Friedrich-Alexander-Universität Erlangen-Nürnberg
Technische Fakultät, Department Informatik

BENJAMIN MEMPEL
MASTER THESIS

DEFINITION EINER DSL MITTELS QDA

Eingereicht am 9. Dezember 2014

Betreuer: Prof. Dr. Dirk Riehle, M.B.A.
Professur für Open-Source-Software
Department Informatik, Technische Fakultät
Friedrich-Alexander-Universität Erlangen-Nürnberg

Versicherung

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

Erlangen, 9. Dezember 2014

License

Erlangen, 9. Dezember 2014

Abstract

The design of a domain-specific language, DSL, requires extensive technical and professional know-how. Essential is not only the creation of syntax, semantics and related tools of language, but also an extensive knowledge of the domain. In order to acquire the knowledge of domain experts, we present an approach based on qualitative data analysis (QDA). We illustrate the process of creating a DSL based on the derivation of a domain model of qualitative analyzed interviews with domain experts. We then present the integration of the acquired specialized concepts and activities in a DSL. Qualitative feedback from domain experts could confirm the successful implementation of the DSL and according to that also the entire process.

Zusammenfassung

Die Erstellung einer domänenspezifischen Sprache, DSL, erfordert sowohl umfangreiche technische als auch fachliche Kompetenz. Entscheidend ist nicht nur die Erstellung von Syntax, Semantik und zugehöriger Tools der Sprache, sondern auch eine umfangreiche Kenntnis der Domäne. Zur Erhebung des Fachwissens von Domänenexperten stellen wir einen Ansatz auf Basis von qualitativer Datenanalyse (QDA) vor. Wir veranschaulichen den Entstehungsprozess einer DSL anhand der Ableitung eines Domänenmodells aus qualitativ analysierten Interviews mit Domänenexperten. Anschließend stellen wir die Integration der gewonnenen Fachkonzepte und Aktivitäten in eine DSL dar. Mittels qualitativer Feedbacks von Domänenexperten konnte eine erfolgreiche Umsetzung der DSL und damit verbunden des gesamten Prozesses bestätigt werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

API ..... Application Programming Interface
AST ..... Abstract Syntax Tree
CPR ..... Curved Ülanar Reformation
CT ..... Computertomography
DARE ..... Domain Analysis and Reuse Environment
DSL ..... Domain Specific Language
EBNF ..... Erweiterte Backus-Naur-Form
EMF ..... Eclipse Modeling Framework
FAST ..... Family-Oriented Abstractions, Specification and Translation
FODA ..... Feature-Oriented Domain Analysis
GPL ..... General Purpose Language
IDEF0 ..... Integration Definition for Function Modeling
MINIP ..... Minimum Intensity Projection
MIP ..... Maximum Intensity Projection
MPR ..... Multiplanare Rekonstruktion
MPS ..... Meta Programming System
MRT ..... Magnetresonanztomographie
PACS ..... Picture Archiving and Communication System
PET ..... Positronen Emissions Tomographie
QDA ..... Qualitative Datenanalyse
ROI ..... Region Of Interest
SPECT ..... Single-Photon Emission Computed Tomography
VRT ..... Volumen Rendering Technik
XML ..... Extensible Markup Language

1 Einführung

Domänenspezifische Sprachen finden in der heutigen Softwareentwicklung eine immer größere Verbreitung, wie dies von Fowler (2010) und Voelter, Benz et al. (2013) übereinstimmend beschrieben wird. Diese Sprachen sind auf ein spezielles Anwendungsgebiet, die Anwendungsdomäne, fokussiert. Zum einen bietet die Verwendung einer solchen Sprache Möglichkeiten zur Reduktion des Aufwandes, der durch die Spezifikation und zugehörige Implementierung von Anwendungen verursacht wird. Zum anderen ermöglicht sie Domänenexperten einen einfachen Einstieg, um Probleme ihrer Domäne mit fest definierten Begrifflichkeiten darzustellen (Van Deursen, Klint & Visser, 2000; Mernik, Heering & Sloane, 2005; Fowler, 2010; Voelter, Benz et al., 2013).

Im Kontext dieser Arbeit stellen wir eine domänenspezifische Sprache im Bereich des Medical Imaging der Siemens Healthcare AG vor. Software wie syngo.via der Siemens Healthcare AG, bietet Anwendern wie Ärzten oder medizinisch-technisch Angestellten eine Unterstützung bei der Befundung klinischer Aufnahmen wie CT- oder MRT-Aufnahmen (Siemens Healthcare AG, 2014c). Durch die breite Funktionspalette, die Nutzern zur Verfügung gestellt wird, entstehen komplexe Workflows, die häufig in ähnlicher Art und Weise ausgeführt werden. So entstand die Idee, solche Workflows mittels einer domänenspezifischen Sprache beschreiben zu können, um standardisierte Befundungsergebnisse zu erhalten. Ebenso kann die Sprache von Produkt Coaches oder Testern verwendet werden, um standardisierte Abläufe zu erzeugen.

Die Entwicklung von domänenspezifischen Sprachen wirft jedoch Probleme auf, die in verschiedenen Publikationen beschrieben werden (Kang, Cohen, Hess, Novak & Peterson, 1990; Van Deursen et al., 2000; Mernik et al., 2005; Fowler, 2010; Voelter, Benz et al., 2013). Zum Beispiel wird ein hoher Anspruch an die Entwickler und deren Kompetenz gestellt. Es wird zum einen Domänenwissen benötigt, zum anderen ist Expertise bei der Entwicklung von Sprachen notwendig. Weiterhin können sich Schwierigkeiten beim Finden eines entsprechenden Abstrak-

tionsniveau einer Sprache ergeben. Außerdem ist das Ermitteln der notwendigen Anforderungen und Sprachmerkmale ein zentrales Problem bei der Entwicklung domänenspezifischer Sprachen.

1.1 Forschungsfrage

Im Kontext dieser Arbeit wird explorativ ein neuer Prozess zur Ableitung einer domänenspezifischen Sprache aus qualitativen Daten getestet und bewertet. Mit Hilfe dieses neuen Prozesses klären wir die Frage, ob eine Ableitung einer domänenspezifischen Sprache durch qualitative Datenanalyse (QDA) möglich ist. Der Prozess gliedert sich in zwei Schritte, dem Ableiten eines Domänenmodells und anschließend die Ableitung der Sprache. Mit Hilfe der QDA sollen relevante Informationen aus Interviews und anderen qualitativen Daten extrahiert werden, um ein Phänomen oder menschliches Verhalten, unter Einbezug des zugehörigen Kontextes zu verstehen (Glaser & Strauss, 1965; Merriam, 1998; Yin, 2010). In dieser Arbeit analysieren wir speziell das Verhalten der Anwender einer klinischen Software bei der Durchführung einer Befundung. Zur Verdeutlichung des Gesamtprozesses wird in dieser Arbeit ein Fallbeispiel definiert, welches die einzelnen Schritte von der Analyse der qualitativen Daten, über die Erstellung eines Domänenmodells, die Ableitung einer domänenspezifischen Sprache, bis hin zur qualitativen Bewertung der Sprache durch Anwender veranschaulicht.

