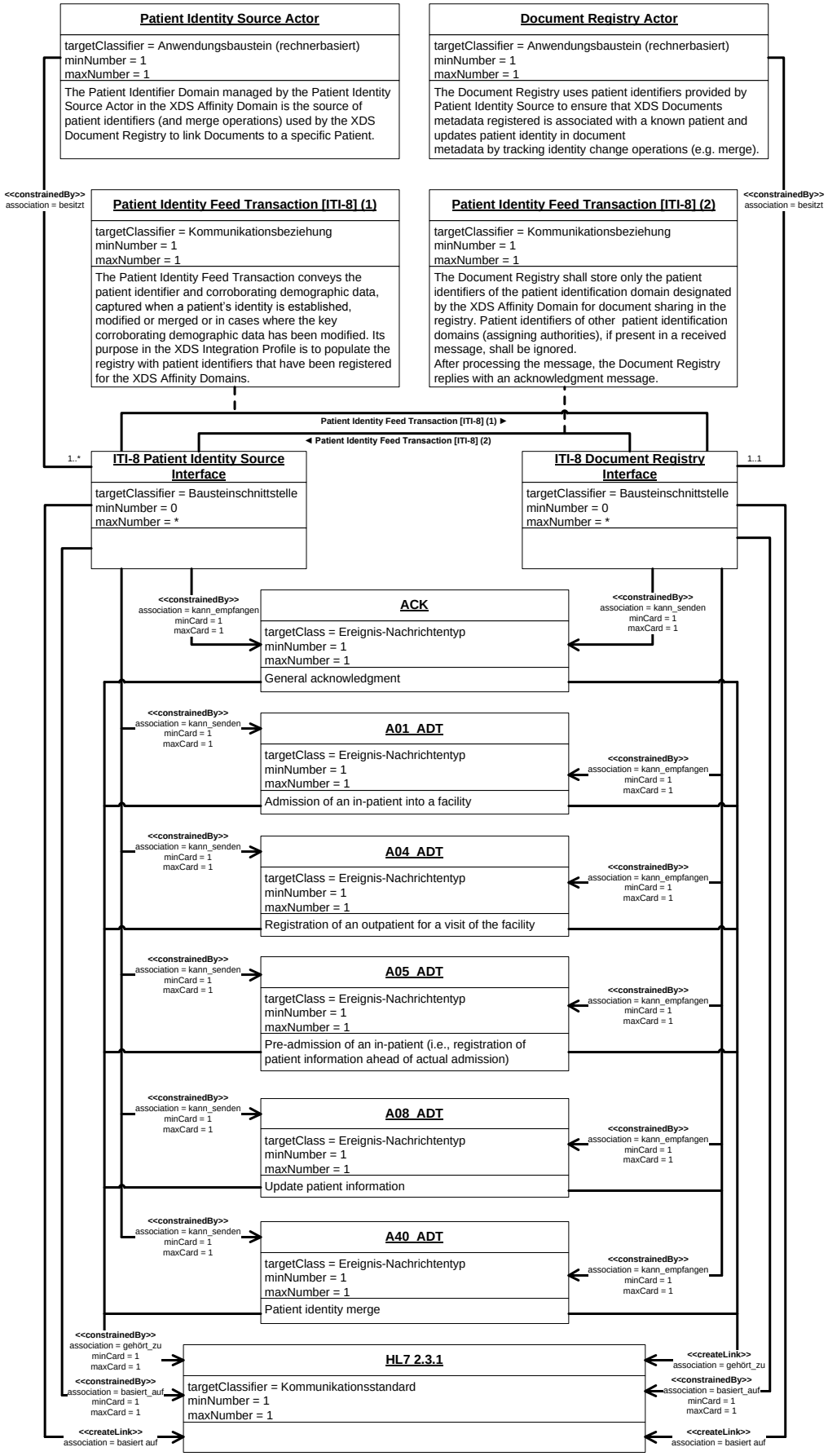

Abbildung 71 – SubPattern der Patient Identity Feed Transaction (Teil 1)

Die Document Registry beinhaltet neben Dokumentmetadaten auch Patienteninformationen, wie etwa Patienten-ID, Name, Geburtsdatum etc., zum Zweck der Verknüpfung von Dokumenten und Patienten sowie der Möglichkeit zu deren Verifikation durch einen Document Consumer. Der Patient Identity Source Actor versorgt dazu die Document Registry mit derartigen Patienteninformationen für die dazugehörige Affinity Domain. Dies erfolgt durch die Patient Identity Feed Transaction [ITI-8], welche die Informationen über die ITI-8 Interfaces überträgt. Die Document Registry antwortet daraufhin mit einer Bestätigungsnachricht. Der Nachrichtenaustausch basiert dabei auf dem HL7 Kommunikationsstandard.

Bemerkungen:

ITI-8 Interfaces: Ein Patient Identity Source Actor darf mehrere ITI-8 Patient Identity Source Interfaces besitzen, da er potentiell der Patientenidentifikation in verschieden Affinity Domains dienen kann. Eine Document Registry ist hingegen auf eine Affinity Domain beschränkt und besitzt daher auch nur eine ITI-8 Document Registry Interface.

6.3.1.8 Patient Identity Feed Transaction – Part 2

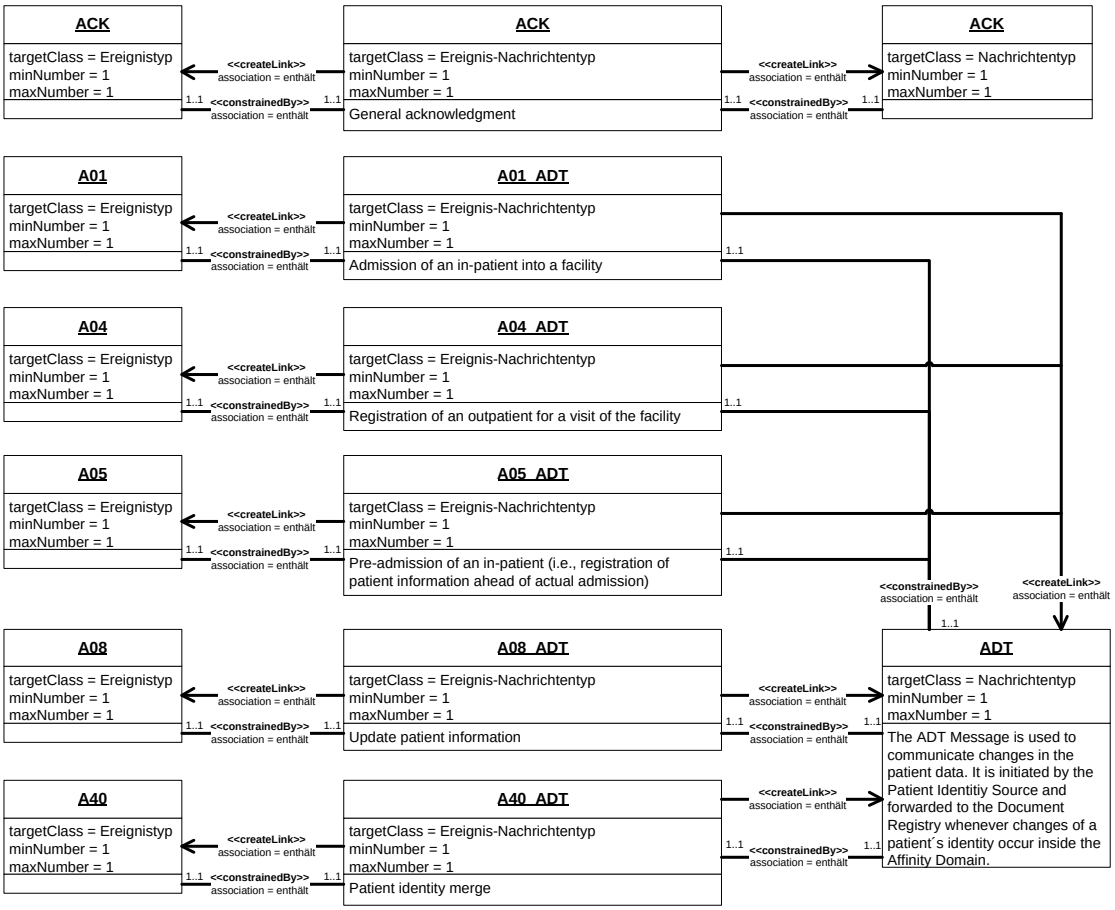

Abbildung 72 – SubPattern der Patient Identity Feed Transaction (Teil 2)

Die Patient Identity Feed Transaction erlaubt das Übertragen mehrerer Ereignis-Nachrichtentypen, was im Vorhandensein verschiedener Ereignisse im Zusammenhang mit der Patientenidentifikationsverwaltung begründet ist. Etwa A01 bezeichnet die stationäre Aufnahme eines Patienten, A08 eine Aktualisierung von Informationen zu einem Patienten. Diese Ereignistypen sind aus der HL7 Spezifikation abgeleitet und werden gemeinsam mit einem ADT-Nachrichtentyp zu Ereignis-

Nachrichtentypen zusammengefasst. Die Struktur der Nachricht ist dabei vom auslösenden Ereignis abhängig. Die entsprechenden Konventionen hierfür sind in den Anlagen bereitgestellt.

Nach dem Erstellen einer neuen Patientenidentität oder Änderungen an einer bereits bestehenden wird die Document Registry über den Patient Identity Source Actor mittels des entsprechenden Ereignis-Nachrichtentyps hierüber informiert. Der ADT-Nachrichtentyp trägt dabei die Patienteninformationen, nach deren Verarbeitung die Document Registry mit einer Bestätigungsnachricht reagiert – repräsentiert über den ACK-Nachrichtentyp und ausgelöst durch den ACK-Ereignistyp.

Eine Document Registry kann nun Dokumente und Patienten referenzieren, wodurch die Grundvoraussetzung für den Aufbau der Affinity Domain geschaffen wurde.

Bemerkungen:

Der Nachrichtentyp ADT sollte als Träger der Patienteninformationen mit dem Objekttyp auf der Fachlichen Ebene des 3LGM² über die wird_repräsentiert_durch-Beziehung verknüpft werden, welcher den Patienten repräsentiert.

Das Konzept der HL7-basierten Kommunikation ist nicht auf das XDS Integration Profile beschränkt, wurde hier aber ins Pattern übernommen, da zum einen zusätzliche Einschränkungen durch die IHE definiert worden sind und zum anderen eine Übersicht im Pattern dazu notwendig ist, wie Dokumente und Patienten referenziert werden können.

6.3.1.9 Federated Document Source/Repository

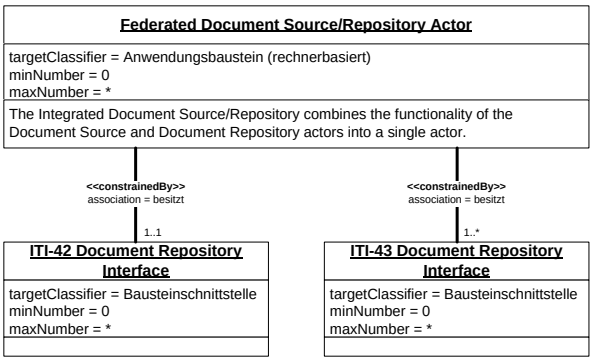

Abbildung 73 – SubPattern des Federated Document Source/Repository Actor

In Abbildung 73 ist der sogenannte Federated Document Source/Repository Actor zu sehen, welcher im XDS Integration Profile eingeführt wurde, um den Fall abzudecken, in dem eine Einrichtung bzw. ein Anwendungssystem gleichzeitig als Erzeuger und Bereitsteller von Dokumenten auftritt. In Bezug auf die XDS Affinity Domain stellt er damit nach außen hin ein Document Repository dar und muss demnach auch dessen Schnittstellen implementieren. Diese sind identisch mit denen in 6.3.1.4 und 6.3.1.6. Dabei hat sich hier die Dokumenterzeugung – im Gegensatz zum Document Source – in eine rein interne Funktionalität gewandelt und ist damit im Sinne von XDS irrelevant geworden. Ein Federated Document Source/Repository Actor sollte deshalb auch nicht die Provide and Register Document Set-b Transaction unterstützen.

6.3.2 Anwendung

Alle eben gezeigten SubPatterns ergeben gemeinsam das Konstruktionspattern des XDS Integration Profile der IHE. Hierbei ist, wie oben bereits erwähnt, zu beachten, dass zur Vervollständigung noch die nicht dargestellten Attribute in den Anlagen hinterlegt wurden. Im Hinblick auf eine Implementierung des 3LGM²-CF sind diese dann entsprechend für die Laufzeitinstanzen der Komponenten des Construction Framework zu übernehmen. Vorschläge für eine derartige Umsetzung wurden in 4.5 für das Tool3LGM² gegeben. So wäre es dann möglich, nach den eben gemachten Vorgaben das Pattern im Tool anzulegen, womit es anschließend für beliebig viele Informationssystemmodelle geladen und angewendet werden kann. Dabei erfährt der Modellierer eine Lenkung hin zu einem Pattern-konformen Modell, etwa unterstützt durch die Erweiterung des Kontextmenüs sowie den Möglichkeiten zur Konsistenzprüfung. Darüber hinaus können durch die gleichzeitig bereitgestellten Beschreibungen die Modelle direkt als Quelle von technischen Details und Implementierungshinweisen verwendet werden.

Ein typisches Beispiel für die Anwendung wäre nun ein Modell im Tool3LGM², welches die Informationssysteme einer oder mehrerer Einrichtungen repräsentiert, die im Sinne von XDS Dokumente austauschen möchten. Der Modellierer, wird dazu das Konstruktionspattern laden, wodurch es jetzt im Tool verfügbar ist. Entsprechend der in 4.5.2 geschilderten Möglichkeiten für die Anwendung von Patterns wird er nun beginnen, den Dokumentenaustausch zu modellieren. Dazu wird er die XDS Affinity Domain als Organisationseinheit instanziieren, einen zentralen Document Registry und Patient Identity Source Actor sowie mehrerer Document Repository Actors anlegen und sie über die entsprechenden Interfaces und Transactions verbinden. Dabei ist, durch die im Pattern definierten DirectConstraints, dafür gesorgt, dass tatsächlich auch die richtigen Beziehungen angelegt werden. Des Weiteren erfolgt nun ein Zuweisen der jeweiligen Requests und Responses. Dabei kann den durch sie bereitgestellten Anwendungshinweisen entnommen werden, inwiefern eine Verknüpfung zur Fachlichen Ebene hergestellt werden soll, sodass ein Einbinden in das bestehende Modell gewährleistet wird. Darüber hinaus kann ein händisches Zuordnen eines Kommunikationsstandards entfallen, da es durch die im Pattern definierten CreationLinks bereits automatisch geschieht. Ebenso erfolgt dies für die Zuweisung von Requests bzw. Responses und den jeweiligen Request bzw. Response Events und Messages. Hierdurch wird eine deutliche Beschleunigung des Modellierungsprozesses erzielt eben durch das automatische Anlegen bestimmter Beziehungen. In einem nächsten Schritt werden nun noch die Anwendungsbausteine identifiziert, die Dokumente bereitstellen bzw. diese abfragen möchten und entsprechend vom Document Source bzw. Document Consumer Actor abgeleitet. Auch für sie werden die entsprechenden Transactions inklusive der benötigen Interfaces, Requests und Repsonses angelegt, wobei der Modellierer hier gleichermaßen durch die DirectConstraints zu einem Pattern-konformen Modell gelenkt wird. Auf diese Weise kann der Dokumentenaustausch nach XDS im Tool3LGM² mit Unterstützung des Patterns modelliert werden.

Über die so eröffneten Perspektiven der Nutzbarmachung dieses Wissens für die Modellierung ist nun eine geeignete Unterstützung für die Planung eines solchen transinstitutionellen Informationssystems geschaffen worden. Der erleichterte Zugang zu den zugrundeliegenden Konzepten sorgt dabei für eine Beschleunigung der Planungsaktivitäten und enthebt den Modellierer von der Notwendigkeit einer intensiven Einarbeitung. Neben einer Vereinfachung des Modellierungsprozesses ist auch die Durchführung der so geplanten Änderungen leichter möglich, da im Modell direkt Informationen über die benötigten Kommunikationsstandards, Austauschformate und sonstigen technischen Anforderungen abrufbar sind. Dabei ist zugleich die Sicherheit gegeben, bewährte Ansätze für die Umsetzung zu nutzen.