1.2 Gliederung

Zunächst gehen wir in Kapitel 2 auf Merkmale qualitativer Forschung und die Analyse qualitativer Daten ein. Zum einen werden Möglichkeiten zur Gewinnung qualitativer Daten (siehe Abschnitt 2.2.3) dargestellt und zum anderen Vorgehensmodelle von der Datengewinnung bis zur Analyse umrissen (siehe Abschnitt 2.1). Insbesondere wird die Analyse der Daten durch Codierung nach Corbin und Strauss (2008) erläutert (siehe Abschnitt 2.4). Im Kapitel 3 werden Grundlagen zu domänenspezifischen Sprachen dargelegt und existierende Vorgehensweisen zur Erstellung solcher Sprachen umrissen. Häufig beschriebene Vorgehensweisen sind FODA (Kang et al., 1990) und FAST (Weiss et al., 1999), die beleuchtet werden. Neben den Vorgehensmodellen wird das Framework Xtext (Itemis AG, o. J.) (siehe Abschnitt 3.2.2) und die Entwicklungsumgebung MPS von JetBrains (Jetbrains, 2014a) (siehe Abschnitt 3.2.2) dargestellt, die eine umfangreiche Un-

terstützung bei der Entwicklung domänenspezifischer Sprachen bieten. Im nachfolgenden Kapitel 4 geben wir eine Erläuterung zur Domäne im Medical Imaging Bereich der Siemens Healthcare AG. Ausgehend von dem Kapitel zur qualitativen Forschung wird im Kapitel 5 auf die Gewinnung der qualitativen Rohdaten und deren Analyse eingegangen. Aufbauend auf die Analyse wird im anschließenden Kapitel 6 die Erstellung des Domänenmodells dargestellt. Die Entwicklung eines Featuremodells, wie dies bei FODA (Kang et al., 1990) beschrieben wird, in Verbindung mit einem Klassendiagramm, stellt einen zentralen Aspekt dar. Aus diesem Modell wird im Anschluss eine domänenspezifische Sprache abgeleitet und auf die Merkmale dieser Sprache eingegangen (siehe Kapitel 7). Die Bewertung der DSL wird durch qualitative Anmerkungen der Domänenexperten vorgenommen und im Kapitel 8 dargelegt. Abschließend wird in Kapitel 9 eine Zusammenfassung des hier durchgeführten Frameworks und ein weiterer Ausblick gegeben.

1.3 Abgrenzung der Arbeit

Ziel dieser Arbeit ist die Erprobung eines neuen Vorgehensmodells zu Erstellung einer DSL mit Hilfe von QDA. In dieser Arbeit wird kein Vergleich verschiedener Vorgehensmodelle zur Entwicklung von domänenspezifischen Sprachen und deren Bewertung unternommen. Ausgewählte Vorgehensmodelle werden umrissen. Ebenso wird kein Vergleich verschiedener Entwicklungsumgebungen und deren Eignung durchgeführt.

Des Weiteren wird in dieser Arbeit keine Betrachtung des mit dem Thema verwandten Requirements Engineering durchgeführt, da diese Aspekte bei jeglicher Erstellung von Software relevant und aus diesem Grund zu allgemein sind. Aus diesem Grund sei zu diesem Thema auf weiterführende Literatur verwiesen (Sommerville & Sawyer, 1997; Rupp, Simon & Hocker, 2009).

2 Qualitative Forschung

Um unsere Umwelt und die Dinge, die uns interessieren, zu verstehen und neues Wissen zu schaffen ist Forschung von Nöten. Die Wissenschaft bedient sich verschiedenster Ansätze und dient dem Erkenntnisgewinn, um neue Sachverhalte zu erschließen und nachzuvollziehen. Diese Ansätze lassen sich in qualitativer und quantitativer Forschung unterscheiden. In beiden Punkten ist das Erschließen des Forschungsfeldes, ein zentrales Anliegen. Wissenschaftler aus den Sozialwissenschaften ergründen menschliches Verhalten und deren Umfeld, dabei stoßen sie auf das Problem, menschliches Verhalten quantifizierbar und messbar zu beschreiben (Hancock, Ockleford & Windridge, 1998). In dieses Problemfeld tritt die qualitative Forschung, die untersucht, warum Vorgänge ablaufen, wie sie in unserem sozialen Umfeld ablaufen (Yin, 2010). Es gilt somit unvoreingenommen zu verstehen, wie Menschen ihr Umfeld wahrnehmen, wie sich Phänomene definieren und Sachverhalte erklären lassen. In diesen und weiteren Fällen empfiehlt sich die Verwendung qualitativer Forschung (Hancock et al., 1998; Yin, 2010). Neben der Erstellung einer Forschungsfrage kann eine zielgerichtete Literaturrecherche sinnvoll sein (Hancock et al., 1998). Weiterhin sollte im Prozess der Theorieentwicklung darauf geachtet werden, dass sich diese auf gewonnen Daten begründet (Yin, 2010). Im Kontext dieser Arbeit wollen wir mit Hilfe qualitativer Forschung verstehen, wie Nutzer einer medizinischen Software vorgehen, um medizinische Bilddaten auszuwerten und Befunde zu erstellen. Aus der Analyse dieser Daten wird in einem weiteren Schritt ein Domänenmodell und eine Domänenspezifische Sprache erstellt. Im Verlauf dieses Kapitels werden zunächst Merkmale qualitativer Forschung erläutert. Im Anschluss daran werden ausgewählte Vorgehensweisen qualitativer Forschung vorgestellt. Des Weiteren wird auf Möglichkeiten zur Datengewinnung eingegangen, um abschließend auf die Analyse der Daten, die qualitative Datenanalyse einzugehen.

2.1 Merkmale Qualitativer Forschung

Quantitative Forschung versucht die Realität zu verstehen und zu beschreiben, indem sie nach generellen Gesetzmäßigkeiten sucht (Yin, 2010). Das bedeutet, dass bei quantitativer Forschung die Fokussierung und Definition des Interessenfelds im Mittelpunkt steht, um externe Einflüsse minimal zu halten. Die qualitative Forschung versucht, den Kontext des Interessenfeldes in Beziehung zur realen Welt zu betonen, um andere Perspektiven in die Forschung einfließen zu lassen und die Dynamik der realen Welt zu beschreiben (Hancock et al., 1998; Laws & McLeod, 2004; Yin, 2010). Das Ziel qualitativer Forschung ist die Entwicklung von Erklärungen und das Verständnis für Phänomene in den zu untersuchenden Forschungsgebieten.

Laut Yin (2010) lässt sich die qualitative Forschung mit fünf Merkmalen beschreiben:

1. Die Bedeutung des Lebens von Menschen im Kontext der realen Welt zu studieren.
2. Unterschiedliche Sichten und Perspektiven von Personen in Studien darzustellen.
3. Die Bedingungen des Kontextes, in dem Menschen leben, von der Studie mit abzudecken.
4. Einblicke geben in existierende oder sich entwickelnde Konzepte, die helfen können, menschliches Sozialverhalten zu verstehen.
5. Das Bestreben, mehr als ein Beweismittel zu verwenden, um nicht von einer Datenquelle abhängig zu sein.

Bei der Erläuterung der aufgezählten Merkmale folgen wir Yin (2010). Im Kontext dieser Arbeit bedeutet das erste Merkmal, zu verstehen wie verschiedene Personen mit ausgewählten, medizinischen Abläufen umgehen und wie sie diese in ihren Arbeitsalltag integrieren. Der zweite Punkt stellt einen der wichtigsten Punkte qualitativer Forschung dar. Yin führt aus, dass es um das Verständnis und die Darstellung, welche Sichten und Perspektiven Personen in dieser Studie haben, geht. Im dritten Punkt wird der Kontext des Verhaltens adressiert, da dieser bestimmte Verhaltensweisen beeinflusst. Für die Arbeit ist hierbei vor allem interessant, zu verstehen, wer, wann, warum und mit welchen Bedürfnissen bestimmte Applikationen verwendet. Beim vierten Punkt geht es um das Verständnis des Verhaltens. Das Verhalten soll entweder durch bereits existierende Konzepte er-

klärt werden oder zur Entwicklung neuer Verhaltenskonzepte führen. Als letzter Punkt wird von Yin die notwendige Diversität von Daten angesprochen. Um das zu untersuchende Phänomen in seiner Ganzheit zu verstehen, ist es notwendig, mehrere Datenquellen zu verwenden. Die Diversität ist behilflich, Stärken und die Schwächen der jeweiligen Vorgehensweisen bei der Datensammlung aufzuzeigen. Dies soll das Vertrauen in die gewonnenen Daten stärken.Dieser Vorgang wird *Triangulation* genannt. Er soll dabei eine Unterstützung und Erhärtung der Konstrukte und Hypothesen liefern (Eisenhardt, 1989). Somit kann Diversität vor allem durch Dokumente, Beobachtungen und Interviews geschehen (Laws & McLeod, 2004). Diese Daten besitzen die Eigenschaft, sich nicht oder schwierig adäquat numerisch abbilden lassen (Hancock et al., 1998).

Merriam (1998) unterstreicht, dass bei qualitativer Forschung die Schaffung einer Bedeutung der Daten im Vordergrund steht und dies durch einen induktiven Kurs der Analyse und detailliert beschriebene Funde erreicht werden kann. Während sich Yin (2010) primär auf die Inhalte der Forschung bezieht, nimmt Merriam (1998) den Forscher in die Charakterisierung der qualitativen Forschung auf und unterstreicht dessen Rolle. Beide heben die intensive Beschreibung der gefundenen Daten und der zugehörigen Analyse hervor. Die Unterschiede der Charakterisierung stellen ein Hinweis auf die verschiedenen Möglichkeiten qualitativer Forschung dar. Da kein einheitliches Vorgehen bei der qualitativen Forschung existiert, werden wir im nächsten Abschnitt ausgewählte Vorgehensweisen erläutern.

2.2 Ausgewählte Vorgehensweisen qualitativer Forschung

Neben den zu untersuchenden Daten wird ein Vorgehen gewählt, das die Analyse unterstützt und dafür sorgt, dass alle Schritte nachvollziehbar sind. Im folgenden Abschnitt werden wir zum einen auf die *Grounded Theory* und zum anderen auf die *Fallstudienanalyse* eingehen, da diese der Datenanalyse in dieser Arbeit zu Grunde liegen. Neben diesen beiden Verfahren existieren weitere zur Analyse qualitativer Daten, dafür sei auf folgende Arbeiten verwiesen (Hancock et al., 1998; Merriam et al., 2002; Laws & McLeod, 2004; Yin, 2010).

2.2.1 Grounded Theory

Grounded Theory wurde als Ansatz entwickelt, der qualitative Daten aus einer beobachtenden Feldstudie im Gesundheitswesen auswertete (Glaser & Strauss, 1965, 1967). Es geht um die Entwicklung neuer Theorien auf der Grundlage von Sammlung und Auswertung von Daten zu einem bestimmten Phänomen. Sie stellt eine iterative Vorgehensweise zur parallelen Sammlung qualitativer Daten und deren Analyse dar, die durch bestimmte Verfahrensvorgaben einen umfassenden Forschungsansatz ermöglicht. Corbin und Strauss (2008) definieren später Grounded Theory als eine Methode, die eine Menge von Prozeduren verwendet um eine induktive abgeleitete, grundlegende Theorie über ein Phänomen zu entwickeln. Die gewonnene Theorie soll auf den empirisch gewonnenen Daten basieren und in diesen „grounded“, also verankert, sein. Bereits existierende Ansätze zur Analyse von qualitativen Daten wurden erst mit der Grounded Theory organisiert (Glaser & Strauss, 1967). Die Datenanalyse wurde im Weiteren verfeinert, so dass die Grundlagen der Studie beleuchtet und die zugrunde liegenden Konzepte untersucht werden (Strauss & Corbin, 1998). Dieses Vorgehen führt zu einem tieferen Verständnis des untersuchten Phänomens und ist dementsprechend in den Daten fundiert.

Die wichtigsten Schritte, die Glaser und Strauss (1967) zur Grounded Theory zusammenfasst, sind die systematische Analyse von Dokumenten, Interviewnotizen oder Feldnotizen, die durch kontinuierliches Kodieren und den Vergleich von Daten zu einer wohlgeformten Theorie führen sollen. Dabei stehen die Datensammlung und deren Analyse zu einer Theorie in einem korrelativen Verhältnis. Das heißt zum einen, dass die gesammelten Daten die Theorie beeinflussen und zum anderen nimmt die Analyse der Daten Einfluss auf die Datensammlung, sodass während der Analyse weitere Schritte zur Datensammlung entschieden werden. Es geht darum, aus einem Themenbereich relevante, theoretische Konstrukte zu extrahieren, die dazu dienen, aus der Theorie und den Daten eine Art innere Beziehung zu entwickeln. Auf dieser Art der Beziehung beruht die Grounded Theory (Glaser & Strauss, 1967).