Auf diese Weise konnte ein Schritt in hin zum transinstitutionellen Dokumentenaustausch im Gesundheitswesen gemacht werden. Auf Basis einer entsprechenden Umsetzung ist es dann möglich, patientenbezogene Informationen über Einrichtungsgrenzen hinweg auszutauschen. Hierdurch kann am dem Ort, wo die Behandlung eines Patienten erforderlich ist, eine bessere Entscheidungsunterstützung erzielt und damit zur Qualität der Patientenversorgung beigetragen werden.

7 Zusammenfassung

Informationsverarbeitung ist ein Qualitätsfaktor, welcher alle Stufen ausgehend von den Strukturen, über die Prozesse bis hin zum Erreichen der Ziele eines Unternehmens durchzieht. Es ist dabei Aufgabe des Informationsmanagements, ein systematisch funktionierendes Informationssystem zu etablieren und dieses an den Unternehmenszielen auszurichten. Strategische Veränderungen, wechselnde Rahmenbedingungen oder auch technologischer Fortschritt lösen dabei einen Änderungsbedarf am Informationssystem aus, was dessen Anpassung an diese neuen Umstände erforderlich macht.

Im Bereich der Informationssysteme im Gesundheitswesen gilt es dabei, insbesondere das Ziel der Unterstützung einer bestmöglichen Patientenversorgung zu verfolgen, was durch effiziente Informationsprozesse erreicht wird. Hierfür sind zunächst strukturelle Grundlagen zu schaffen, wie etwa die Datenintegrität oder auch die Angemessenheit von Anwendungssystemen an die zu unterstützenden Aufgaben. Um dabei den Qualitätsansprüchen Genüge zu tun, ist eine systematische Planung notwendig, zu der durch den Einsatz geeigneter Methoden beigetragen werden kann. Eine effektive Möglichkeit hierfür ist die Modellierung. Durch sie kann eine abstrakte übersichtliche Darstellung des Informationssystems erreicht werden, was die Voraussetzungen für eine analytische Bewertung sowie die Identifikation und Planung der notwendigen Änderungen schafft. Hierfür hat sich das 3LGM² als geeignet gezeigt, welches durch das Aufgliedern des Informationssystems in drei Ebenen die Modellierung von sowohl logischer Aufgabenunterstützung und Datenkommunikation als auch deren Realisierung durch physische Komponenten ermöglicht (siehe Kapitel 2). Darüber hinaus besitzt es eine softwareseitige Umsetzung in Form des Tool3LGM², welches den praktischen Einsatz des Metamodelles gewährleistet.

Als weiteres Mittel für die Unterstützung der Planungsaktivtäten wurde der Einsatz von Entwurfswissen identifiziert, welches bewährte und erprobte Konzepte für die Architektur von Informationssystemen beinhaltete. Insbesondere im Kontext des Gesundheitswesens rückte dabei die IHE in den Blickpunkt des Interesses, deren Hauptaugenmerk auf der Integration von Anwendungssystemen liegt, um der Heterogenität in Informationssystemen entgegenzuwirken. Dazu wurde auf den Einsatz und die Adaption bewährter Standards, wie etwa HL7, DICOM oder auch SOAP, fokussiert, um eine bessere Zusammenarbeit von Computersystemen zu erreichen. Ziel dabei war es, eine adäquate Unterstützung der Prozesse zu erreichen und zwar durch das Schaffen der jeweils notwendigen Strukturen. Die IHE stellt dazu derartiges Wissen über Technical Frameworks bereit, welche sich in Integration Profiles untergliedern, die jeweils einen bestimmten Problembereich hinsichtlich dessen Integration adressieren. Die Technical Frameworks bzw. Integration Profiles beschreiben dabei nicht nur allgemeine Grundlagen zur Lösung der jeweiligen Problemstellung. Sie geben darüber hinaus noch in detaillierter Weise Informationen für die konkrete Anwendung von Kommunikationsstandards, wie sie oben bereits aufgeführt worden sind, bis hin zu der Struktur bestimmter Nachrichtenformate für den Austausch von Daten zwischen den Anwendungssystemen. Hierbei ergab sich allerdings die Schwierigkeit, dieses Entwurfswissen nur begrenzt für die Planungsaktivtäten einsetzen zu können. Ausgelöst wurde das Problem durch die hohe Komplexität und Unübersichtlichkeit der Zusammenhänge zwischen Grundlagen und technischen Spezifikationen, was durch die textuelle Form noch zusätzlich kompliziert wurde.

Es stellte sich daher die Frage, ob es eine bessere Möglichkeit gibt, derartiges Wissen für die Planung von Informationssystemen einzusetzen. Im Rückblick auf die Vorteile der Modellierung war es nur naheliegend, eben sie als Ansatz hierfür zu verwenden, mit dem Ziel, eine bessere Übersichtlichkeit und Transparenz der dargestellten Sachverhalte zu erreichen. Hierbei galt es, zwei wesentliche Aspekte zu beachten: den der Deskriptivität und den der Konstruktivität. Deskriptivität bezeichnete dabei die Formalisierung des Entwurfswissens als Modell selbst; Konstruktivität die Nutzbarmachung dieser Modelle für die Modellierung der Informationssysteme. Somit war es erforderlich, übersichtliche Darstellung und die Fähigkeit zur Abbildung in Informationssystemmodelle zu kombinieren.

Die Lösung zu dieser Problemstellung wurde dann über die *Konstruktionspatterns* erreicht (siehe 3.3). Der Begriff selbst ist dabei mit Patterns, wie sie aus dem Bereich der Softwareentwicklung bekannt sind, insoweit verwandt, als dass im Mittelpunkt die Dokumentation bewährter Konzepte aus erprob-

ten Entwürfen steht, bei ihnen allerdings nicht auf die Anwendung einer Programmiersprache, sondern einer Modellierungssprache abgezielt wird. Um den Ansprüchen der Deskriptivität und Konstruktivität zu genügen, wurden dazu Konstruktionspatterns zum einen selbst als Modelle eingeführt und zum anderen für sie die Definition einer *Interpreterfunktion* vorgenommen, die deren Abbildung auf die Modellierungssprache bzw. das Metamodell der Informationssystemmodelle ermöglicht. Diese Funktion sorgt dafür, dass jedes Element und jede Beziehung in einem Pattern wiederum als Vorlage für Elemente und Beziehungen in einem solchen Modell dienen. Hierfür ist allerdings eine Vorbedingung vonnöten, damit dies gelingen kann: Bei dem Metamodell bzw. dessen Elementen und Beziehungen muss es sich tatsächlich um instanziierbare Konzepte, sprich Klassifizierungen handeln, wie es beispielsweise beim Klassendiagramm der UML gegeben ist. In solch einem Fall würden dann die Elemente und Beziehungen eines Patterns auf Klassen und Assoziationen im Metamodell abgebildet werden und sind dann Vorlagen für die dazugehörigen Instanzen in konkreten Modellen. Der Hintergrund ist dabei, abstrakte Strukturen im Pattern definieren zu können, einschließlich Restriktionen bezüglich der Beziehungen, die Instanzen in Modellen miteinander eingehen dürfen. Dies dient dazu, den Konstruktionsprozess von Modellen so zu lenken, dass diese mit den Vorgaben im Entwurfswissen übereinstimmen. Die Interpreterfunktion ermöglicht es dabei, Elemente und Beziehungen eines Modelles von Vorlagen in einem Pattern abzuleiten. Diese werden dabei als Instanzen der Klassifizierung im Metamodell angelegt, auf die die Interpreterfunktion die jeweilige Vorlage aus dem Pattern abgebildet. Für diese Instanzen kommen dann die eben erwähnten Restriktionen zur Anwendung, welche zur Konformität des Modelles mit dem Pattern bzw. dem so repräsentierten Entwurfswissen beitragen. Ein Modellierer kann somit durch die Verwendung von Patterns dahingehend entlastet werden, dass eine intensive Einarbeitung in das Entwurfswissen und eine „Übersetzung“ der darin enthaltenen Konzepte in das Metamodell entfallen. Demnach ergibt sich nur ein einmaliger Aufwand durch das Erzeugen des Patterns. Wissen auf diese Weise formalisiert, kann daraufhin beliebig oft bei der Erstellung von Modellen eingesetzt werden und ist damit effizient hinsichtlich der Wiederverwendbarkeit.

Im Hinblick auf die Nutzbarmachung von Entwurfswissen für die Modellierung von Informationssystemen im Gesundheitswesen sollte es nun zu einer Anwendung von Konstruktionspatterns kommen. Hier galt es, zunächst ein geeignetes Metamodell für die Repräsentation derartiger Informationssysteme zu wählen, was den oben aufgeführten Anforderungen bezüglich der Instanziierbarkeit genügt. Die Wahl fiel dabei auf das 3LGM², da es zum einen der Dokumentation von Informationssystemen im Gesundheitswesen dient und gleichzeitig in Form eines UML Klassendiagrammes definiert ist (siehe 2.1.1.1, 2.1.2.1 & 2.1.3.1). Ziel war es nun, zunächst die Erstellung 3LGM²-konformer Konstruktionspatterns zu unterstützen. Da es sich bei ihnen gemäß ihrer Definition wiederum selbst um Modelle handelt, ist hierfür eine Modellierungssprache erforderlich gewesen. Dazu wurde das *3LGM² - Construction Framework (3LGM²-CF)* vorgestellt (siehe Kapitel 4). Dieses dient der Modellierung derartiger Patterns sowie deren Abbildung auf das 3LGM² und ist dabei speziell auf eine mögliche Implementierung im Tool3LGM² ausgerichtet.

Das 3LGM²-CF basiert auf den Grundbausteinen *PatternNode*, *PatternRelation* sowie *DirectConstraint* und *CreationLink* (siehe 4.4.3). PatterNodes und PatternRelations sind dabei Vorlagen für Instanzen von Klassen bzw. Assoziation oder Assoziationsklassen im 3LGM². Über sie kann Entwurfswissen in abstrakten Strukturen formalisiert werden, indem darin enthaltene Konzepte als Klassifizierung im 3LGM² interpretiert werden. DirectConstraints ermöglichen es dann, Einschränkungen zu definieren, welche Beziehungen diese Instanzen untereinander eingehen dürfen. Ein Modell muss dabei alle diese Constraints einhalten, um konform mit einem Pattern sein zu können. CreationLinks dienen abschließend dazu, das automatische Anlegen bestimmter Elemente und Verknüpfungen auszulösen, insbesondere in den Fällen, wo gewisse Zusammenhänge zwingend im Modell vorhanden sein müssen. Über sie wird vor allem auch eine Beschleunigung des Modellierungsprozesses erreicht.

Für PatternNodes und PatternRelations ist es des Weiteren möglich, verschiedene Attribute zu definieren. Von großer Bedeutung ist dabei das Beschreibungsattribut (*„description“*). Dieses ermöglicht das Anfügen wichtiger Informationen aus dem Entwurfswissen in textueller Form und dient damit dem tieferen Verständnis des Patterns. Ein weiterer Hintergrund für das Definieren dieses Attributes ist die Tatsache, dass auch das Tool3LGM² das Anlegen von Beschreibungstexten für Elemente und Bezie-

hungen anbietet. Da PatternNodes und PatternRelations als Vorlagen für Instanzen in Modellen dienen, können deren Beschreibungsattribute für diese Elemente und Beziehungen übernommen werden. Demnach ist es so möglich, vordefinierte Beschreibungen anzufertigen und dann auf konkrete Instanzen zu übertragen. Auf diese Weise kann also auch das Wissen, was textuell im Pattern erfasst wurde, in ein Modell gelangen. Damit sind Modellnutzer dann in der Lage, über die Elemente und Beziehungen darauf zuzugreifen, wodurch nun auch sie, neben dem Modellierer, von dem so formalisierten Wissen profitieren.

Auf Basis des 3LGM²-CF können jetzt Konstruktionspatterns für Informationssystemmodelle im 3LGM² definiert werden. Die jeweilige Abbildung vom Pattern in das Metamodell wurde hierbei durch eine inhärente Interpreterfunktion bewerkstelligt: PatternNodes bzw. PatternRelations korrespondieren zu jeweils einer Klasse bzw. Assoziation oder Assoziationsklasse im 3LGM². Dies wird allerdings nicht durch das Angeben einer Funktion im eigentlichen Sinne realisiert. Stattdessen ist für jeden PatternNode und jede PatternRelation ein Attribut definiert, welches die Klassifizierung im 3LGM² angibt, auf die sie abgebildet werden. Dieses ist dabei jeweils durch den Patternmodellierer festzulegen. Analog funktioniert dies für DirectConstraints und CreationLinks: Diese beziehen sich jeweils auf eine Assoziation im Metamodell. Dazu erhalten auch sie ein Attribut, in dem eben diese festgelegt werden kann. Die Gesamtheit dieser Attribute in einem Pattern repräsentiert dann die dazugehörige Interpreterfunktion. Über sie können nun Konstruktionspatterns in 3LGM²-Modellen interpretiert und angewendet werden. Das 3LGM² - Construction Framework sieht dabei eine gezielte Aufteilung von Patternerstellung und Patternanwendung vor (siehe 4.2 & 4.3):

Patternmodellierer recherchieren im Allgemeinen Entwurfswissen, welches relevant ist im Bezug auf Informationssysteme im Gesundheitswesen und formalisieren es in Konstruktionspatterns. Typische Beispiele hierfür sind etwa die oben schon angesprochenen IHE Integration Profiles. Unabhängig davon ergeben sich Szenarien für die Anwendung von Patterns beim Aufkommen von Änderungsbedarf in konkreten Informationssystemen. Dies kann etwa ausgelöst sein durch neue strategische Ausrichtungen oder auch Defizite der Informationsverarbeitung selbst. Unter Verwendung der Modellierung auf Basis des 3LGM² für die jetzt notwendigen Planungsaktivitäten kann es zu einem Einsatz von Patterns kommen. Dabei ist die Grundvoraussetzung, dass Entwurfswissen über die durchzuführende Weiterentwicklung des Informationssystems existiert und durch einen Patternmodellierer in einem Konstruktionspattern erfasst wurde. Durch die Definition mittels 3LGM²-CF können diese im 3LGM² interpretiert werden und ermöglichen es so, das Wissen auf das Modell des Informationssystems zu übertragen. Damit wird der Modellierer durch die darin definierten Vorgaben und Restriktionen hin zu einem korrekten Modell gelenkt und hat dabei die Sicherheit, auf bewährtes Wissen zu bauen. Um allerdings einen solchen praktischen Einsatz von Konstruktionspatterns zu erzielen, ist eine softwaremäßige Umsetzung des 3LGM²-CF erforderlich. Da es für das 3LGM² bereits eine Implementierung in Form des Tool3LGM² gibt, war es naheliegend, diese Umsetzung in Form einer Erweiterung des Tools zu konzipieren. Hierfür wurde eine technische Spezifikation des 3LGM²-CF gegeben und die Möglichkeiten für eine Realisierung im Tool3LGM² aufgezeigt.

Auf Basis dieser Anforderungen an die Patterndefinition wurde die Umsetzung für das Tool3LGM² konzipiert. Diese unterteilt sich in Patternerstellung und Patternanwendung. Für das Erstellen von Patterns wurden dazu unter anderem Modifikationen der grafischen Ansichten und Menüs vorgeschlagen (siehe 4.5.1). Dabei sind sowohl Wege für das Anlegen von PatternNodes, PatternRelations, DirectConstraints und CreationLinks aufgezeigt worden als auch Möglichkeiten für das Editieren von Attributen (siehe 4.5.1.3). Nach dem Anlegen eines Patterns kann dieses dann gespeichert werden und steht nun für die Anwendung in 3LGM²-Modellen zur Verfügung. Um dies zu unterstützen, wurden auch hier die notwendigen Änderungen am Tool3LGM² aufgezeigt. So soll es möglich sein, während des Modellierens ein Pattern laden zu können, was dann im Modell verfügbar gemacht wird. Dazu ist neben dem typischen Erzeugen von Elementen und Beziehungen das Anlegen von Instanzen als Ableitungen von PatternNodes oder PatternRelations zur Verfügung zu stellen. Dabei handelt es sich dann jeweils um Instanzen der Klassifizierung im 3LGM², auf die der PatternNode bzw. die PatternRelation durch die Interpreterfunktion abgebildet wird. Demnach hängt dieser Punkt entscheidend vom Patternmodellierer ab, der diese Interpretation des Entwurfswissens im 3LGM² definiert hat. Bei der Ableitung einer Instanz von einem PatternNode oder einer PatternRelation kommen nun auch die DirectConstraints zum Einsatz. Diese beschränken die möglichen Beziehungen, die Instanzen mitei-

nander eingehen können. Eine Verletzung dieser Vorgaben bedeutet dabei eine Inkonsistenz des Modelles. Das Tool3LGM² sollte dazu die Funktionalität bereitstellen, dies zu erkennen, wie es in 4.5.3 gezeigt wurde. Hierdurch wird insbesondere dafür gesorgt, dass der Modellierer die Probleme erkennen und beheben kann. Somit wird zur Konformität des Modelles mit dem Pattern beigetragen und damit auch zur Übereinstimmung mit dem zugrundeliegenden Entwurfswissen. Des Weiteren erfährt der Modellierer eine Unterstützung durch die im Pattern definierten CreationLinks, welche das automatische Anlegen bestimmter Elemente und Verknüpfungen auslösen, etwa im Fall zwingend notwendiger Beziehungen im Modell (siehe 4.5.4).

Neben diesen strukturellen Vorgaben eines Konstruktionspatterns erfolgt auch die Anwendung textuell erfasster Informationen. Hierzu werden die Beschreibungsattribute von PatternNodes und PatternRelations als Beschreibungstexte aller Instanzen im Modell gesetzt, die von ihnen abgeleitet sind. Auf die so übernommenen Informationen kann daraufhin jederzeit über die Instanzen zugegriffen werden.

Somit ist nun gezeigt worden, wie eine Umsetzung des 3LGM²-CF möglich ist. Das Tool3LGM² kann so in die Lage versetzt werden, das Erstellen und Anwenden von Konstruktionspatterns zu unterstützen. Durch die Spezifikation des 3LGM²-CF ist dabei das Zusammenführen von Modellierung und Entwurfswissen gelungen, was durch die entsprechende Implementierung im Tool3LGM² für einen praktischen Einsatz zugänglich gemacht werden kann.

Eine wichtige Anwendung des 3LGM²-CF sind dabei die Technical Frameworks bzw. Integration Profiles der IHE (siehe 5.1). Diese stellen umfangreiches Entwurfswissen zu Informationssystemen im Gesundheitswesen bereit, insbesondere im Hinblick auf die Integration von Computersystemen, sind aber aufgrund der oft hohen Komplexität nur bedingt einsetzbar für die erforderlichen Planungsaktivitäten. Dies wurde speziell beim *XDS Integration Profile* deutlich, welches sich mit dem transinstitutionellen Austausch patientenbezogener Dokumente befasst. Kerngedanke ist dabei der Aufbau einer *XDS Affinity Domain* als Verbund medizinischer Einrichtungen zum Zwecke des Dokumentenaustausches. Als Ergebnis entsteht so eine ganzheitlichen Krankenakte (*„Longitudinal Health Record“*), welche die Dokumentationen der einzelnen Einrichtungen für einen Patienten vereinigt und zugänglich macht (siehe 6.2). Durch die Verfügbarkeit dieser umfassenden Informationen wird so eine potentiell bessere Versorgung gewährleistet. Damit repräsentiert das XDS Integration Profile eine wichtige Weiterentwicklung von Informationssystemen im Gesundheitswesen. Durch die bereits erwähnte hohe Komplexität und Unübersichtlichkeit sind allerdings die Einsatzmöglichkeiten für die Planung eingeschränkt. Im Hinblick auf die hohe Relevanz dieses Integration Profile ist es dennoch unabdingbar, jenes Wissen auf geeignete Weise für diese Weiterentwicklung von Informationssystemen verfügbar zu machen. Daher wurde eine Transformation mittels des 3LGM²-CF vorgenommen. Als Ergebnis ist dabei ein Konstruktionspattern erzielt worden, welches das XDS Integration Profile in 3LGM²-konformer Weise abbildet, die strukturellen Zusammenhänge darstellt und darüber hinaus umfassende Beschreibungen beinhaltet (siehe 6.3.1). Auf Basis einer Implementierung des 3LGM²-CF im Tool3LGM² nach den obigen Vorgaben kann dann anhand dieser formalen Transformation des XDS Integration Profile das entsprechende Konstruktionspattern eben im Tool3LGM² angelegt werden. Daraufhin ist dieses für die Modellierung in beliebig vielen Modellen einsetzbar und trägt somit zu einer erleichterten Planung und Umsetzung des transinstitutionellen Dokumentenaustausches im Gesundheitswesen bei.

Durch das Errichten eines transinstitutionellen Informationssystems mit standardisierten Kommunikationsschnittstellen zwischen den beteiligten Einrichtungen wird es möglich, einen effizienten Dokumentenaustausch bereitzustellen, welcher eine bessere Anpassung an die Informationsbedürfnisse bei Entscheidungen bezüglich der Versorgung eines Patienten bietet. Doppelbehandlungen und mehrfache Datenerfassung, welche kosten- und zeitintensiv sind, können so vermieden werden, was zu einer Steigerung der Effizienz in der Erbringung medizinischer Dienstleistungen führt. Durch den Einsatz bewährter Kommunikationsstandards ergibt sich darüber hinaus noch ein weiterer wichtiger Vorteil: Für Einrichtungen, die übereinkommen, patientenbezogene Informationen zu teilen und auszutauschen, genügt jeweils die einmalige Implementation der benötigten Schnittstellen. Danach ist eine Kommunikation von Dokumenten zwischen all diesen Einrichtungen möglich, was vor allem dem Entstehen mehrerer bilateraler Kommunikationsbeziehungen entgegenwirkt, bei welchem jeweils zwischen zwei Kommunikationspartnern eine Neuentwicklung der Schnittstellen und Nachrichtenformate erforderlich ist. Dies bedeutete einen stetig anwachsenden Aufwand für die Umsetzung und wäre

damit ineffizient hinsichtlich der Kosten. Durch den Einsatz standardisierter Methoden hingegen kann dieses Problem umgangen werden, wodurch insbesondere auch das Anbinden weiterer Versorgungseinrichtungen deutlich vereinfacht wird.

8 Diskussion

Das XDS Integration Profile beschreibt die notwendigen Strukturen für den einrichtungsübergreifenden Austausch patientenbezogener Dokumente. Es wird dabei auf eine geeignete Unterstützung der ablaufenden Prozesse abgezielt – etwa die Suche nach Dokumenten über eine Registry. Hierzu identifiziert das XDS Integration Profile Einrichtungen bzw. Computersysteme als Actors, die damit eine gewisse Rolle in diesem Kontext einnehmen. Für deren geeignete Interaktion werden Transactions definiert, welche jeweils eine standardisierte Kommunikation bestimmter Daten beschreiben. Auf diese Weise wird zur Interoperabilität der Computersysteme beigetragen. Diese Kombination aus Actors und Transactions findet allerdings nicht nur im XDS Integration Profile Anwendung, sondern bildet vielmehr die Basis aller Integration Profiles für jedes Technical Framework (siehe 5.1). Als Vorbereitung für die Transformation des XDS Integration Profile in ein Konstruktionspattern wurden zunächst diese Konzepte dem 3LGM² gegenübergestellt. Dabei zeigte sich, dass dort äquivalente Konzepte zu Actor und Transaction existieren, in Form des *rechnerbasierten Anwendungsbausteines* und der *Kommunikationsbeziehung*. Auf Basis dieser Überlegungen konnte so ein Schema entwickelt werden, Integration Profiles im Allgemeinen auf das 3LGM² abbilden zu können. Dabei handelt es sich um ein abstraktes Konstruktionspattern, welches unabhängig vom konkreten Integration Profile die entsprechenden Strukturen über das 3LGM²-CF repräsentiert (siehe 5.2). Eine Anwendung hierfür stellte das XDS Integration Profile dar, dessen Transformation in ein Konstruktionspattern auf genau dieser abstrakten Vorlage basierte. Nun ist es denkbar, dieses Prinzip auf andere Integration Profiles auszuweiten. Dabei kann dann wiederum das abstrakte Konstruktionspattern angewendet werden, wodurch sich insbesondere die Möglichkeit ergibt, dessen Fähigkeiten für die Transformation von IHE Profilen im Allgemeinen weiter zu überprüfen. Potentiell ergibt sich so die Chance, ein schnelleres Überführen von IHE Konzepten in Patterns zu erreichen und damit auch eine Steigerung der Verfügbarkeit von Entwurfswissen für die Modellierung von Informationssystemen im Gesundheitswesen.

Unter Betrachtung der Technical Frameworks ist bei ihnen in einigen Fällen von der „Gruppierung von Actors“ [1] die Rede, wie es etwa im ITI Technical Framework dargestellt wird. Gemeint ist damit, dass die Funktionalität mehrerer Actors in einem einzigen vereinigt wird. Computersysteme in der Rolle eines solchen zusammengesetzten Actor müssen dann alle für sie definierten Transactions unterstützen. Im Hinblick auf die Spezifikation des 3LGM² - Construction Framework wurde dort allerdings die implizite Vorannahme gemacht, eine ‚Eins zu eins‘-Beziehung zwischen Konzepten im Entwurfswissen und 3LGM² zu haben. Dies ermöglicht es, einzelne Integration Profiles abzubilden. Die allgemeine Definition eines Konstruktionspatterns bzw. der dazugehörigen Interpreterfunktion lässt aber auch eine mehrfache Zuweisung zu (siehe 3.3). Im Falle von IHE und 3LGM² würde dies etwa bedeuten, dass ein rechnerbasierter Anwendungsbaustein von mehreren Actors abgeleitet sein dürfte. Es wäre daher denkbar, eine Erweiterung des 3LGM²-CF vorzunehmen, welche die Fähigkeit einer mehrfachen Ableitung bereitstellt. Hierbei ergibt sich allerdings die Schwierigkeit, Widersprüche auflösen zu müssen: Das 3LGM²-CF ermöglicht etwa durch DirectConstraints die Definition von Restriktionen hinsichtlich der Beziehungen in einem Modell. Diese kamen dabei für die Instanzen zur Anwendung, welche von einer Vorlage im Pattern – also von einem PatternNode oder einer PatternRelation – abgeleitet wurden. Dürfen nun diese Instanzen von mehreren Vorlagen aus (potentiell verschiedenen) Patterns abgeleitet sein, kann es zu gegensätzlichen Vorgaben kommen, welche Beziehungen sie untereinander besitzen können. Im Falle der Gruppierung von IHE Actors ist zwar davon auszugehen, dass derartige Restriktionen konsistent sind, im Allgemeinen könnten sich aber dennoch Widersprüche ergeben. Die Erweiterung des 3LGM²-CF müsste daher in der Lage sein, diese

[1] Vgl. [ITI-TF] - 2.1 Dependencies among Integration Profiles.

aufzulösen bzw. eine Vorgabe dahingehend zu machen, wie Implementierungen des Construction Framework mit ihnen umgehen sollen. Dazu zählt insbesondere das Abgleichen verschiedener Kardinalitäten, welche für die DirectConstraints definiert worden sind. Hieraus ergäbe sich dann die Möglichkeit, etwa das Gruppieren von IHE Actors in Modellen abbilden zu können, indem die entsprechenden Anwendungsbausteine von allen erforderlichen Actors abgeleitet werden.

Als Alternative zur eben beschriebenen Erweiterung des 3LGM²-CF wäre es auch möglich, ein analoges Verfahren auf Patternebene zu realisieren. Um zu dem Beispiel des Gruppierens von IHE Actors zurückzukehren, ist es dort vorstellbar, dieses bereits im Pattern durchzuführen. Dabei würden diese Actors zu einem neuen PatternNode vereinigt, welcher alle Beziehungen und Constraints von ihnen übernimmt. Nutzer eines Patterns würden so dahingehend entlasten werden, dass nur noch ein einmaliges Ableiten im Modell von diesem zusammengesetzten Actor notwendig ist. Nichtsdestotrotz gilt es auch hier, Widersprüche in den Constraints aufzulösen. Dies erfolgt jetzt allerdings nicht bei der Anwendung eines Patterns, sondern bereits bei dessen Erstellung. Auch hierfür sollte eine Implementierung des 3LGM²-CF, wie oben bereits beschrieben, die notwendige Funktionalität anbieten. Neben dem Auflösen gegensätzlicher Constraints ist es hier des Weiteren erforderlich, die vordefinierten Beschreibungsattribute der PatternNodes, welche es zu vereinigen gilt, abzugleichen. Für die Nutzer des Patterns entfällt damit der Aufwand, dies bei den Instanzen auf Modellebene durchführen zu müssen.

Durch derartige Erweiterungen könnten so die Einsatzmöglichkeiten des 3LGM²-CF vergrößert und damit eine breitere Unterstützung für das Abbilden von Entwurfswissen erreicht werden. Grundsätzlich wird aber für einen praktischen Einsatz immer eine Implementierung notwendig sein, wie es etwa für das Tool3LGM² gezeigt wurde. Daraufhin ist es möglich, das hier vorgestellte Konstruktionspattern des XDS Integration Profile über das Tool3LGM² anzulegen und für die Planungsaktivitäten zum transinstitutionellen Dokumentenaustausch einzusetzen. Wie an diesem Beispiel deutlich wurde, ist das 3LGM² - Construction Framework ein geeignetes Mittel, Entwurfswissen für die Modellierung mit dem 3LGM² verfügbar zu machen und bietet darüber hinaus das Potenzial der Erweiterung durch zusätzliche Funktionalität.

Literaturverzeichnis

WINTER et al. 2011:

WINTER A, HAUX R, AMMENWERTH E, BRIGL B, HELLRUNG N, JAHN F. Health Information Systems: Architectures and Strategies. 2nd ed. London: Springer; 2011.

LEHMANN et al. 2005:

LEHMANN T, editor. Handbuch der Medizinischen Informatik. 2nd ed. München: Hanser; 2005.

3LGM² TOOL:

3LGM² Tool [Internet]. Leipzig: Institut für Medizinische Informatik, Statistik und Epidemiologie (IMISE); [cited 2011 Jun 27]. [3LGM²] - Baukasten. Available from: http://www.3lgm2.de/Baukasten/index.jsp.

WINTER et al. 2003:

WINTER A, BRIGL B, WENDT T. Modeling Hospital Information Systems (Part 1): The Revised Three-layer Graph-based Meta Model 3LGM². Methods of Information in Medicine. 2003;42(5):544-551.

LEINER et al. 2003:

LEINER F, GAUS W, HAUX R, KNAUP-GREGORI P, PFEIFFER KP. Medizinische Dokumentation: Grundlagen einer qualitätsgesicherten integrierten Krankenversorgung. 4th ed. Stuttgart: Schattauer; 2003.

OMG CORBA:

Common Object Request Broker Architecture (CORBA) [Internet]. Needham (Massachusetts): Object Management Group, Inc.; c1997-2011 [cited 2011 Apr 5]. CORBA/IIOP Specification. Available from: http://www.omg.org/technology/documents/corba_spec_catalog.htm.

IHE TF:

Technical Frameworks [Internet]. IHE International; c2011 [cited 2011 Mar 10]. IHE.net Technical Frameworks. Available from: http://www.ihe.net/Technical_Framework/index.cfm.

DONABEDIAN 1982:

DONABEDIAN A. The definition of quality and approaches to its assessment: Explorations in quality assessment and monitoring. 2nd ed. Ann Arbor (Michigan): Health Administration Press; 1982.

HL7:

HL7 [Internet]. Health Level Seven International; c2007-2011 [cited 2011 Apr 3]. HL7 Standards. Available from: http://www.hl7.org/implement/standards/index.cfm?ref=nav.

DICOM:

DICOM [Internet]. Rosslyn (Virginia): The Association of Electrical and Medical Imaging Equipment Manufacturers (NEMA); c2011 [cited 2011 Apr 3]. DICOM Homepage. Available from: http://medical.nema.org/.