Im folgenden Abschnitt erläutern wir Schlüsselelemente der Grounded Theory und folgen Glaser und Strauss (1967) und Strauss und Corbin (1998). Grundlegend liegt die Fokussierung auf hervorgehenden Erkenntnissen aus den gesammelten Daten, anstatt der a-priori Bildung von Konzepten durch Vorwissen wie beispielsweise Literaturreviews. Das Ziel ist, dass der Forscher keine Vorannahmen über das trifft, was er untersucht, so dass alle Konzepte aus den Daten entstehen. Dies kann schwierig sein, da üblicherweise erste Hypothesen beim An-

gehen eines Forschungsgebietes entstehen. Als weiterer Punkt sei die theoretische Stichprobenerhebung, das *theoretische Sampling*, genannt.

Das theoretische Sampling von Daten beruht darauf, die Daten so zu wählen, dass mit den neuen Daten ein tieferes und oder breiteres Verständnis erzeugt werden kann. Dabei muss entschieden werden, ob weitere Stichproben neue und relevante Erkenntnisse und Daten vermuten lassen. Zum einen soll das Datenvolumen vermindert und zum anderen nur wichtige Daten in die Studie aufgenommen werden. Beim theoretischen Sampling besteht eine enge Verbindung mit dem eigentlichen Analyseschritt, insbesondere dem Kodieren. Diese Art des Samplings ist somit in einen iterativen Prozess eingebunden, in dem kontinuierlich neue Daten erhoben werden, um diese parallel zu kodieren und zu analysieren. Somit wird während der Analyse entschieden, welche Daten im nächsten Schritt erhoben werden sollen. Die Kriterien für die Auswahl neuer Daten sind die Relevanz, in Bezug auf bereits erhobene Daten und in Bezug auf die Forschungsfrage und das gesetzte Ziel, welches mit den neuen Daten erreicht werden soll.

Weiterhin sollte eine *theoretische Sensitivität* gegeben sein. Diese beschreibt die Fähigkeit, zu erkennen, was in den vorhandenen Daten wichtig ist, um darin Bedeutung zu finden. Es kann hilfreich sein, naiv und ohne Vorwissen an das Forschungsgebiet heranzugehen.

Neben der Sammlung der Daten soll deren Analyse gleichzeitig ablaufen. Dieser Prozess wird mit dem Begriff des *permanenten Vergleichs* beschrieben. Er hilft dabei, die Forschungsfragen zu verfeinern und eventuell können analysierte Datenfunde neu interpretiert werden. Ein gleichzeitiger Ablauf kann sich als schwierig gestalten, deshalb wird empfohlen, während der Transkription Notizen zu den jeweiligen Daten anzufertigen.

Zusätzlich spielt die Betrachtung des Kontextes des zu untersuchenden Studienfeldes eine Rolle (Glaser & Strauss, 1965, 1967). Die Grounded Theory besitzt ihre Stärke in der Untersuchung von kleinen, ausgewählten Themengebieten im Kontext der Einbettung in ihre reale Umgebung. Partielle Phänomene können, unter Einbezug der Studienumgebung, detailliert untersucht werden. Die Analyse der Daten wird solang durchgeführt, bis die sogenannte *theoretische Sättigung* eintritt (Strauss & Corbin, 1998; Yin, 2010). Diese Sättigung tritt ein, wenn durch die Analyse weiterer Interviews keine oder wenige neue Rückschlüsse gewonnen werden können. Es wird vorausgesetzt, dass die Analyse der Daten und die Sammlung der Daten iterativ ausgeführt werden. Während der Analyse wird entschieden, ob weitere Daten benötigt werden oder nicht, wie dies beim theoretischen Sampling der Fall ist.

Ein wichtiger Unterschied zu anderen Ansätzen der qualitativen Forschung ist die Betonung der Entwicklung einer neuen Theorie und nicht ausschließlich die Auswertung qualitativer Daten (Corbin & Strauss, 1994). Eine weitere Methode zur Untersuchung qualitativer Daten ist die qualitative Analyse von Fallstudien, die im nächsten Abschnitt erläutert werden soll.

2.2.2 Fallstudienanalyse

Merriam (Merriam et al., 2002) beschreibt die Fallstudie als eine intensive Beschreibung und Analyse eines Phänomens oder einer sozialen Einheit, wie ein Individuum, eine Gruppe, eine Institution oder eine ganze Gemeinschaft. Des Weiteren beschreibt sie diesen Ansatz als Methode zur Beschreibung eines Phänomens in seiner Tiefe.

Das zu untersuchende Phänomen wird im Vergleich zu anderen Methoden der qualitativen Forschung deutlicher definiert, damit kann ein bestimmter Fall untersucht werden. Ansonsten wird kein spezifischer Fall, sondern qualitative Daten untersucht (Merriam et al., 2002). Dieser Ansatz empfiehlt sich, wenn es nicht möglich ist, alle Variablen zu kontrollieren, die für die Forschung von Interesse sind (Benbasat, Goldstein & Mead, 1987; Merriam, 1998; Yin, 2014). Das bedeutet, dass von Seiten der Forschung wenig oder kein Einfluss auf das Phänomen genommen werden kann und die Daten somit unverfälscht analysiert werden können. Einige Fallstudien sind rein beschreibende Studien, andere sind interpretierende oder evaluierende Fallstudien oder eben deren Kombination, auf die wir im Rahmen dieser Arbeit nicht näher eingehen (Merriam, 1998).

Ebenso ist die Gewinnung neuer Theorien durch Fallstudien möglich und wird von Eisenhardt (1989) beschrieben und im Folgenden dargestellt. Eisenhardts Framework unterteilt sich in acht Schritte. Zunächst wird die Forschungsfrage definiert und eventuell Vorannahmen getroffen. Im nächsten Schritt wird festgelegt, welche Fälle untersucht werden, um in einem weiteren Schritt festzulegen, wie die Daten erfasst werden sollen. Nach dem Erschließen des Feldes und dem Sammeln der Daten wird in die Analyse der Daten übergegangen. Aus der Analyse wird eine Hypothese geformt, um die sich daraus entwickelnde Theorie schärfen zu können. In vielen Fällen wird als nächster Schritt die Literaturrecherche empfohlen um unterstützende oder widersprechende Daten zu erhalten und diese mit der entstehenden Theorie zu vergleichen. Diese Schritte werden solang wiederholt, bis durch die Analyse keine neuen Erkenntnisse mehr gewonnen werden. Ausgewählte Aspekte dieses Ablaufs werden im Weiteren kurz erläutert. Bei Eisenhardt (1989)

Ansatz, kann die die Analyse der Daten auf der Grounded Theory beruhen, andere Ansätze sind möglich, bleiben hier aber unerwähnt.

Vor dem Beginn sollte eine initiale Definition der Forschungsfrage vorhanden sein (Eisenhardt, 1989). Dies hilft zum einen den Fokus der Forschung einzuschränken und zum anderen die Datenmenge zu verkleinern. Des Weiteren kann eine erste Erstellung von Konstrukten und Konzepten hilfreich sein, um der Theoriebildung eine Form zu geben, wenn dies möglich ist. Ebenso wie bei Grounded Theory sollte im Optimalfall zu Beginn keine Theorie vorhanden sein. Andernfalls können existierende theoretische Perspektiven zur Voreingenommenheit führen und die Funde limitieren. Abschließend kann es hilfreich sein, die entstehende Theorie mit dem Thema entsprechender Literatur zu vergleichen (Eisenhardt, 1989). Neben unterstützender Literatur, sollte ebenso nach widersprüchlicher Literatur gesucht werden, um ein umfassendes Bild zu erhalten. Der Prozess der Datenanalyse soll auch hier beendet werden, wenn die theoretische Sättigung (siehe Abschnitt 2.2) eingetreten ist (Eisenhardt, 1989).

Dieser Prozess stellt, wie die Grounded Theory, ein strikt iteratives Vorgehen dar (Eisenhardt, 1989), während der Analyse der ersten gewonnenen Daten wird entschieden, welche Daten weiterhin benötigt werden. Als Vorteile dieses Prozesses in Bezug auf die Arbeit, können die testbaren Konstrukte gesehen werden, die aus der Analyse entstehen (Eisenhardt, 1989). Dieser Prozess ermöglicht mit hoher Wahrscheinlichkeit eine valide neue Theorie, da eine enge Verzahnung mit den Daten gegeben ist. Insbesondere die ersten vier Schritte dieses Ansatzes sind für die Datengewinnung im Rahmen dieser Arbeit sinnvoll und werden in einer ähnlichen Art und Weise durchgeführt. Die Fallstudienanalyse empfiehlt sich dann, wenn „Wie“- und „Warum“-Fragen gestellt werden (Benbasat et al., 1987; Eisenhardt & Graebner, 2007; Lacey & Luff, 2009; Yin, 2014).

2.2.3 Abschließende Bemerkungen zu qualitativen Vorgehensweisen

In jedem Fall ist die ausführliche Erläuterung, wie eine neue Theorie gewonnen wurde oder ein Phänomen verstanden wurde wichtig. Die Vorgehensweisen sollen sicherstellen, dass der Prozess jederzeit nachvollzogen werden kann. Dies soll auch für diese Arbeit sichergestellt werden, so dass nachvollzogen werden kann, wie aus den zu untersuchenden, qualitativen Daten ein Domänenmodell und in einem weiteren Schritt eine DSL entwickelt wurde. In dieser Arbeit entschieden wir uns für die explorative Fallstudienanalyse in Verbindung mit der Grounded

Theory, da sich diese Verfahren anbieten, wenn keine Literatur zur spezifischen Forschungsfrage existiert (Eisenhardt & Graebner, 2007; Urquhart, Lehmann & Myers, 2010). Die Verknüpfung dieser beiden Methoden konnte in verschiedenen Arbeiten bestätigt werden (Eisenhardt, 1989; Eisenhardt & Graebner, 2007; Yin, 2010, 2014). Grundlegend für die Analyse ist die Datensammlung, auf die im Folgenden eingegangen wird.

2.3 Methoden zur Datensammlung

Es existiert eine Vielzahl von Methoden um qualitative Daten zu sammeln. In diesem Abschnitt gehen wir auf das Interview ein, da es in dieser Arbeit als zentrales Instrument zur Datengewinnung dient. Neben dem einfachen Interview mit Einzelpersonen sind weitere Methoden denkbar, wie das Interview von Zielgruppen mit mehreren Personen, die Beobachtung von Personen oder Situationen, sowie die Sammlung von Dokumenten und von Berichten (Hancock et al., 1998; Yin, 2010).

Das Interview lässt sich in verschiedene Kategorien unterteilen. Als erste Form eines Interviews sei die *strukturierte Variante* genannt (DiCicco-Bloom & Crabtree, 2006; Yin, 2010). Alle Fragen sind fest formuliert und werden jedem Interviewten in gleicher Art und Weise gestellt. Strukturierte Interviews eignen sich für die Gewinnung quantitativer Daten (DiCicco-Bloom & Crabtree, 2006).

Dem gegenüber steht das *unstrukturierte Interview*, das durch die Ausführungen des Interviewten geprägt ist (DiCicco-Bloom & Crabtree, 2006). Es entspricht eher einer freien Konversation, aus der gewünschte Daten gewonnen werden können.

Für die Gewinnung qualitativer Daten bietet sich somit eine Zwischenform der Interviews an, das *semi-strukturierte Interview* (DiCicco-Bloom & Crabtree, 2006). Yin (2010) bezeichnet diese Art in Zusammenhang mit qualitativer Forschung auch als *qualitatives Interview*. In dieser Form wird eine Reihe offener Fragen verwendet, die auf dem zu untersuchenden Phänomen basieren und zusätzlich den Einbezug des Kontextes des Phänomens ermöglichen (Yin, 2010). Ebenso ist es dem Interviewer möglich auf die Antworten des Interviewten einfacher zu reagieren, so dass das Interview flexibler gestaltet werden kann und Detailfragen intensiver geklärt werden können. Auf diese Weise werden weder der Interviewer noch die Interviewten bei deren Fragen oder Antworten eingeschränkt und haben damit beide Einfluss auf die adressierten Themen.

Weiterhin können neben dem Interview Notizen zum Interview angefertigt werden, da diese Beobachtungen und Eindrücke liefern, welche für die Analyse selbst hilfreich sein können (DiCicco-Bloom & Crabtree, 2006; Yin, 2010).