EB XML:

ebXML [Internet]. Burlington (Massachusetts): Organization for the Advancement of Structured Information Standards (OASIS); c2011 [cited 2011 Apr 3]. ebXML – Enabling A Global Electronic Market. Available from: http://www.ebxml.org/.

XDS:

XDS Integration Profile [Internet]. IHE International; c2011 [cited 2011 May 13]. IHE.net IHE Profiles. Available from: http://www.ihe.net/profiles/index.cfm.

3LGM²:

3LGM² [Internet]. Leipzig: Institute for Medical Informatics, Statistics and Epidemiology (IMISE); [cited 2011 Jun 29]. [3LGM²] - Metamodell. Available from: http://www.3lgm2.de/Metamodell/index.jsp.

SCHÜTTE 1998:

SCHÜTTE R. Grundsätze ordnungsgemäßer Referenzmodellierung: Konstruktion konfigurations- und anpassungsorientierter Modell. Wiesbaden: Gabler; 1998.

BROCKE 2003:

VOM BROCKE J. Referenzmodellierung: Gestaltung und Verteilung von Konstruktionsprozessen. Advances in Information Systems and Management Science (*vol 4*). Berlin: Logos; 2003.

MALICH 2007:

MALICH S. Qualität von Softwaresystemen: Ein pattern-basiertes Wissensmodell zur Unterstützung des Entwurfs und der Bewertung von Softwarearchitekturen [Dissertation]. Universität Duisburg-Essen; 2007.

UML IS:

OMG Unified Modeling Language (OMG UML), Infrastructure v2.3 [Internet]. Needham (Massachusetts): Object Management Group, Inc.; c1997-2011 [cited 2011 Jul 01]. UML 2.3. Available from: http://www.omg.org/spec/UML/2.3/.

Literaturverzeichnis

UML SU:

OMG Unified Modeling Language (OMG UML), Superstructure v2.3 [Internet]. Needham (Massachusetts): Object Management Group, Inc.; c1997-2011 [cited 2011 Jul 01]. UML 2.3. Available from: http://www.omg.org/spec/UML/2.3/.

MOF:

Meta Object Facility (MOF) Core Specification v2.0 [Internet]. Needham (Massachusetts): Object Management Group, Inc.; c1997-2011 [cited 2011 Jul 01]. MOF 2.0. Available from: http://www.omg.org/spec/MOF/2.0/.

XML NS:

Namespaces in XML 1.0 (Third Edition) [Internet]. World Wide Web Consortium (W3C); c2011 [cited 2011 Jun 30]. Namespaces in XML 1.0 (Third Edition). Available from: http://www.w3.org/TR/xml-names/.

XML:

Extensible Markup Language (XML) 1.0 (Fifth Edition) [Internet]. World Wide Web Consortium (W3C); c2011 [cited 2011 Jun 30]. Extensible Markup Language (XML) 1.0 (Fifth Edition). Available from: http://www.w3.org/TR/xml-names/.

ITI-TF:

IT-Infrastructure Technical Framework - Revision 7.0 [Internet]. IHE International; c2011 [cited 2011 Jul 3]. IHE.net Technical Frameworks. Available from: http://www.ihe.net/Technical_Framework/index.cfm.

Weitere Quellen (nicht im Text referenziert)

Towards More Integrated Implementation of Healthcare Information Systems:

HUSSEIN R, WINTER A. Towards More Integrated Implementation of Healthcare Information Systems: Using the 3LGM² for modeling the IHE-Scheduled Workflow Integration Profile. IEEE Computer Society. 2008;32:650-652.

Agfa Healthcare ORBIS:

[Internet] Available from: http://www.agfa.com/germany/de/he/solutions/krankenhausweite_it/technologien/cool/index.jsp.

SNOMED CT:

[Internet] Available from: http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html.

Geschäftsmodelle für Grid Computing in der Medizin und Biomedizin:

SCHOLZ S. Geschäftsmodelle für Grid Computing in der Medizin und Biomedizin [Dissertation]. Universität Hannover; 2009.

Enzyklopädie der Wirtschaftsinformatik:

[Internet] Available from: http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Softwarearchitektur/Wiederverwendung-von-Softwarebausteinen/Referenzmodell.

JDOM Element:

[Internet] Available from: http://www.jdom.org/docs/apidocs/index.html?org/jdom/Element.html.

Object Constraint Language (OCL) v2.2:

[Internet] Available from: http://www.omg.org/spec/OCL/2.2.

9 Anlagen

Anlagenverzeichnis

9.1 Anlage I – Bezeichnungen und Beschreibungen zum XDS Konstruktionspattern

Im Folgenden werden die verbleibenden Attribute für das Konstruktionspattern zum XDS Integration Profile gegeben. Diese umfassen:

- Für die PatternObjects:
  - *qualifiedName*
  - *description*
  - *applicationHint*
- Für die SubPatterns:
  - *description*
- Für das Pattern:
  - *description*
  - *keywords*

In Fällen, wo keine Angaben zu einem Attribut gemacht werden, ist dieses als unbestimmt zu betrachten und erhält damit keinen Wert im Konstruktionspattern.

Der erste Abschnitt bezieht sich auf das Pattern allgemein, der zweite speziell auf die Actors und der dritte auf die Transactions.

Vorbemerkung:

In Analogie zum Konstruktionspattern selbst sind auch hier die Bezeichnungen und Beschreibungen denen im XDS Integration Profile entlehnt, um möglichst genau den Wortlaut der IHE wiederzugeben.

9.1.1 Global

9.1.1.1 XDS Integration Profile (Pattern)

description:

Aims of this pattern:

1. Visualizing the IHE XDS Integration Profile
2. Giving a basic overview for information managers in healthcare to plan a realization of cross-enterprise document sharing.
3. Connecting these basic concepts with the technical parts of the ITI-Technical Framework to guide towards an implementation of this Integration Profile.

Objective:

The Cross-Enterprise Document Sharing IHE Integration Profile facilitates the registration, distribution and access across health enterprises of patient electronic health records. Cross-Enterprise Document Sharing (XDS) is focused on providing a standards-based specification for managing the sharing of documents between any healthcare enterprise, ranging from a private physician office to a clinic to an acute care in-patient facility.

Background:

The IHE Technical Framework identifies functional components of a distributed healthcare environment (referred to as IHE actors), solely from the point of view of their interactions in the healthcare enterprise.

Each integration profile is a representation of a real-world capability that is supported by a set of actors that interact through transactions. Actors are information systems or components of information systems that produce, manage, or act on categories of information required by operational activities in the enterprise. Transactions are interactions between actors that communicate the required information through standards-based messages.

The IHE actors and transactions described in the IHE Technical Framework are abstractions of the real-world healthcare information system environment. While some of the transactions are traditionally performed by specific product categories (e.g. HIS, Clinical Data Repository, Radiology Information Systems, Clinical Information Systems or Cardiology Information Systems), the IHE Technical Framework intentionally avoids associating functions or actors with such product categories. For each actor, the IHE Technical Framework defines only those functions associated with integrating information systems. The IHE definition of an actor should therefore not be taken as the complete definition of any product that might implement it, nor should the framework itself be taken to comprehensively describe the architecture of a healthcare information system.

Remarks:

This pattern is originated from IHE's XDS Integration Profile. The majority of the textual parts – like names, description etc. – are cited from the ITI - Technical Framework (ITI-TF), allowing to reproduce and convey IHE’s intentions to users most directly.

keywords:

IHE, XDS, Cross-enterprise-document-sharing, information-system

9.1.1.2 XDS Affinity Domain (SubPattern)

description:

The XDS IHE Integration Profile assumes that enterprises belong to one or more XDS Affinity Domains. An XDS Affinity Domain is a group of healthcare enterprises that have agreed to work together using a common set of policies and share a common infrastructure.

Examples of XDS Affinity Domains include:

- Community of Care supported by a regional health information organization in order to serve all patients in a given region.
- Nationwide EHR
- Specialized or Disease-oriented Care
    - Cardiology Specialists and an Acute Cardiology Center
    - Oncology network
    - Diabetes network
- Federation of enterprises
    - A regional federation made up of several local hospitals and healthcare providers
- Government sponsored facilities (e.g., VA or Military)
- Insurance Provider Supported Communities

Cross-Enterprise Document Sharing enables a number of healthcare delivery organizations belonging to an XDS Affinity Domain (e.g. a community of care) to cooperate in the care of a patient by sharing clinical records in the form of documents as they proceed with their patients' care delivery activities. Federated document repositories and a document registry create a longitudinal record of information about a patient within a given XDS Affinity Domain. This profile is based upon ebXML Registry standards, SOAP, HTTP and SMTP. It describes the configuration of an ebXML Registry in sufficient detail to support Cross Enterprise Document Sharing.

The XDS Integration Profile is not intended to address all cross-enterprise EHR communication needs. Some scenarios may require the use of other IHE Integration profiles, such as Patient Identifier Cross-Referencing, Audit Trail and Node Authentication, Cross-Enterprise User Authentication, and Retrieve Information for Display. Other scenarios may be only partially supported, while still others may require future IHE Integration profiles, which will be defined by IHE as soon as the necessary base standards are available. Specifically:

1. The management of dynamic information such as allergy lists, medication lists, problem lists, etc is not addressed by XDS. However, the Retrieve Information for Display Integration Profile does provide some transactions (e.g., LIST-ALLERGIES, LIST-MEDS) that may be used to provide an elementary support of such capabilities. A complementary approach to managing updates and structured application access to such dynamic clinical information may be expected as a separate Integration Profile in the future.
2. The placing and tracking of orders (e.g. drug prescriptions, radiology orders, etc.) is not supported by XDS. This does not preclude the use of XDS to store and register orders and corresponding results when such artifacts need to be recorded in the patient's health record. However, XDS provides no facilities for tracking progress of an order through its workflow, and therefore is not intended for order management. A complementary approach to cross-enterprise order workflow (ePrescription, eReferral) may be expected as separate Integration Profiles in the future.
3. The operation of any XDS Affinity Domain will require that a proper security model be put in place. It is expected that a range of security models should be possible. Although the XDS Integration Profile is not intended to include nor require any specific security model, it is expected that XDS implementers will group XDS Actors with actors from the IHE Audit Trail and Node Authentication and will need an Access Control capability that operates in such a cross-enterprise environment. Specific IHE Integration Profiles complementary to XDS are available (e.g. Cross-Enterprise User Authentication, Document Digital Signature, etc).
4. The establishment of independent but consistently XDS Affinity Domains will call for their

federation, as patients expect their records to follow them as they move from region to region, or country to country. IHE foresees a need for transferring information from one XDS Affinity Domain to another, or to allow access from one XDS Affinity Domain to documents managed in other XDS Affinity Domains. XDS has been designed with this extension in mind. An XDS Domains Federation Integration Profile that complements XDS may be anticipated in the future.

5. XDS does not address transactions for the management or configuration of an XDS Affinity Domain. For example, the configuration of network addresses or the definition of what type of clinical information is to be shared is specifically left up to the policies established by the XDS Affinity Domain.

9.1.1.3 XDS Affinity Domain

qualifiedName:

IHE:ITI:XDS-b:2007:XDSAffinityDomain

applicationHint:

The organizational units representing the enterprises inside 3LGM²-models should be set as children of the owning XDS Affinity Domain via the is_part_of-relation.

description:

The IHE actors and transactions described in the IHE Technical Framework are abstractions of the real-world healthcare information system environment. The reason for defining actors and transactions is to provide a basis for defining the interactions among functional components of the healthcare information system environment.

An XDS Affinity Domain is an administrative structure made of a well-defined set of Document Source Actors, set of Document Repositories, set of Document Consumers organized around a single Document Registry Actor that have agreed to share clinical documents.

DataModel

The Data Model of the XDS Integration Profile distinguishes between two main types of data:

1. Patient Demographic Data: Since the central focus of the XDS Integration Profile is “sharing documents”, it is critical that each document be reliably associated with the corresponding patient (Patient Id). The XDS Document Registry is not intended to be an authority for patient identification and demographics information. This Integration Profile uses a Patient Identity Source Actor as the authoritative source of Patient Identifiers (master patient ID) for the XDS Affinity Domain. The Patient Identifier Domain managed by the Patient Identity Source Actor in the XDS Affinity Domain is the source of patient identifiers (and merge operations) used by the XDS Document Registry to link Documents to a specific Patient. This Patient Identifier Domain is called the XDS Affinity Domain Patient Identification Domain (XAD-Pid Domain).
2. Document Data: A typical patient goes through a sequence of encounters in different care settings. In each care setting, the resulting patient information is created and managed by multiple care delivery information systems (EHR-CRs). Through a sequence of care delivery activities, a number of clinical documents are created. The XDS Documents shared by the EHR-CR and tracked by the XDS Registry form a Longitudinal Record for the patients that received care among the EHR-CRs of the XDS Affinity Domain. This shared clinical record is called an EHR-LR in this Integration Profile. The EHR-LR provides the means to share the relevant subset of these documents, as they are contributed by the various EHR-CRs that are

part of the same XDS Affinity Domain. For sharing those documents the XDS Profile proposes the Registry Data Model which is comprised of the following components:

a. XDS DOCUMENT: An XDS Document is the smallest unit of information that may be provided to a Document Repository Actor and be registered as an entry in the Document Registry Actor. An XDS Document is a composition of clinical information that contains observations and services for the purpose of exchange with the following characteristics: Persistence, Stewardship, Potential for Authentication, and Wholeness. These characteristics are defined in the HL7 Clinical Document Architecture Release 1 specification. An XDS Document may be human readable (with the appropriate application). In any case, it should comply with a published standard defining its structure, content and encoding. IHE intends to define content-oriented Integration Profiles relying on such content standards to be used in conjunction with XDS. The XDS Integration Profile manages XDS Documents as a single unit of information; it does not provide mechanisms to access portions of an XDS Document. Only the Document Sources or Document Consumers have access to the internal information of the XDS Document. When submitted for sharing, an XDS Document is provided to the Document Repository Actor as an octet stream. When retrieved through the Retrieve Document transaction, it shall be unchanged from the octet stream that was submitted. The Document Source Actor is responsible to produce the metadata that will be submitted to the Document Registry Actor to form the XDS Document Entry that will be used for query purposes by XDS Consumer Actors. The Document Source maintains responsibilities over the XDS Documents it has registered. It shall replace XDS Documents that may have been submitted in error. See ITI TF-1: Appendix K for a more detailed discussion of the concept of XDS Document. XDS Documents are required to be globally uniquely identified. See ITI TF-2x: Appendix B for a definition of globally unique identifiers.
b. XDS DOCUMENT ENTRY: The XDS Integration Profile provides a means to place documents in a repository chosen by the Document Source, and also to place information about this document (or metadata) in an entry of the Document Registry that manages the XDS Affinity Domain. This entry is called: “XDS Document Entry”. The term metadata reflects that this information is “about” the documents. The purpose of well-specified document metadata is to enable a uniform mechanism for Document Consumers to locate clinical documents of interest much in the way a card catalog in a library helps readers find the book they want. The XDSDocumentEntry comprises attributes which are representing the metadata. The Registry Actor may be queried against those attributes, which are thereby used to filter the XDSDocumentEntries.
c. XDS FOLDER: The purpose of an XDS Folder is to provide a collaborative mechanism for several XDS Document Sources to group XDS Documents for a variety of reasons (e.g. a period of care, a problem, immunizations, etc.) and to offer the Document Consumers a means to find all Document Entries placed in the same Folder. The following principles apply to an XDS Folder:
    i. A Folder groups a set of XDS Documents related to the care of a single patient,
    ii. One or more Document Source Actors may submit documents in a given Folder,
    iii. A Folder may be created by a Document Source and/or predefined in an XDS Affinity Domain,
    iv. The content of a Folder is qualified by a list of codes/meaning,
    v. Document Source Actors may find existing Folders by querying the Document Registry or by means outside the scope of XDS (e.g. Cross-enterprise workflow, such ePrescription, eReferral, etc),
    vi. Once created a Folder is permanently known by the Document Registry,
    vii. Placing previously existing Documents in Folders is not recorded as part of the Submission Set,
    viii. Folders in XDS may not be nested,

ix. The same documents can appear in more than one Folder, and
x. Folders have a globally unique identifier.

d. XDS SUBMISSION SET: An XDS Submission Set is related to care event(s) of a single patient provided by the care delivery organization EHR-CR performing the submission request. It creates a permanent record of new XDS Documents as well as pre-existing (i.e. already registered) XDS Documents that have a relationship with the same care event(s). It also includes the record of new XDS Folders creation. An XDS Submission Set shall be created for each submission request. It is related to a single Document Source Actor and is conveyed by a single Provide & Register Document Set Transaction or a Register Document Set Transaction. The Document Registry may be queried to find all documents registered in the same XDS Submission Set. The same XDS Document, initially registered as part of a Submission Set, may also be referenced by later XDS Submission Set. This allows older documents relevant to the present care of a patient to be associated with more recent Submission Sets. XDS provides complete flexibility to EHR-CRs to relate Documents and Submission Sets to an encounter, a visit, an episode of care, or various workflow processes within EHR-CRs.

e. XDS SUBMISSION REQUEST: An XDS Submission Request is a means to share XDS Documents. There are two types of an XDS Submission Request:
   i. PROVIDE AND REGISTER DOCUMENT SET-b REQUEST (conveyed by a Document Source Actor in a Provide and Register Document Set Transaction to the Document Repository Actor)
   ii. REGISTER DOCUMENT SET-b REQUEST (conveyed by a Document Repository Actor in a Register Document Set Transaction to the Document Registry Actor)

Document data & metatdata

XDS Submission Request: A PROVIDE AND REGISTER DOCUMENT SET-b REQUEST carries zero or more XDS DOCUMENTS and metadata encapsulated in a REGISTER DOCUMENT SET-b REQUEST. The REGISTER DOCUMENT SET-b REQUEST itself is then comprised of a single XDS SUBMISSION SET as well as XDS DOCUMENT ENTRIES and XDS FOLDERS.