Das Interview stellt in dieser Arbeit das vordergründige Mittel der Wahl zur Datengewinnung dar. Nachdem entschieden wurde, wie die Daten für die Studie gewonnen werden, ist es wichtig zu entscheiden, welche Daten relevant sind und ihren Beitrag für die Studie leisten sollen. Dies soll ebenso beitragen, den Umfang der Daten zu minimieren. Die Auswahl der Daten wird Sampling genannt. In dieser Arbeit wählen wir die Daten mit dem theoretischen Sampling nach Strauss und Corbin (1998) aus, welches in Abschnitt 2.2 erläutert wurde. Für weitere Verfahren, wie das *Random Sampling* oder das *Schneeball Sampling* sei auf folgende Publikationen verwiesen (Hancock et al., 1998; Yin, 2010). Ziel der Datenerhebung sollte es sein, möglichst eine theoretische Sättigung (siehe Abschnitt2.2) zu erreichen. Nach und insbesondere schon während der Datensammlung wird zum zentralen Punkt der qualitativen Forschung übergegangen, der Analyse, die im nächsten Abschnitt diskutiert wird.

2.4 Analyse qualitativer Daten QDA

Nachdem erste Daten gesammelt wurden, wird zum zentralen Schritt der qualitativen Forschung übergegangen, der Analyse und Auswertung der gewonnenen Daten. Zur Analyse der Interviews dieser Arbeit soll primär nach dem Analyseprozess der Grounded Theory vorgegangen werden. Als zentrales Element der Analyse präsentiert sich das Kodieren der Daten. Eng mit dem Kodieren verbunden,ist das bereits erläuterte theoretische Sampling.

2.4.1 Kodieren

Das Kodieren stellt den zentralen Analyseprozess der Grounded Theory dar, dessen Ziel nicht nur die Markierung und Beschriftung von Textstellen ist, sondern den kompletten Analyseprozess beschreibt und damit zur Bildung einer neuen Theorie führt. In dieser Arbeit ist die Erstellung einer neuen Theorie nicht das vordergründige Ziel, vielmehr soll der Analyseprozess verwendet werden, um aus den Daten ein Domänenmodell abzuleiten und daraus eine Domänen Spezifische Sprache zu generieren. Bevor wir auf die einzelnen Schritte eingehen, sei erwähnt, dass das Kodieren und die Analyse im Allgemeinen keiner festen Rei-

henfolge unterliegen (Yin, 2010). Es ist dem Forscher überlassen, welche Schritte er unternimmt, um die Daten zu analysieren. Er muss somit eine geeignete Kombination und Anordnung von Schritten einer optimalen Analyse finden. Aus diesem Grund gibt es eine große Anzahl verschiedener Vorgehensweisen der qualitativen Forschung. Eine der Analyse wichtige, zugrunde liegende Methode ist die des permanenten Vergleichs (Strauss & Corbin, 1998; Yin, 2010). Sie beschreibt den fortwährenden Vergleich, der aus der Analyse gewonnenen Daten mit neu hinzugekommenen Daten um beispielsweise die Interviewfragen anzupassen. Gegebenenfalls ist eine Neukodierung analysierter Daten notwendig. Der Analyseprozess ist somit ein iterativer Prozess, der aus der Auswahl von neuen Daten, aus der Analyse der Daten und der daraus resultierenden Gewinnung einer Theorie besteht.

Der Prozess des Kodierens unterteilt sich nach Strauss und Corbin (1998) in drei Analysephasen:

- Im ersten Schritt, dem *offenen Kodieren*, werden Konzepte benannt und aus dem Text Kategorien gewonnen.
- Der nächste Schritt ist das *axiale Kodieren*, bei dem übergeordnete Kategorien gefunden und die Beziehungen zwischen den Kategorien herausgearbeitet werden. Aus diesem Schritt resultiert eine hierarchische Struktur der Kategorien.
- Im dritten Schritt, dem *selektiven Kodieren*, soll die Hauptkategorie gefunden und die Theorie ausgearbeitet werden.

Ein weiterer wichtiger Aspekt, der während des Kodierens ständig durchzuführen ist, ist das Schreiben der sogenannten theoretischen Memos (Charmaz, 2008, S. 166). Dies sind kurze Texte, die dem Erfassen von Ideen zur Theorie dienen und somit dem Forscher helfen, einen einfacheren Überblick über erste Gedanken zu bekommen, die damit sofort zur Theoriegewinnung eingebracht werden. Sie können somit eine Stütze für den Prozess des Kodierens und der Analyse sein. Im weiteren Verlauf sollen die drei Schritte des Kodierens nach Strauss und Corbin Strauss und Corbin (1998) detaillierter dargestellt und erläutert werden.

Offenes Kodieren

In diesem Schritt ist das Ziel, den in den qualitativen Daten enthaltenen Elementen eine ausdrucksstarke Bezeichnung zu geben. Es ist notwendig, den Text in Sinneinheiten zu zerlegen und deren Bedeutung zu interpretieren. Diesen Einhei-

ten werden *Codes* zugeordnet. Codes stellen eine Beschreibung der Einheiten dar. Auf diese Codes aufbauend werden Abstraktionen herausgearbeitet, sogenannte *Konzepte*. Ein weiterer Schritt ist das Zusammenfassen der Codes und Konzepte zu *Kategorien*. Mit Hilfe dieser Kategorien und auf Basis ebendieser wird die Theorie erstellt. Wichtige Elemente der Theorie sind ebenfalls, die Eigenschaften und Dimensionen der Konzepte und Kategorien. Jede dieser gewonnenen Kategorien besitzt spezifische Eigenschaften, Attribute, die die Kategorien definieren und ihnen eine Bedeutung geben. Die *Dimensionen*, als letzter Begriff, beschreiben die Variation der Eigenschaften einer Kategorie. Somit unterstreichen sie die Variationen einer Theorie. Beispielhaft erklären wir das Kodieren an den in dieser Arbeit geführten Interviews. Häufig wurde beschrieben, dass Anwender zum Beispiel Abstände oder Ausdehnungen messen. Solche Beschreibungen wurden mit dem Code *measure distance* versehen. Wenn der Text ausreichend in Konzepte unterteilt wurde und die Kategorisierung abgeschlossen ist, folgt das axiale Kodieren.

Axiales Kodieren

Beim axialen Kodieren geht es darum, gewonnene Konzepte zu verfeinern und diese zu strukturieren. Mit diesem Vorgehen sollen wichtige Kategorien für die Theorie herausgearbeitet werden, welche als *Achsenkategorien* bezeichnet werden. Für die Gewinnung neuer Kategorien sind die Verknüpfung und die Beziehung zwischen Achsenkategorien und damit zwischen den Konzepten grundlegend. Des Weiteren sollen die Beziehungen zwischen den Achsenkategorien und deren Subkategorien ermittelt werden. Ziel des axialen Kodierens ist die Erstellung eines durchgängigen Kategoriennetzes aus Beziehungen zwischen den herausgearbeiteten Hauptkategorien und den dazugehörigen Subkategorien. Die Beziehungen zwischen den Kategorien können auf Zeit, Raum, Ursache-Wirkung und weiteren Faktoren basieren. Während der Analyse dieser Arbeit konnten wir mehrere Achsenkategorien definieren. Wie wir in Kapitel 4 darstellen, besteht die Domäne aus verschiedenen Applikationen. In den Interviews wurden Operationen und Eigenschaften beschrieben, die in allen Applikationen Verwendung finden. Während des Kodierens konnte eine Achsenkategorie *general operations* gebildet werden, die alle übergreifenden Operationen beinhaltet. Der anschließende Schritt, das selektive Kodieren, dient dem Auffinden der Kernkategorie.