(Note: Submission Requests without new XDS Documents are possible. In this case the metadata refers to previously registered documents (e.g. for the reason of placing them into new folders).

XDS SUBMISSION SETS, XDS DOCUMENT ENTRIES and XDS FOLDERS are sets of attributes that together represent the metadata for XDS DOCUMENTS. These Attributes required in almost all sections of the Integration Profile are listed in ITI-TF-3. Moreover Vol3 of the ITI-TF explains how to link Submission Sets, Document Entries and Folders. In addition, it is shown how these concepts are mapped to the applied standards for the communication of the documents.

EHR-CR & XDS

An EHR-CR or Care-delivery Record abstracts the information system or systems of a care delivery organization, which may support a broad variety of healthcare facilities: private practice, nursing home, ambulatory clinic, acute care in-patient facility, etc.

Typically a patient goes through a sequence of encounters in different care settings. It is out of the scope of this IHE Integration Profile to define or restrict the type of care provided, nor the internal workflow of a care delivery organization. The EHR-CR system participates only to the cross-enterprise clinical document sharing as Document Source and Document Consumer Actors according to the following principles:

1. EHR-CR as Document Source contributes documents in any one of the document formats that are supported by the XDS Affinity Domain (e.g. CDA Release 1, CDA Release 2 with

specific templates, DICOM Composite SOP Classes, ASTM-CCR, CEN ENV 13606 etc).
2. This Profile does not require that the EHR-CR as Document Sources and Consumers store and manage their internal information in the form of documents as they are shared throughout the XDS Affinity Domain.
3. By grouping a Document Source with a Document Repository, an EHR-CR may leverage existing storage provide a unified access mechanism without needing to duplicate storage.
4. EHR-CRs as Document Sources and Consumers are responsible to map their local codes into the XDS Affinity Domain codes if necessary.

9.1.1.4 SOAP 1.2

qualifiedName:

W3C:SOAP:1.2

description:

For an example of a SOAP-based message see:

ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/XDS.b/ProvideAndRegisterDocumentSet-bRequest_SOAP.xml.

9.1.1.5 HL7 2.3.1

qualifiedName:

OASIS:HL7:2.3.1

9.1.2 Actors

9.1.2.1 Actors (SubPattern)

description:

The reason for defining actors and transactions is to provide a basis for defining the interactions among functional components of the healthcare information system environment. In situations where a single physical product implements multiple functions, only the interfaces between the product and external functions in the environment are considered to be significant by the IHE initiative. Therefore, the IHE initiative takes no position as to the relative merits of an integrated environment based on a single, all-encompassing information system versus one based on multiple systems that together achieve the same end. IHE demonstrations emphasize the integration of multiple vendors' systems based on the IHE Technical Framework.

At its current level of development, it defines a coordinated set of transactions based on ASTM, DICOM, HL7, IETF, ISO, OASIS and W3C standards.

9.1.2.2 Document Source Actor

qualifiedName:

IHE:ITI:XDS-b:2007:DocumentSourceActor

description:

Relations to the Provide and Register Document Set Transaction

An implementation of the Document Source actor shall be capable of the following operations:

- Submit one or more documents. Whether a submission contains a single or multiple documents depends on workflows, policies, and other external factors which are outside of the scope of this profile. The Document Source shall supply all necessary document metadata attributes with the exception of the four below: (Those are assigned by the Document Repository)
  - XDSDocumentEntry.repositoryUniqueId
  - XDSDocumentEntry.hash
  - XDSDocumentEntry.size
  - XDSDocumentEntry.URI
- Submit a document as a replacement for another document already in the registry/repository

An implementation of the Document Source actor may support one or more of the following XDS.b

options:

- Document Replace Option: In this option the Document Source offers the ability to submit a document as a replacement for another document already in the registry/repository.
- Document Addendum Option In this option the Document Source shall offer the ability to submit a document as an addendum to another document already in the registry/repository.
- Document Transformation Option In this option the Document Source shall offer the ability to submit a document as a transformation of another document already in the registry/repository.
- Folder Management Option. In this option the Document Source offers the ability to perform the following operation:
  - Create a folder
  - Add one or more documents to a folder

Note: In order to support document replacement/addendum/transformation grouping with the Document Consumer may be necessary in order to Query the registry (e.g. for UUIDs of existing document entries)

Note: In order to support document addition to an existing folder, grouping with the Document Consumer may be necessary in order to Query the registry (e.g. for UUIDs of existing folder).

These operations are discussed in ITI TF-3: 4.1.3.4 Other Properties of Submission Requests.

9.1.2.3 Document Repository Actor

qualifiedName:

IHE:ITI:XDS-b:2007:DocumentRepositoryActor

description:

Relations to the Provide and Register Document Set Transaction

The Document Repository shall, upon receipt of a Provide and Register Document Set-b [ITI-41] transaction send a corresponding Register Document Set-b [ITI-42] transaction to the Document Registry actor. A Document Repository shall be capable of accepting submissions containing multiple documents.

Note: The Document Source may submit single documents or multiple documents depending on its needs.

A Document Repository shall validate the following metadata elements received as part of a Provide and Register transaction:

- XDSDocumentEntry.uniqueId – a submission shall be rejected if not unique within the repository and the hashes of the two documents do not match. If the hashes of the documents match, the Document Repository shall accept the duplicate document.
- XDSSubmissionSet.sourceId – a Document Repository may choose to accept submissions only from certain sources and use this field to perform the filtering.

The Document Repository actor shall create and insert the following attributes for each document received from the Provide and Register Document Set-b [ITI-41] transaction into the resulting Register Document Set-b [ITI-42] transaction metadata.

- XDSDocumentEntry.repositoryUniqueId (identifies the Repostory which holds the document)
- XDSDocumentEntry.hash (used for validation of the submitted document)
- XDSDocumentEntry.size (the size of the XDSDocument)
- XDSDocumentEntry.URI (identifies an XDSDocument inside a Repository for direct access)

A Register Document Set-b transaction with this modified metadata shall be issued to the Document Registry.

Note: the document URI attribute is optional for XDS.b implementations. If the XDSDocumentEntry.URI attribute is present, then the Document Repository shall support the Retrieve Document transaction (ITI TF-2a:3.17). If the attributes “hash” and “size” are received in a Provide and Register Document Set-b [ITI-41] transaction, they shall be ignored.

The Document Repository shall ensure that when any Retrieve Document Set transaction is received requesting a specific document(s), it shall be provided to the Document Consumer unchanged from the octet stream that was submitted (full fidelity repository) and shall match the size and hash

attributes of the XDSDocumentEntry object.

Relations to the Retrieve Document Set transaction

When receiving a Retrieve Document Set Request, a Document Repository or an Initiating Gateway shall generate a Retrieve Document Set Response containing the requested documents or error codes if the documents could not be retrieved. The conditions of failure and possible error messages are given in the ebRS standard and detailed in ITI TF-3: 4.1.13 Error Reporting.

- See ftp://ftp.ihe.net/TF_Implementation_Material/ITI/wsdl/XDS.b_DocumentRepository.wsdl for the scheme of the Repository

9.1.2.4 Document Registry Actor

qualifiedName:

IHE:ITI:XDS-b:2007:DocumentRegistryActor

description:

Relations to the Register Document Set-b Transactions

Upon receipt of a Register Document Set-b Request message, the Document Registry with the aid of the Registry Adaptor shall do the following:

- Accept all valid SubmitObjectsRequests.
- Perform metadata validations
- Update the registry with the contained metadata
- Return a RegistryResponse message given the status of the operation.

Relations to the Patient Identity Feed Transactions

1.) PID Segement

The Document Registry shall be capable of accepting attributes in the PID segment of an ADT Message as specified here:

SEQ	LEN	DT	OPT	TBL#	ITEM#	ELEMENT NAME
3	250	CX	R		00106	Patient Identifier List

(Adapted from the HL7 standard, Version 2.3.1 See ITI TF-2x: C.1 for further description)

Note: This table reflects only the attributes required to be handled by the Document Registry (receiver). Other attributes of the PID Segment may be ignored.

If subcomponents 2 and 3 (the universal ID and the universal ID Type of Assigning Authority) of the

Patient Identification Domain of the XDS Affinity Domain in PID-3.4 are not filled in the message (as described in ITI TF-2a: 3.8.4.1.2.3) the Document Registry shall fill subcomponents 2 and 3 of the Patient Identification Domain of the XDS Affinity Domain prior to storing the patient identity in the registry. The assigning authority information filled by the Document Registry is based on its configuration of the Patient Identification Domain of the XDS Affinity Domain (See ITI TF-2a: 3.8.4.1.4.1 for a list of required Document Registry configuration parameters).

2.) MRG Segment

The Document Registry shall be capable of accepting attributes in the MRG segment as specified in the following table. Other attributes may exist, but the Document Registry shall ignore them.

SEQ	LEN	DT	OPT	TBL#	ITEM#	ELEMENT NAME
1	250	CX	R		00211	Prior Patient Identifier List
2	250	CX	O		00212	Prior Alternate Patient ID
3	250	CX	O		00213	Prior Patient Account Number
4	250	CX	R2		00214	Prior Patient ID
5	250	CX	O		01279	Prior Visit Number
6	250	CX	O		01280	Prior Alternate Visit ID
7	250	XPN	R2		01281	Prior Patient Name

Adapted from the HL7 Standard, Version 2.3.1 In addition, the Document Registry shall perform the Expected Actions as specified in ITI TF-2a: 3.8.4.1.4. When the Document Registry receives the ADT^A40 message type of the Patient Identity Feed transaction, it shall merge the patient identity specified in MRG-1 (secondary patient identity) into the patient identity specified in PID-3 (primary patient identity) in its registry. After the merge, all Document Submission Sets (including all Documents beneath them) under the secondary patient identity before the merge shall point to the primary patient identity. The secondary patient identity shall no longer be referenced in the future services provided by the Document Registry.

Relations to Registry Stored Query and Retrieve Document Set Transactions

Document Registry actors implementing this transaction shall support all queries and all parameters defined for each query. (see Section L. of this model)

The Document Registry actor shall

- Accept a parameterized query in an AdhocQueryRequest message
- Verify the required parameters are included in the request.
- Errors shall be returned for the following conditions:
    - Unknown query ID (error code XDSUnknownStoredQuery)
    - Required parameter missing (error code XDSStoredQueryParamNumber)

  See ITI TF-3: 4.1.13 Error Reporting for additional error codes and general information on formating error responses.
- Retrieve the internal implementation template of the query based on the Query ID supplied in the query request.
    - The Document Registry shall accept the homeCommunityId value if it is specified in

a Registry Stored Query request.
    - If a patient identifier specified as a parameter to the query is unknown to the Document Registry it shall return a sucessful response with no elements.
- Return XML formatted metadata in an AdhocQueryResponse message.
- The Document Registry may specify the homeCommunityID attribute on any appropriate elements The homeCommunityId attribute corresponds to the ´home´ attribute specified in the ebRIM standard.
    - If returntype=´LeafClass´ the ExtrinsicObject and RegistryPackage elements shall contain the home attribute.
    - If returnType=´ObjectRef´ the ObjectRef element shall contain the home attribute

This transaction may return both errors and results in an AdhocQueryResponse message. To do this, the returned AdhocQueryResponse message would contain both a RegistryObjectList element and a RegistryErrorList element. See ITI TF-3: 4.1.13 for additional details on formatting of error responses.

It is the responsibility of the Document Registry actor to translate between version 2.1 and version 3.0 formats when returning v2.1 objects in v3.0 query responses.

Identifier of the Patient Identification Domain

The following items are expected to be parameters that are configurable on the Document Registry Actor:

- Identifier of the Patient Identification Domain of the XDS Affinity Domain. This identifier shall be specified with 3 components of the HL7 assigning authority (data type HD): namespaceID, universal ID and universal ID type. The universal ID shall be an ISO OID (Object Identifier), and therefore the universal ID Type must be "ISO".

→ see ftp://ftp.ihe.net/TF_Implementation_Material/ITI/wsdl/XDS.b_DocumentRegistry.wsdl for the scheme of the registry.

INITIALIZATION

A standard ebXML Registry must be initialized with key Classification Schemes and object types to support XDS. An ebXML Registry SubmitObjectsRequest is available to perform this initialization. It includes:

- Classification Schemes that anchor the definition of ExternalIdentifiers
- Additions to the ObjectType ClassificationScheme that introduces a general XDS ClassificationNode that anchors these additions. The usable new ClassificationNodes are: XDSDocumentEntry, XDSDocumentEntryStub, XDSFolder, and XDSSubmissionSet. XDSDocumentEntry and XDSDocumentEntryStub are used as new objectTypes for use in an ExtrinsicObject to create XDS specific object types. XDSFolder and XDSSubmissionSet are used to classify RegistryPackage objects to label them as XDS Folders or XDS SubmissionSets.
- External Classification Schemes to support attribute coding.

This initialization includes the assignment of UUIDs to these definitions. These pre-assigned UUIDs shall be used when implementing XDS. The SubmitObjectsRequest initializing the Registry is deposited in XDSb.Registry.Initialization.zip.

9.1.2.5 Patient Identity Source Actor

qualifiedName:

IHE:ITI:XDS-b:2007:PatientIdentitySourceActor

description:

The Patient Identifier Domain managed by the Patient Identity Source Actor in the XDS Affinity Domain is the source of patient identifiers (and merge operations) used by the XDS Document Registry to link Documents to a specific Patient. This Patient Identifier Domain is called the XDS Affinity Domain Patient Identification Domain (XAD-Pid Domain).

For a given Patient Identifier Domain there shall be one and only one Patient Identity Source Actor, but a given Patient Identity Source Actor may serve more than one Patient Identifier Domain.

9.1.2.6 Document Consumer Actor

qualifiedName:

IHE:ITI:XDS-b:2007:DocumentConsumerActor

description:

The Document Consumer Actor queries a Document Registry Actor for documents meeting certain criteria, and retrieves selected documents from one or more Document Repository actors.

9.1.2.7 Federated Document Source/Repository Actor

qualifiedName:

IHE:ITI:XDS-b:2007:FederatedDocumentSource/RepositoryActor

description:

The Integrated Document Source/Repository does not initiate nor accept the Provide and Register Document Set transaction. This actor may replace the Document Repository actor from the perspective of the Register Document Set or Retrieve Document transactions.

9.1.3 Transactions

Vorbemerkung:

Die Beschreibungen zu den Transactions selbst (hier gekennzeichnet durch den Suffix „[ITI-#]“) sind in den SubPatterns für beide PatternObjects, die die Transaction repräsentieren (dort erkennbar am Suffix „(1)“ bzw. „(2)“), anzuwenden. Damit besitzen die jeweils davon abgeleiteten Kommunikationsbeziehungspaare denselben Beschreibungstext.

9.1.3.1 Provide and Register Document Set-b Transaction (SubPattern)

description:

Conduction

A Document Source Actor initiates the PROVIDE AND REGISTER DOCUMENT SET TRANSACTION. For each document in the submitted set, the Document Source Actor provides both the documents as an opaque octet stream and the corresponding metadata to the Document Repository. The Document Repository is responsible to persistently store these documents, and to register them in the Document Registry using the Register Documents transaction by forwarding the document metadata received from the Document Source Actor.

To support composite documents, an XDS Document may be a multipart document. The Document Repository must handle multi-part data sets as an “opaque entity”. The Document Repository does not need to analyze or process its multi-part structure nor the content of any parts in the context of the XDS Integration Profile.

Registry Adaptor Enforcement of Attributes

ebRIM version 2.1 datatype Slot/ValueList/Value is limited to 128 characters by that standard. Many HL7 datatypes, which the attribute tables show as being encoded as a Slot, can be much larger. The Document Source shall encoded these Slots so they fit into the 128 character space allocated to them. This may require some information to be excluded. This profile gives no guidance as to how information is to be excluded to make this coding limit.

The Registry Adaptor shall reject any submission which includes attribute values whose size exceeds the specification in the standard.

The enforcement for each metadata attribute is included in this model via a Userfield. The respective values can be found at the attributes of XDSDocumentEntry, XDSFolder and XDSSubmissionSet.

Atomicity Requirements

XDS Submission requests shall be atomic operations. The result of a Submission Request is to update either:

- a Registry or
- a Registry and a Repository.

All changes requested are successfully applied or no changes are made. More specifically:

1. Atomicity shall be managed by an XDS registry adaptor. (see ITI TF-3: 4.1.11 for details on registry adaptor.addressing the fact that the ebXML Registry specification does not guarantee that a SubmitObjectsRequest is atomic). XDS specifies the mechanism through which atomicity is to be implemented and where it is needed.

2. All objects shall have their Status attribute set to Submitted when the objects are first created in the

ebXML registry. An ebXML ApproveObjectsRequest, shall be issued within the XDS Registry Adaptor to change the Status attribute to Approved. This completes the transaction.

3. The following types of objects shall be have their status set to Approved to be considered publicly available:

- XDSSubmissionSet (ebXML RegistryPackage)
- XDSFolder (ebXML RegistryPackage)
- XDSDocumentEntry (subclass of ebXML ExtrinsicObject)

If an error occurs storing documents in the repository then all documents stored as part of the PROVIDE AND REGISTER DOCUMENT SET REQUEST shall be removed. If an error occurs storing metadata in the registry, then the following actions are performed:

- All metadata stored as part of the REGISTER DOCUMENT SET REQUEST shall be removed from the registry
- All documents stored as part of the PROVIDE AND REGISTER DOCUMENT SET REQUEST shall be removed. This only applies if the REGISTER DOCUMENT SET REQUEST is a result of a PROVIDE AND REGISTER DOCUMENT SET REQUEST.

Registry queries from the Registry Query transaction shall not find XDS Submission Sets, XDS Folders or XDSDocumentEntry objects until after the above atomic operation that creates them has completed successfully and the status attributes have been set to Approved.

9.1.3.2 Provide and Register Document Set-b Transaction [ITI-41]

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-41

description:

Use

A Document Source Actor initiates the Provide and Register Document Set Transaction. For each document in the submitted set, the Document Source Actor provides both the documents as an opaque octet stream and the corresponding metadata to the Document Repository. The Document Repository is responsible to persistently store these documents, and to register them in the Document Registry using the Register Documents transaction by forwarding the document metadata received from the Document Source Actor.

Message transfer

The Provide and Register Document Set-b transaction passes a Provide and Register Document Set Request (see ITI TF-3: 4.1.3.1) from a Document Source to a Document Registry.

A Provide and Register Document Set-b transaction shall carry:

- Metadata describing zero or more documents
- Within metadata, one XDSDocumentEntry object per document
- XDS Submission Set definition along with the linkage to new documents and references to existing documents
- Zero or more XDS Folder definitions along with linkage to new or existing documents
- Zero or more documents

The documents and metadata go to the Document Repository actor and then the metadata is forwarded on to the Document Registry actor. They move in this direction for several reasons:

- Allows best reuse of ebXML Registry specified metadata and web services protocols
- Document Source only needs to know the identity of the Document Repository. Document Repository knows the identity of the Document Registry. If Provide and Register Document Set-b transaction were sent to the Document Registry then routing decisions for documents would be more complex.
- Resulting protocols are simpler
- Simplifies the common case where the Document Source and the Document Repository are grouped.

Referenced Standards

This transaction aligns with the Registry Services standard (ebRS) for the format of the document metadata as defined in ITI TF-3: 4.1. The ebRS standard covers the interaction with a service that includes a registry with integrated repository. From the point of view of the Document Source, the separate nature of the Document Registry and Document Repository actors is not relevant.

Implementors of this transaction shall comply with all requirements described in: ITI TF-2x: Appendix V: Web Services for IHE Transactions.

- ebRIM - OASIS/ebXML Registry Information Model v3.0
- ebRS - OASIS/ebXML Registry Services Specifications v3.0
- Appendix V - ITI TF-2x:Appendix V Web Services for IHE Transactions. Contains references to all Web Services standards and requirements of use
- MTOM - SOAP Message Transmission Optimization Mechanism http://www.w3.org/TR/soap12-mtom/
- XOP - XML-binary Optimized Packaging http://www.w3.org/TR/2005/REC-xop10-20050125/

Response

The Document Repository sends a Provide and Register Document Set-b Response when the processing of a Provide and Register Document Set-b Request is complete.

The Provide and Register Document Set-b Response message shall carry the status of the requested operation and an error message if the requested operation failed. The conditions of failure and possible error messages are given in the ebRS standard and detailed in ITI TF-3: 4.1.13 Error Reporting.

The following events can trigger this message:

- Documents stored to repository successfully and metadata stored to registry successfully (The registry part is carried out as part of a Register Document Set-b transaction)
- Documents stored to repository successfully but an error occurred in storing the metadata to the registry
- Documents were not successfully stored to the repository

The Document Source now knows that the transaction succeeded/failed and can continue. The metadata added to the registry as a result of this transaction is now available for discovery via Registry Stored Query transactions. The document(s) added to the repository are now available for retrieval.

Further details are provided in ITI-TF-2b (3.41).

9.1.3.3 ITI-41 Document Source Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-41_DocumentSourceInterface

9.1.3.4 ITI-41 Document Repository Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-41_DocumentRepositoryInterface

9.1.3.5 Provide and Register Document Set-b Request

qualifiedName:

IHE:ITI:XDS-b:2007:ProvideAndRegisterDocumentSet-bRequest

applicationHint:

The request is initiated whenever a Document Source wants to submit documents to a Document Repository. A modeler should therefore add an event type to the request that indicates the creation of the data resp. the document intended for being provided inside the affinity domain. This is the reason for the absence of a special event type-PatternObject, since these are not restricted to XDS only. Rather, it is recommended to use such events that are already applied to the common internal communication of data between the different Application components for the sake of integrating the cross-enterprise document sharing into the information processes of the respective facility. The linking should be done via the contains-association.

Moreover Provide and Register Document Set-b Request Messages have to be connected to the Domain Layer of the 3LGM². According to IHE's specification message types derived from this PatternNode have to carry the documents as well as information about the patient, which requires an entity type representing the patient as well as entity types representing the content of the documents being linked with those message types via the is_represented_by-association. (Note: An XDS Document may be composite).

If applicable, at first such event types or entity types have to be created.

Each component interface derived from ITI-41 Document Source Interface that wants to send this particular combination of event type and message type should therefore be linked with the corresponding instance of the Provide and Register Document Set-b Request via a send-association.

9.1.3.6 Provide and Register Document Set-b Request Message

qualifiedName:

IHE:ITI:XDS-b:2007:ProvideAndRegisterDocumentSet-bRequestMessage

applicationHint:

Provide and Register Document Set-b Request Messages have to be connected to the Domain Layer of the 3LGM². According to IHE's specification message types derived from this PatternNode have to carry the documents as well as information about the patient, which requires an entity type representing the patient as well as entity types representing the content of the documents being linked

with those message types via the is_represented_by-association. (Note: An XDS Document may be composite).

description:

Specification

A Provide and Register Document Set-b Request is the collection of metadata and documents transferred between a Document Source and a Document Repository using a single ebXML SubmitObjectsRequest. The Provide and Register Document Set-b Request is used by the Document Source to submit XDSDocuments to the Document Repository and to forward the associated metadata to the Document Registry. This request contains:

- Metadata (within a Register Document Set-b Request):
  - XDS DOCUMENT ENTRIES: A new registry object type is declared as a subclass of ebXML ExtrinsicObject. Its name is XDSDocumentEntry. An object of this type in the XDS registry is used to represent a document in an XDS repository. An XDSDocumentEntry object in the registry contains a reference to a single document in a single repository. Note: A repository may hold documents that are not indexed in the registry. ITI TF-2x: Appendix H defines the metadata to initialize an ebXML registry to serve as an XDS Document Registry.
  - XDS FOLDERS: An XDS Folder is an ebXML RegistryPackage classified as XDSFolder. This folder is used to bundle XDSDocumentEntry objects. Folders shall not be nested inside other folders. The patientId attribute of the XDSDocumentEntry objects it contains shall match the patientId attribute on the folder itself. This shall be enforced by the Registry Actor. Note: The nesting of folders may be considered as a future extension to this transaction.
  - One XDS SUBMISSION SET: An XDS SubmissionSet is an ebXML RegistryPackage, classified as XDSSubmissionSet that is used to bundle XDSDocumentEntry, XDSFolder and Association objects for submission. Submission Sets, once shared, are immutable. A Submission Set has a set of attributes that are described in ITI TF-3: 4.1.8 Submission Set Metadata. Submission Sets exist for two reasons:
    - To support atomic submission to the registry
    - To make a permanent record in the registry of:
      - The existence and status of the submission
      - The XDS Folders and XDSDocumentEntry objects included in the submission
- Zero or more documents (A stream of bytes stored in a Document Repository Actor and pointed to by an XDS Document Entry)

Submissions that add metadata to the registry without adding documents to the repository are possible.

Usage

The Provide and Register Document Set-b message shall include the metadata attributes (as defined in ITI TF-3: 4.1.7 Document Definition Metadata) that will be forwarded by the Document Repository to the Document Registry using the Register Document Set-b transaction [ITI-42]. The Document Repository receives this message. Each document within the message shall be stored into the Document Repository as an octet stream with an associated MIME type. A detected failure shall result in an error message being returned to the Document Source thus terminating this transaction. The Document Source shall supply all necessary document metadata attributes with the exception of the four below – the Document Repository shall modify the received document metadata before initiating the Register Document Set-b transaction to the Document Registry by adding/replacing:

- The repositoryUniqueId for this Document Repository to allow for the Document Consumer

to correctly identify the proper Document Repository for each document (XDSDocumentEntry.repositoryUniqueId).
- A hash value (XDSDocumentEntry.hash)
- A size (XDSDocumentEntry.size).
- Optionally a URI identifier (XDSDocumentEntry.URI) that can be used by a Document Consumer to reference the document. This is only required if the repository is an XDS.a Document Repository therefore supporting the Retrieve Document [ITI-17] transaction.

A Register Document Set-b transaction with this modified metadata shall be issued to the Document

Registry. The Document Repository shall ensure that when any Retrieve Document Set transaction is received requesting a specific document(s), it shall be provided to the Document Consumer unchanged from the octet stream that was submitted (full fidelity repository) and shall match the size and hash attributes of the XDSDocumentEntry object.

An example for a Provide and Register Document Set-b Request is given in:

ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/XDS.b/ProvideAndRegisterDocumentSet-bRequest.xml.

A detailed overview and examples of the metadata attributes' representation inside such requests can be found in ITI-TF-3.

9.1.3.7 Provide and Register Document Set-b Response

qualifiedName:

IHE:ITI:XDS-b:2007:ProvideAndRegisterDocumentSet-bResponse

9.1.3.8 Provide and Register Document Set-b Response Event

qualifiedName:

IHE:ITI:XDS-b:2007:ProvideAndRegisterDocumentSet-bResponseEvent

applicationHint:

Event types derived from this PatternNode have no connected functions via the is_triggered_by_an_activity_of-association.

description:

The event indicates that the Document Repository wants to respond to the incoming transaction.

9.1.3.9 Registry Response Message

qualifiedName:

IHE:ITI:XDS-b:2007:RegistryResponseMessage

applicationHint:

Message types derived from this PatternNode have no connected entity types via the is_represented_by-association.

description:

The Registry Response Message shall carry the status of the requested operation and an error message if the requested operation failed.

The conditions of failure and possible error messages are given in the ebRS standard and detailed in ITI TF-3: 4.1.13 Error Reporting.

Table for the "status" -attribute of the RegistryResponseType (Acknowledgement) message type:

Value	Result
Success	All metadata and documents was successfully registered
Failure	Metadata and documents not stored

An example can be found in:

ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/XDS.b/RegisterDocumentSet-bResponse.xml.

9.1.3.10 Register Document Set-b Transaction (SubPattern)

description:

A Document Repository Actor initiates the REGISTER DOCUMENT SET TRANSACTION. This transaction allows a Document Repository Actor to register one or more documents with a Document Registry, by supplying metadata about each document to be registered. This document metadata will be used to create an XDS Document Entry in the registry. The Document Registry Actor ensures that document metadata is valid before allowing documents to be registered. If one or more documents fail the metadata validation, the Register Document Set transaction fails as a whole.

Note: The Register Document Set-b transaction may be performed by the Document Repository without a previous Provide and Register Document Set-b transaction.

Note: A Repository may contain documents not registered in the Registry.

Submission Sets and patients

A Submission Set is restricted in terms of mixing documents from different patients. All documents

included by value in a Submission Set shall have their patientId attribute set to the same value. This restriction does not apply to documents included by reference.

Attributes

repositoryUniqueId:

Allows the Document Consumer to correctly identify the proper Document Repository for each document

The combination of XDSDocumentEntry.uniqueId and XDSDocumentEntry.repositoryUniqueId attributes value shall later be accepted in a Retrieve Document Set transaction [ITI-43] for that document and the document shall be returned.

hash:

For detecting the improper resubmission of XDS Documents

size:

Size in bytes of the byte stream (the XDSDocument)

URI:

Can be used by a Document Consumer to reference the document directly (optionally).

The Document Repository actor shall create and insert the URI (XDSDocumentEntry.URI) attribute for each document received from the Provide and Register Document Set-b [ITI-41] transaction into the Register Document Set-b [ITI-42] transaction metadata if it will support retrieval of that document via the Retrieve Document [ITI-17] transaction. (this applies for an XDS.a Document Repository)

If this attribute is present in the Provide and Register Document Set-b [ITI-41] transaction it shall be replaced. If the Retrieve Document [ITI-17] transaction is not supported then this attribute shall not be present in Register Document Set-b [ITI-42] transaction metadata (removed by the Document Repository actor if necessary).

9.1.3.11 Register Document Set-b Transaction [ITI-42]

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-42

description:

Use

A Document Repository Actor initiates the Register Document Set transaction. This transaction allows a Document Repository Actor to register one or more documents with a Document Registry, by supplying metadata about each document to be registered. This document metadata will be used to create an XDS Document Entry in the registry. The Document Registry Actor ensures that document metadata is valid before allowing documents to be registered. If one or more documents fail the metadata validation, the Register Document Set transaction fails as a whole.

To support composite documents, an XDS Document may be a multipart document. The Document Repository must handle multi-part data sets as an “opaque entity”. The Document Repository does not need to analyze or process its multi-part structure nor the content of any parts in the context of the XDS Integration Profile.

If the registry rejects the metadata, then, the following shall occur:
- An error is returned
- The error status includes an error message
- The request is rolled back

Message transfer

The Register Document Set-b transaction passes a Submission Request from a Document Repository actor to a Document Registry actor.