Selektives Kodieren

Das selektive Kodieren unterscheidet sich grundlegend wenig vom axialen Kodieren. Hierbei bewegt man sich auf einem abstrakteren Level des Kodierens. Die gefundenen Kategorien werden hier durch erneutes Klassifizieren auf ein höheres, abstrakteres Level gehoben um eine *Kernkategorie* zu erhalten. Diese Kernkategorie wird durch die enthaltenden Kategorien beschrieben und kann als das zentrale Phänomen der Analyse betrachtet werden. Alle Fragen und insbesondere die Forschungsfrage zielen auf dieses Phänomen ab. Ist die Kernkategorie in den wichtigen Bestandteilen umfassend vorhanden und verstanden, dann kann von der theoretischen Sättigung ausgegangen werden. Wurde keine theoretische Sättigung erreicht, sollten vorherige Schritte wiederholt und gegebenenfalls weitere Daten analysiert werden. Im Kontext dieser Arbeit kann eine Kernkategorie beschrieben werden, die den diagnostischen Ablauf aus Sicht der Anwender darstellt (siehe Kapitel 5).

2.4.2 Abschließende Bemerkungen zur Analyse

Neben der Erstellung einer neuen Theorie ergibt sich die Möglichkeit Hypothesen aus der Theorie abzuleiten, die mittels der analysierten Daten verifiziert werden können (Strauss, 2004). Weiterhin entsteht die Möglichkeit, die entstandene Theorie und die dazugehörigen Hypothesen mit der Datenbasis zu testen, um mögliche Unstimmigkeiten zwischen den Daten und den entstandenen Interpretationen aufzudecken. Das Kodieren der Daten, wie es in der Grounded Theory beschrieben wurde, ist nicht die einzige Vorgehensweise, aber wohl das häufigst angewandte Vorgehen (Yin, 2010). Yin (2010) beispielsweise schlägt ein Vorgehen in fünf Punkten vor, dass wir aber nicht weiter erläutern. Wie von uns dargestellt, ist es für die Analyse der Daten notwendig, das Vorgehen aufzubereiten, damit es jederzeit nachvollzogen werden kann und somit die Ergebnisse der Analyse plausibel dargestellt werden. Die Analyse der Daten, die wir in dieser Arbeit untersuchen, folgt dem Verfahren des Kodierens der Grounded Theory nach Strauss und Corbin (1998).

2.5 Zusammenfassung Qualitative Forschung

Die qualitative Forschung stellt ein iteratives Vorgehen dar, das aus mehreren, parallelen ablaufenden Schritten besteht. Zu Beginn steht die Definition der For-

schungsfrage, durch die das Forschungsfeld definiert wird. Im weiteren Vorgehen muss entschieden werden, welche Daten erschlossen werden sollen und auf welche Weise dies geschehen soll. So muss entschieden werden, ob zum Beispiel mit Interviews gearbeitet werden soll oder ob existierende Dokumente dafür verwendet werden. Wenn erste Daten vorhanden sind, kann mit deren Analyse begonnen werden. Hierbei ist zu beachten, dass die Analyse nicht losgelöst von den anderen Schritten stattfindet, sondern parallel laufen kann, um zum Beispiel zu entscheiden, welche Daten verwendet werden. Wenn durch die Analyse eine entsprechende Theorie geformt oder bestimmte Hypothesen gewonnen werden konnten, kann es hilfreich sein, eine Literaturrecherche durchzuführen. Diese dient dem Vergleich, um eventuell vorhandene Literatur zu bestimmten Themen zu erschließen, diese zu verfeinern, oder sogar zu verwerfen. Abschließend sollten die Daten entsprechend aufbereitet und dargestellt werden. Viele der hier beschriebenen Schritte stellen kein striktes Vorgehen dar und lassen dem Forschenden einen entsprechenden großen Freiraum bei solchen Studien. Grundlegend ist, das Vorgehen während der Studie ausreichend detailliert zu beschreiben, so dass jederzeit eine Nachvollziehbarkeit der getätigten Schritte gewährleistet ist.

3 Domänenspezifische Sprachen – DSLs

Im Kapitel 2 sind wir auf qualitative Forschung und die Analyse qualitativer Daten eingegangen. Diese Analyse wird als neues Verfahren zur Erstellung Domänenspezifischer Sprachen mit dieser Arbeit verifiziert und getestet. In diesem Kapitel werden wir Domänenspezifische Sprachen, DSLs, allgemein erläutern und definieren. Anschließend wird zum einen auf die Domänenanalyse und zum anderen auf mögliche Implementierungen solcher Sprachen eingegangen. Im letzten Abschnitt gehen wir auf Verfahren zur Erstellung Domänenspezifischer Sprachen ein, um anschließend deren Herausforderungen darzulegen.

3.1 Allgemeine Erläuterung zu DSLs

Fowler (2010, S. 27ff.) definiert DSLs als Programmiersprachen mit einer eingeschränkten Aussagekraft, die sich auf eine bestimmte Domäne beziehen. Die Domäne beschreibt ein Spezialgebiet, mit dem sich Personen intensiv auseinandersetzen und ein hohes Wissen darüber besitzen (Duden, 2014). Eine DSL kann nicht nur als eine Programmiersprache verstanden werden, sondern allgemeiner als eine formale Sprache definiert sein. Anders als Programmiersprachen um formale Programme zu formulieren, können DSLs genutzt werden, um zum Beispiel Text strukturiert darzustellen wie mit XML oder die Möglichkeit, Darstellungen mittels einer Sprache zu beschreiben, wie dies in HTML möglich ist (Fowler, 2010, S. 101). Van Deursen et. al (Van Deursen et al., 2000) definieren domänenspezifische Sprache wie folgt:

> „A domain-specific language (DSL) is a programming language or executable specification language that offers, through appropriate notations and abstractions, expressive power focused on, and usually

restricted to, a particular problem domain.“(S. 1)

Dieser Definition folgend, können mittels domänenspezifischer Sprachen ausgewählte und relevante Aspekte einer Domäne kompakt beschrieben werden. Es sollte ein klarer Fokus auf die Domäne existieren, die zu dem klar abgegrenzt ist.