A Register Document Set-b transaction shall carry:
- Metadata describing zero or more documents
- XDS Submission Set definition along with the linkage to new documents and references to existing
- documents
- An optional XDS Folder definitions along with linkage to new or existing documents

Referenced Standards

Implementors of this transaction shall comply with all requirements described in ITI TF-2x: Appendix V: Web Services for IHE Transactions.
- ebRIM - OASIS/ebXML Registry Information Model v3.0
- ebRS - OASIS/ebXML Registry Services Specifications v3.0
- HL7V2 - HL7 Version 2.5
- Appendix V - ITI TF-2x:Appendix V Web Services for IHE Transactions. Contains references to all Web Services standards and requirements of use.

Response

The Document Registry finishes processing a Register Document Set-b Request Message and shall respond with:
- Register Document Set-b Response

The Document Repository now knows that the transaction succeeded/failed and can continue. The metadata added to the registry as a result of this transaction is now available for discovery.

Further details are provided in ITI-TF-2b (3.42).

9.1.3.12 ITI-42 Document Repository Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-42_DocumentRepositoryInterface

9.1.3.13 Registry Adaptor

qualifiedName:

IHE:ITI:XDS-b:2007:RegistryAdaptor

description:

General Functionality

The XDS Registry Adaptor is a set of functionality that is not provided for in the ebXML registry standard, but is instead specified by XDS to support integration into the healthcare environment. This adaptor has the following responsibility:

- Validate patient ID: Patient IDs (XDSDocumentEntry.patientId attribute) shall be a known patient ID and registered against the Patient ID Domain of the XDS Affinity Domain managed by the patient Identity Source Actor. The adaptor shall verify that all documents in a folder are for the same patient. Specifically, verify that the patientId attribute of the folder matches the patientId attribute of each document in the folder. The patientId attribute of an XDSDocumentEntry object shall match the patientId attribute on any folder that holds it.

- Maintain Folder attribute 'lastUpdateTime': The XDS Folder attribute lastUpdateTime shall be updated by the adaptor every time a new document is added to an XDS Folder.

- Validate submitted metadata: The adaptor shall verify that submitted metadata meets XDS Registry metadata specification

- Verify coded values: The adaptor shall verify that coded fields (ebXML external classifications) contain valid XDS specified values or where the XDS Affinity Domain constrains code values, to verify them (See ITI TF-3: 4.1.10).

- Validate MIME types: The adaptor shall validate that the mimeType document attribute for all documents received is on the approved list for this XDS Affinity Domain.

- Validate coding: The adaptor shall enforce the number of classifications offered against a document. Code lists are allowed to be multiples. Codes are required to be singular.

- Accept submissions containing multiple documents: The adapter shall be capable of accepting submissions containing multiple documents.

- Ensure submissions are atomic: The adaptor shall make submission to registry an atomic operation – see ITI TF-3: 4.1.3.3 Atomicity Requirements for Submission Requests for atomicity requirements. If the registry submission is successful then the adaptor shall label all Document Entry, Folder, and Submission Set objects as Approved. The ebRIM specification provides the ApproveObjectsRequest for this purpose. If the registry submission fails then the adaptor shall remove from the registry all objects stored as part of this submission set. The ebRIM specification provides the RemoveObjectsRequest for this purpose.

- Support document replacement: When a Submission Request includes a 'RPLC' or 'XFRM_RPLC' association indicating that a document is being replaced, the following shall be true:
    - Document to be replaced must have status = Approved.

- The association's sourceObject attribute shall contain the id (UUID or symbolic id) of an ExtrinsicObject representing an XDSDocumentEntry included in the Submission Set.
- The association's targetObject attribute shall contain the UUID of an ExtrinsicObject (XDSDocumentEntry) already in the registry.
- Verify the ExtrinsicObject pointed to by the Association's targetObject attribute is present in the registry and has status of Approved. The error XDSReplaceFailed shall be thrown if this object is not contained in the registry or has status other than Approved. This ensures that only the most recent version of a document can be replaced.
- Submit the Submission Request to the registry.
- If the submission is successful, label the replacement document as Approved and the replaced document as Deprecated. The ebRIM requests ApproveObjectsRequest and DeprecateObjectsRequest are available to do this.
- If the Document being replaced is a member of one or more Folders, generate HasMember Associations connecting the replacement Document with each of the Folders holding the original Document. This makes the replacement Document a member of all Folders where the original Document is a member.

9.1.3.14 Register Document Set-b Request

qualifiedName:

IHE:ITI:XDS-b:2007:RegisterDocumentSet-bRequest

9.1.3.15 Register Document Set-b Request Event

qualifiedName:

IHE:ITI:XDS-b:2007:RegisterDocumentSet-bRequestEvent

applicationHint:

Event types derived from this PatternNode have no connected functions via the is_triggered_by_an_activity_of-association.

description:

The Register Document Set-b Request message is triggered when:

- A Document Repository wants to register metadata for a set of documents it holds. These documents may have been stored in the Document Repository by a Document Consumer (using the Provide and Register Document Set-b transaction [ITI-41]) or generated internally by an Integrated Document Source/Repository.

9.1.3.16 Register Document Set-b Request Message

qualifiedName:

IHE:ITI:XDS-b:2007:RegisterDocumentSet-bRequestMessage

applicationHint:

In accordance to the Provide and Register Document Set-b Request Message, the Register Document Set-b Request Message carrying the meta data for the documents has to be linked with the entity type representing the patient inside the information system. This has to be done with an is_represented_by-association, linking this entity type and the message type derived from the Register Document Set-b Request Message.

description:

Specification

The Register Document Set-b Request is the second part of a Provide and Register Document Set-b Request beside the XDSDocument.

A Register Document Set-b Request is the collection of metadata transferred between a Document Repository and a Document Registry in a single ebXML SubmitObjectsRequest. This request contains:

- A collection of metadata to be stored in the registry including:
  - Metadata for new documents (one XDSDocumentEntry for each new XDSDocument)
  - Folders to be created
  - Documents to be added to folders
- A single XDS Submission Set object, contained within the metadata, organizing the metadata

The Document Repository sends this metadata to the Document Registry.

An example for a Register Document Set-b Request is given in:

ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/XDS.b/RegisterDocumentSet-bRequest.xml.

A detailed overview and examples of the metadata attributes' representation inside such requests can be found in ITI-TF-3.

9.1.3.17 Register Document Set-b Response

qualifiedName:

IHE:ITI:XDS-b:2007:RegisterDocumentSet-bResponse

9.1.3.18 Register Document Set-b Response Event

qualifiedName:

IHE:ITI:XDS-b:2007:RegisterDocumentSet-bResponseEvent

description:

The event indicates that the Document Registry wants to respond to the incoming transaction.

applicationHint:

Event types derived from this PatternNode have no connected functions via the is_triggered_by_an_activity_of-association.

9.1.3.19 Registry Stored Query Transaction (SubPattern)

description:

The REGISTRY STORED QUERY TRANSACTION is issued by the Document Consumer Actor on behalf of a care provider (EHR-CR) to a Document Registry.

The Document Registry Actor searches the registry to locate documents that meet the provider's specified query criteria. It will return registry metadata containing a list of document entries found to meet the specified criteria including the locations and identifier of each corresponding document in one or more Document Repositories.

This transaction differs from Query Registry [ITI-16] by storing the query in the Document Registry actor and referencing the query in the transaction instead of passing SQL. With the Query Registry Transaction [ITI-16], SQL language queries are transmitted to the Registry actor and results are returned. In a Stored Query, the definition of the query is stored on the Registry actor.

Managing Large Query Responses

EbXML version 3.0 supports query results pagination (ebRS version 3.0 chapter 6.2). The interactions between the stored query capability and the query results pagination capability within the standard have never been reconciled and are not recommended for use together. It is recommended instead that query pagination be implemented within the Document Consumer actor. This can be accomplished by specifying returnType="ObjectRef" on all large queries (FIND QUERY returning object references). This returns a list of references (UUIDs) instead of full objects (large XML structures). This is practical for queries returning thousands of objects. To construct a page for display, a small number of objects can be retrieved through a second query (GET QUERY requesting full metadata). This is repeated for each page. As an example, the following sequence of queries could be used to list a large number of documents:

- FindDocuments query with returnType="ObjectRef" which returns a large collections of ObjectRefs (UUIDs)
- GetDocuments query with returnType="LeafClass" issued with a subset of the above returned UUIDs which returns the details to construct one page of listing

OR

- GetDocumentsAndAssocations query with returnType="LeafClass" issued with a subset of the above returned UUIDs which returns the details to construct one page of listing. By retrieving the Association objects, the existence of document replacement, transformation, and ammedment can be included into the display.

9.1.3.20 Registry Stored Query Transaction [ITI-18]

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-18

description:

Use

The Registry Stored Query will return registry metadata containing a list of document entries found to meet the specified criteria including the locations and identifier of each corresponding document in one or more Document Repositories.

The definition of the query is stored on the Registry actor. To invoke the query, an identifier associated with the query is transmitted along with parameters defined by the query. This has the following benefits:

1. Malicious SQL transactions cannot be introduced
2. Alternate database styles and schemas can be used to implement the Document Registry actor. This is because the style of SQL query statements is directly related to the table layout in a relational database.

This profile does not define how Stored Queries are loaded into or implemented in the Document Registry actor.

Message transfer

The Registry Stored Query transaction supports a variety of types of queries. Examples include the

following:

- Query by patient (Id) for a time interval, by document type(s), by practice setting(s), by author person
- Query by Document Source
- Query for XDS Folders updated during a time interval
- Query for all documents in a Folder or Submission Set
- Query by time of submission

Referenced Standards

Implementors of this transaction shall comply with all requirements described in ITI TF-2x: Appendix V: Web Services for IHE Transactions except for the level of SOAP supported. The SOAP level

support depends on the profile conformance of the implementing actor. The following table specifies

the level of conformance to SOAP:

Profile implemented by actor	SOAP level required of actor
XDS.a	SOAP 1.1
XDS.b	SOAP 1.2
XDS.a & XDS.b	SOAP 1.1 and SOAP 1.2 for all Registry Stored Query requests without reconfiguration or restart

ebRIM OASIS/ebXML Registry Information Model v3.0

ebRS OASIS/ebXML Registry Services Specifications v3.0

Response

All queries return:

- Metadata for one or more registry objects, or
- Object references for one or more registry objects (registry UUIDs).

9.1.3.21 ITI-18 Document Consumer Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-18_DocumentConsumerInterface

9.1.3.22 ITI-18 Document Registry Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-18_DocumentRegistryInterface

9.1.3.23 Registry Stored Query Request

qualifiedName:

IHE:ITI:XDS-b:2007:RegistryStoredQueryRequest

applicationHint:

Typically a need for previously created medical documents (e.g. physician's letters) arises with different events in a patient's course of treatment, like the arrival of the patient at the facility or the beginning of the care planning activities. These events causing a Registry Stored Query Transaction are usually not restricted to XDS only and thus are not represented inside this pattern. Preferably, users of the pattern should deploy such event types, that are already applied to represent those events inside the model resp. it is the modeler's job to create them at first. They have to be connected with the event-message types derived from the Registry Stored Query Request via the contains-assocaition.

9.1.3.24 Registry Stored Query Request Message

qualifiedName:

IHE:ITI:XDS-b:2007:RegistryStoredQueryRequestMessage

applicationHint:

Message types derived from this PatternNode should be connected with the entity type representing the patient inside the information system via the is_represented_by-association. This indicates that a Document Consumer's query to the Document Registry is related to one patient for the purpose of finding documents created for this patient.

description:

The semantics of Stored Query are defined in section 6.3. Stored Query Support of ebRS version 3.0. This transaction corresponds to section 6.3.2 Invoking a Stored Query and 6.3.3 Response to a Stored Query Invocation. This profile does not specify how the queries come to be stored in the Registry actor nor how they are to be translated for other database architectures. An example for a Registry Stored Query Request is given in: ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/XDS.b/RegistryStoredQueryRequest.xml. A detailed overview of query types and parameters can be found in ITI-TF-2a (3.18).

9.1.3.25 Registry Stored Query Response

qualifiedName:

IHE:ITI:XDS-b:2007:RegistryStoredQueryResponseMessage

9.1.3.26 Registry Stored Query Response Event

qualifiedName:

IHE:ITI:XDS-b:2007:RegistryStoredQueryResponseEvent

applicationHint:

Event types derived from this PatternNode have no connected functions via the is_triggered_by_an_activity_of-association.

description:

This event is triggered when the Document Registry has detected the document meta data corresponding to the Document Consumer's request and is now about to send it to the Document Consumer.

9.1.3.27 Registry Stored Query Response Message

qualifiedName:

IHE:ITI:XDS-b:2007:RegistryStoredQueryResponseMessage

applicationHint:

Message types derived from this PatternNode have no connected entity types via the is_represented_by-association.

description:

All queries return:

- Metadata for one or more registry objects, or
- Object references for one or more registry objects (registry UUIDs).

Table for the "status" -attribute of the AdhocQueryResponseType message type:

Value	Result
Success	Results shall be returned. Results may contain zero or more entries.
PartialSuccess	Results shall be returned. Results may contain zero or more entries.
Failure	Metadata and documents not stored Results not returned

The response includes a RegistryObjectList which has zero or more RegistryObjects that match the query specified in AdhocQueryRequest. An example is provided in:

ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/XDS.b/RegistryStoredQueryResponse.xml .

The result corresponds to the above RegistryStoredQueryRequest.

More details can be found in ITI-TF-2a (3.18).

9.1.3.28 Retrieve Document Set Transaction (SubPattern)

description:

The RETRIEVE DOCUMENT SET TRANSACTION (ITI TF-2: 3.43) is not supported in XDS.a.

A Document Consumer Actor initiates the Retrieve Document Set transaction. The Document Repository shall return the document set that was specified by the Document Consumer.

9.1.3.29 Retrieve Document Set Transaction [ITI-43]

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-43

description:

Use

Premise:

The Document Consumer has already obtained the XDSDocumentEntry uniqueId and the

Document Repository repositoryUniqueId from the Document Registry by means of the Registry Stored Query transaction.

The Document Repository shall return the document set that was specified by the Document Consumer.

(To support composite documents, an XDS Document may be a multipart document. In this case, the Document Consumer must take appropriate actions to make the multipart content accessible to the user.)

Message transfer

The Document Consumer requests the Document Repository to return the document set specified through the XDSDocumentEntry uniqueIds and Document Repository repositoryUniqueIds of the documents to be retrieved.

Referenced Standards

Implementors of this transaction shall comply with all requirements described in: ITI TF-2x: Appendix V: Web Services for IHE Transactions.

- ebRIM - OASIS/ebXML Registry Information Model v3.0
- ebRS - OASIS/ebXML Registry Services Specifications v3.0
- Appendix V - ITI TF-2x:Appendix V Web Services for IHE Transactions. Contains references to all Web Services standards and requirements of use
- MTOM - SOAP Message Transmission Optimization Mechanism http://www.w3.org/TR/soap12-mtom/
- XOP - XML-binary Optimized Packaging http://www.w3.org/TR/2005/REC-xop10-20050125/

9.1.3.30 ITI-43 Document Consumer Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-43_DocumentConsumerInterface

9.1.3.31 ITI-18 Document Registry Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-43_DocumentRepositoryInterface

9.1.3.32 Retrieve Document Set Request

qualifiedName:

IHE:ITI:XDS-b:2007:RetrieveDocumentSetRequest

applicationHint:

A Retrieve Document Set Request is initiated on behalf of a Document Consumer to retrieve documents selected from a previous Registry Stored Query Response. According to the Registry Stored Query Transaction this is done whenever events in a patient's course of treatment occur, hence the Retrieve Document Set Request Event is skipped here. Instead event-message types derived from this PatternNode should be provided with the same event-types used in the Registry Stored Query Request that has been issued to the Document Registry to get the document meta priorly to this transaction.

9.1.3.33 Retrieve Document Set Request Message

qualifiedName:

IHE:ITI:XDS-b:2007:RetrieveDocumentSetRequestMessage

description:

The Retrieve Document Set Request shall carry the following information:

- A required repositoryUniqueId that identifies the repository from which the document is to be retrieved. This value corresponds to XDSDocumentEntry.repositoryUniqueId.
- A required documentUniqueId that identifies the document within the repository. This value corresponds to the XDSDocumentEntry.uniqueId.
- If available, the homeCommunityId element that identifies the community holding the document.

The homeCommunityId element shall be specified if the XDSDocumentEntry containing the uniqueId of the document contains the homeCommunityId attribute.

The repositoryUniqueId associated to each document requested can be different therefore allowing a single request to identify multiple repositories.

An example can be found in:

ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/XDS.b/RetrieveDocumentSetRequest.xml.

More details are provided in ITI-TF-2 (3.43).

9.1.3.34 Retrieve Document Set Response

qualifiedName:

IHE:ITI:XDS-b:2007:RetrieveDocumentSetResponse

9.1.3.35 Retrieve Document Set Response Event

qualifiedName:

IHE:ITI:XDS-b:2007:RetrieveDocumentSetResponseEvent

applicationHint:

Event tyes derived from this PatternNode have no connected functions via the is_triggered_by_an_activity_of-association.

description:

The Retrieve Document Set Response is initiated when a Document Repository retrieves the documents requested by a Document Consumer.

9.1.3.36 Retrieve Document Set Response Message

qualifiedName:

IHE:ITI:XDS-b:2007:RetrieveDocumentSetResponseMessage

applicationHint:

A Retrieve Document Set Transaction is initiated on behalf of a Document Consumer to retrieve documents selected from a previous Registry Stored Query Response. The Document Repository will then return the matching documents inside a Retrieve Document Set Response Message, thus message types derived from this PatternNode have to be connected with the entity types representing the possible documents being retrieved via the is_represented_by-association.

description:

The Retrieve Document Set Response Message shall carry the following information for each of the returned documents:

- A homeCommunityId. This value shall be the same as the homeCommunityId value in the Retrieve Document Set Request Message. If the homeCommunityId value is not present in the Retrieve Document Set Request Message, this shall not be present.
- A required repositoryUniqueId that identifies the repository from which the document is to be retrieved. This value shall be the same as the value of the repositoryUniqueId in the original Retrieve Document Set Request Message. This value corresponds to XDSDocumentEntry.repositoryUniqueId.
- A required documentUniqueId that identifies the document within the repository. This value shall be the same as the documentUniqueId in the original Retrieve Document Set Request Message. This value corresponds to the XDSDocumentEntry.uniqueId.
- The retrieved document in base64binary encoded format
- The MIME type of the retrieved document
- Errors or warnings in case the document(s) could not be retrieved successfully

Table for the "status" -attribute of the RegistryResponseType (Acknowledgement) message type:

Value	Result
Success	All documents were successfully retrieved

PartialSuccess	Some documents were successfully retrieved
Failure	No documents were successfully retrieved

An example can be found in:

ftp://ftp.ihe.net/TF_Implementation_Material/ITI/examples/XDS.b/RetrieveDocumentSetResponse.xml.

More details are provided in ITI-TF-2 (3.43).

9.1.3.37 Patient Identity Feed Transaction – Part 1 (SubPattern)

description:

The Patient Identity Feed Transaction conveys the patient identifier and corroborating demographic data, captured when a patient's identity is established, modified or merged or in cases where the key corroborating demographic data has been modified. Its purpose in the XDS Integration Profile is to populate the registry with patient identifiers that have been registered for the XDS Affinity Domains.

Format mapping for combined Transactions

The Patient Identify Feed Transaction defined in ITI TF-2a:3.8 for HL7v2 and in ITI TF-2b: 3.44 for HL7v3 uses standard HL7 encoding of Patient Identifiers. This is standard encoding for HL7 applications; receiving applications are expected to extract the required data for their use.

When combined with the other XDS transactions, Document Registry actors and other actors that receive HL7 data with Patient Identifiers are required to map the data received in the HL7 message

to the format specified in those other XDS transactions. In those transactions, the Patient ID is treated using ebXML encoding rules and not HL7 encoding rules. Specifically, the Patient ID will be treated as a string, and extra components entered in that string shall cause those transactions to fail. XDS actors are required to use the specified encoding for Patient ID values in other transactions and not merely copy the value received in an HL7 transaction.

XDS.a & b interoperability

XDS.a implementations shall support Patient Identity Feed (ITI TF-2a: 3.8).

XDS.b implementations shall support either Patient Identity Feed (ITI TF-2a: 3.8) or Patient Identity Feed HL7v3 (ITI TF-2b: 3.44) or both. It is important to note that the version of HL7 implemented by XDS.b and Patient Identity Feed in a single domain or community need to match in order to allow interoperability. In the case of mixed scenarios, translation between Patient Identity Feed (ITI TF-2a:3.8) and Patient Identity Feed HL7v3 (ITI TF-2b: 3.44) will be required via a bridge or interface engine.

Patient Identifier Cross-referencing

A Patient Identifier Domain is called the XDS Affinity Domain Patient Identification Domain (XAD-Pid Domain).

As XDS Document Sources and Consumers may belong to different Patient Identification Domains, these systems need to cross-reference their own local Patient ID to the corresponding patient ID in the

XAD-Pid Domain of the Registry. Preferably, these systems may choose to use the IHE Patient Identifier Cross-referencing Integration Profile (See ITI TF-1: Appendix E.3) for this purpose.

Additional facts

This Integration Profile can be easily extended to support a scenario where no master patient ID is defined (i.e. no Patient Identity Source for the XDS Affinity Domain). Such option, would requiring the use of federated patient identities at the time of query of the XDS Document Registry, may be expected as a future addition to this Integration Profile.

9.1.3.38 Patient Identity Feed Transaction – Part 2 (SubPattern)

description:

Patient Identifier Communication Requirements

ITI Transaction 8 described in ITI TF-2a: 3.8 defines the format requirements for the patient identifier in PID-3. Specifically, the value for PID-3.4, Assigning Authority can be omitted, expressed using the first subcomponent (namespace ID) or the second and third subcomponents (universal ID and universal ID type). These rules shall apply in this profile:

1. If the Patient Identity Source does not include a value for PID-3.4, Assigning Authority, then
    a. PID-3, Patient Identifier List, is constrained to include one entry referring to one identifier.
    b. The Patient Identity Source and Document Registry shall agree that all messages from this source shall refer to a single assigning authority.
2. If PID-3.4 does contain a value for PID-3.4, Assigning Authority, then
    a. The Patient Identifier Source may send multiple patient identifiers with properly formatted components. The Document Registry shall be responsible for selecting the one identifier from the Patient Identifier List (not necessarily in the first position) that is too used to register the selected patient.
    b. As specified in ITI TF-2a: 3.8, the value for PID-3.4, Assigning Authority, can be expressed using the first subcomponent (namespace ID) or the second and third subcomponents (universal ID and universal ID type). Both methods shall be accepted by the Document Registry and shall be considered as equivalent.

9.1.3.39 Patient Identity Feed Transaction [ITI-8]

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-8

description:

Since the central focus of the XDS Integration Profile is “sharing documents”, it is critical that each document be reliably associated with the corresponding patient (Patient Id). This transaction communicates patient information, including corroborating demographic data, after a patient´s identity is established, modified or merged or after the key corroborating demographic data has been modified.

9.1.3.40 ITI-8 Patient Identity Source Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-8_PatientIdentitySourceInterface

9.1.3.41 ITI-8 Document Registry Interface

qualifiedName:

IHE:ITI:XDS-b:2007:ITI-8_DocumentRegistryInterface

9.1.3.42 ACK (Ereignis-Nachrichtentyp)

qualifiedName:

IHE:ITI:XDS-b:2007:ACK

9.1.3.43 A01_ADT

qualifiedName:

IHE:ITI:XDS-b:2007:A01_ADT

9.1.3.44 A04_ADT

qualifiedName:

IHE:ITI:XDS-b:2007:A04_ADT

9.1.3.45 A05_ADT

qualifiedName:

IHE:ITI:XDS-b:2007:A05_ADT

9.1.3.46 A08_ADT

qualifiedName:

IHE:ITI:XDS-b:2007:A08_ADT

9.1.3.47 A40_ADT

qualifiedName:

IHE:ITI:XDS-b:2007:A40_ADT

9.1.3.48 ACK (Ereignistyp)

qualifiedName:

IHE:ITI:XDS-b:2007:ACK

description:

General acknowledgement

9.1.3.49 A01

qualifiedName:

IHE:ITI:XDS-b:2007:A01

description:

Admission of an in-patient into a facility

9.1.3.50 A04

qualifiedName:

IHE:ITI:XDS-b:2007:A04

description:

Registration of an outpatient for a visit of the facility

9.1.3.51 A05

qualifiedName:

IHE:ITI:XDS-b:2007:A05

description:

Pre-admission of an in-patient (i.e., registration of patient information ahead of actual admission)

9.1.3.52 A08

qualifiedName:

IHE:ITI:XDS-b:2007:A08

description:

Update patient information

9.1.3.53 A40

qualifiedName:

IHE:ITI:XDS-b:2007:A40

description:

Patient identity merge

9.1.3.54 ACK (Nachrichtentyp)

qualifiedName:

IHE:ITI:XDS-b:2007:ACK(message)

description:

The simple general acknowledgment (ACK) can be used where the application does not define a special application level acknowledgment message or where there has been an error that precludes application processing. It is also used for accept level acknowlegements.

9.1.3.55 ADT

qualifiedName:

IHE:ITI:XDS-b:2007:ADT

applicationHint:

All message types derived from the ADT-PatternNode should be linked with entity type that represents the patient inside the information system via the is_represented_by-associtation. If no such entity type exists it has to be created at first.

description:

The ADT Message is used to communicate changes in the patient data. It is initiated by the Patient Identity Source and forwarded to the Document Registry whenever changes of a patient´s identity occur inside the Affinity Domain.

The structure of this message depends on the event type it is grouped with inside event-message types (e.g. A01_ADT).

ADT for A01, A04, A05 and A08

Semantics:

- The Patient Identity Feed transaction is conducted by the HL7 ADT message. The Patient Identity Source Actor shall generate the message whenever a patient is admitted, pre-admitted, or registered, or when some piece of patient demographic data changes. Pre-admission of inpatients shall use the A05 trigger event.

- Note: Conventions used in this section as well as additional qualifications to the level of specification and HL7 profiling are stated in ITI TF-2x: Appendix C and C.1.

- Required segments are defined below. Other segments are optional.

Structure:

- Segments:

ADT Patient Administration Message Chapter in HL7 2.3.1

MSH	Message Header	2
EVN	Event Type	3
PID	Patient Identification	3
PV1	Patient Visit	3

- Each message shall be acknowledged by the HL7 ACK message sent by the receiver of ADT message to its sender. See ITI TF-2x: C.1.3 Acknowledgement Mode, for definition and discussion of the ACK message.
- This transaction does not require Patient Identity Source Actors to include any attributes not already required by the corresponding HL7 message. This minimal set of requirements enables inclusion of the largest range of Patient Identity Source Actor systems. This transaction does place additional requirements on the Document Registry Actors, requiring them to accept a set of HL7 attributes beyond what is required by HL7:

1.) MSH Segment: The MSH segment shall be constructed as defined in ITI TF-2x: C.1.2 Message Control. Field MSH-9 Message Type shall have at least two components. The first component shall have a value of ADT; the second component shall have one of the values of A01, A04, A05 or A08 as appropriate. The third component is optional; however, if present, it shall have the following value for each corresponding message type:
   a. ADT_A01 for A01 message type
   b. ADT_A01 for A04 message type
   c. ADT_A05 for A05 message type
   d. ADT_A01 for A08 message type

2.) EVN Segment: The Patient Identity Source Actor is not required to send any attributes within the EVN segment beyond what is specified in the HL7 standard. See Table C.1-4 in ITI TF-2x: C.1.4 Common Segment Definitions for the specification of this segment.

3.) PID Segment: The Patient Identity Source Actor is not required to send any attributes within the PID segment beyond what is specified in the HL7 standard. This message shall use the field PID-3 Patient Identifier List to convey the Patient ID uniquely identifying the patient within a given Patient Identification Domain.The Patient Identity Source Actor shall provide the patient identifier in the ID component (first component) of the PID-3 field (PID-3.1). The Patient Identity Source Actor shall use component PID-3.4 to convey the assigning authority (Patient Identification Domain) of the patient identifier. Either the first subcomponent (namespace ID) or the second and third subcomponents (universal ID and universal ID type) shall be populated. If all three subcomponents are populated, the first subcomponent shall reference the same entity as is referenced by the second and third components.

4.) PV1 Segement: The Admit/ Register or Update Patient message is not required to include any attributes within the PV1 segment beyond what is specified in the HL7 standard.

ADT for A40

Semantics:

- The Patient Identity Feed transaction is an HL7 ADT message. The message shall be generated by the system (Patient Identity Source Actor) that performs the update whenever two patient records are found to reference the same person.

- An A40 message indicates that the Patient Identity Source Actor has done a merge within a

specific Patient Identification Domain. That is, MRG-1 (patient ID) has been merged into PID-3 (Patient ID).

- Note: Conventions used in this section as well as additional qualifications to the level of specification and HL7 profiling are stated in ITI TF-2x: Appendix C and C.1. The segments of the HL7 Merge Patient message listed below are required. The PV1 segment is optional.

Structure:

- Segments:

ADT	A40 Patient Administration Message	Chapter in HL7 v2.3.1
MSH	Message Header	2
EVN	Event Type	3
PID	Patient Identification	3
MRG	Merge Information	3
[PV1]	Patient Visit	3

- Each message shall be acknowledged by the HL7 ACK message sent by the receiver of ADT message to its sender. See ITI TF-2x: C.1.3 Acknowledgement Modes for definition and discussion of the ACK message.

- A separate merge message shall be sent for each pair of patient records to be merged. For example, if Patients A, B, and C are all to be merged into Patient B, two ADT^A40 messages would be sent. In the first ADT^A40 message, patient B would be identified in the PID segment and Patient A would be identified in the MRG segment. In the second ADT^A40 message, patient B would be identified in the PID segment, and Patient C would be identified in the MRG segment.

1.) MSH Segment: MSH segment shall be constructed as defined in ITI TF-2x: C.1.2 Message Control. Field MSH-9 Message Type shall have at least two components. The first component shall have a value of ADT; the second component shall have value of A40. The third component is optional; however, if present, it shall have a value of ADT_A39.

2.) EVN Segment: See ITI TF-2x: C.1.4 for the list of all required and optional fields within the EVN segment.

3.) PID Segment: The Patient Identity Source Actor is not required to send any attributes within the PID segment beyond what is specified in the HL7 standard. This message shall use the field PID-3 Patient Identifier List to convey the Patient ID uniquely identifying the patient within a given Patient Identification Domain.The Patient Identity Source Actor shall provide the patient identifier in the ID component (first component) of the PID-3 field (PID-3.1). The Patient Identity Source Actor shall use component PID-3.4 to convey the assigning authority (Patient Identification Domain) of the patient identifier.Either the first subcomponent (namespace ID) or the second and third subcomponents (universal ID and universal ID type) shall be populated. If all three subcomponents are populated, the first subcomponent shall reference the same entity as is referenced by the second and third components.

4.) MRG Segement: The PID and PV1 segments contain the dominant patient information, including patient identifier and the issuing assigning authority. The MRG segment identifies the "old" or secondary patient records to be de-referenced. HL7 does not require that the "old"record be deleted; it does require that the "old" identifier shall not be referenced in future transactions following the merge. The Patient Identity Source Actor shall send the "old" patient identifier (to be merged) in MRG-1, with the identifier value in the component MRG-1.1 and the assigning authority in the component MRG-1.4. The Patient Identity Source Actor shall populate the same value of the assigning authority in PID-3.4, in the component MRG-1.4. IHE does not require that the Patient Identity Source Actor send any attributes within the MRG segment beyond what is specified in the HL7 standard.

5.) PV1 Segement: The Admit/ Register or Update Patient message is not required to include any attributes within the PV1 segment beyond what is specified in the HL7 standard.

9.1.3.56 Federated Document Source/Repository (SubPattern)

description:

A single information system may act as both Document Source and Document Repository for the documents it creates and registers with the Document Registry. Document Consumers will issue retrieval requests to this federated actor in its role as a repository.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet.

Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

Ort	Datum	Unterschrift