Domänenexperten sollen ohne tief gehende Programmierkenntnisse mit dieser Sprache umgehen können, in dem sie Notationen verwenden, die in ihrer Domäne gebräuchlich sind. Domänenexperten stellen Personen dar, deren Spezialgebiet eine bestimmte Domäne ist. Eine einfache Verständlichkeit der Sprache soll sichergestellt werden, indem sie geläufige Ausdrücke der Domäne enthält. Im Gegensatz zu einer Allzwecksprache (GPL - General Purpose Language) wie zum Beispiel Java, sollte es mit einer DSL nicht möglich sein, ein komplettes Software System zu erstellen, sondern ausgewählte Eigenschaften einer Domäne sollen durch die DSL unterstützt werden (Fowler, 2010, S. 27f.). DSLs lassen sich in verschiedene Kategorien einteilen und unterscheiden. Im folgenden Abschnitt soll auf eine Unterscheidung von DSLs eingegangen werden.

3.1.1 Arten von DSL

DSLs lassen sich in verschiedene Klassen unterteilen, um eine bessere Unterscheidung zu ermöglichen. Einen Schwerpunkt der Unterscheidung stellt die Art der Verwendung nach Voelter, Benz et al. (2013, S. 26f.) dar. Sie beschreiben zum einen technische DSLs, welche primär von Programmierern verwendet werden, also Personen die aktiv an der Entwicklung und Umsetzung von Programmen beteiligt sind. Zum anderen nennen sie Anwendungsdomänen DSLs, die von Personen verwendet werden, die keine Programmierer sind. Mit solchen DSLs soll es Domänenexperten ermöglicht werden, mittels einfacher Konzepte Programme oder Texte in einer Domänen Sprache auszudrücken.

Weiterhin ist eine Klassifizierung nach der Notation vorstellbar (Voelter, Benz et al., 2013; Fowler, 2010). Dies kann eine textuelle DSL sein, die wie übliche Programmiersprachen mittels eines Editors verwendet werden und ein Text eingegeben wird. Jedoch kann in einigen Szenarien eine grafische Notation sinnvoll sein, da hier Konzepte schneller erfasst und verstanden werden können. Eine Entscheidung der Notation ist von den jeweiligen Gegebenheiten der Domäne abhängig und muss dementsprechend an Hand dieser entschieden werden.

Eine dritte Unterscheidung ist nach der Art der jeweiligen Implementierung der DSL möglich. Voelter, Benz et al. (2013, S. 50f.) und Fowler (2010, S. 28) un-

terscheiden jeweils interne und externe DSLs. Fowler (2010, S. 28f.) unterscheidet zusätzlich noch DSLs, die mit speziellen Entwicklungsumgebungen entwickelt werden. Externe DSLs besitzen eine eigene Sprache, also Syntax, die unabhängig von der Sprache ist, mit der eine Anwendung ausgeführt wird. Für diese Sprache kann ein eigener Compiler oder Interpreter verwendet werden, ebenso ist eine Übersetzung in eine GPL denkbar. Dem Gegenüber verwenden interne DSLs direkt den Code einer GPL. Dies bedeutet, dass Code der DSL korrekter Code der GPL ist. Es wird ein Bruchteil der eigentlichen Sprache verwendet, der zusätzlich einen anderen der Domäne entsprechenden Stil besitzt. Ein Programmierer der GPL kann ebenso die DSL verwenden und muss nur die neu entwickelten Konzepte erlernen. Die dritte Möglichkeit, die von Fowler benannt wird, ist die Verwendung von spezialisierten, integrierten Entwicklungsumgebungen, um DSLs zu definieren und zu erstellen. Diese werden von ihm als *Language Workbenches* bezeichnet. Neben der Sprachstruktur, der Syntax, werden zusätzlich Tools, wie ein Spracheditor oder ein Übersetzer zur Verfügung gestellt, die speziell auf die DSL abgestimmt sind. Diese speziellen Entwicklungsumgebungen bieten den Entwicklern viele Möglichkeiten und unterstützen die Sprachentwicklung, benötigen jedoch eine intensive Einarbeitungszeit. Für weiterführende Informationen zu Language Workbenches sei auf Fowler (2005, 2010) und den Abschnitt 3.2.1 verwiesen. Die Klassifikation der DSL sollte früh im Entwicklungsprozess geschehen und in jeden Fall in Abhängigkeit der Domäne betrachtet und entschieden werden. Nur so kann für Domänenexperten eine Sprache entwickelt werden, die deren Anforderungen gerecht wird.

3.1.2 Vor- und Nachteile DSL

Die Erläuterungen zu Vor- und Nachteilen Domänenspezifischer Sprachen folgen hierbei Voelter, Benz et al. (2013, S. 38ff.) und Van Deursen et al. (2000).

Ein erster Vorteil ist die Möglichkeit der gesteigerten Produktivität. Das Arbeiten kann effizienter gestaltet werden, da viel GPL-Code durch wenig DSL-Code ersetzt werden kann. Diese Kapselung kann durch Bibliotheken oder Frameworks erreicht werden, jedoch bietet die DSL eine geeignete Syntax, kann auf statische Fehler prüfen und besitzt einen IDE Support. Außerdem ermöglicht sie Domänenexperten das Schreiben von eigenen Programmen oder die Strukturierung von Text. Des Weiteren kann durch DSLs die Qualität gesteigert werden. DSLs besitzen einen klaren abgesteckten Rahmen, welcher besser testbar ist und zu einer besseren Wartbarkeit führt. Das zentrale Know-How des Business kann auf einem formalen und wartbaren Weg abgebildet werden. Weiterhin ist die einfache Aus-

tauschbarkeit des Code-Generators (siehe Abschnitt 3.2.2) ein weiterer Vorteil. Wenn die Sprache entwickelt wurde, ist eine Übersetzung der Sprache in anderen GPL Code möglich, es muss nur der Code-Generator ersetzt werden. Dieser Vorteil gilt für Language Workbench und externe DSL. Dies bietet die Möglichkeit, die Sprache in verschiedenen Zielsystemen einzusetzen und sie somit Plattformunabhängig ist. Des Weiteren kann durch eine DSL das Denken und Kommunizieren in und über die Domäne geschärft werden, da die Sprache stark an die Domäne angelehnt ist. Sie bietet somit die Möglichkeit einer klaren Kommunikation innerhalb der Domäne und eine Art eines Standards. Weitere Vorteile können die Beteiligung der Experten während der Entwicklung sein, oder die Entwicklung effizienten Zielcodes möglich ist.

Dem gegenüber stehen Herausforderungen oder Nachteile, die dafür sorgen, dass von der Entwicklung einer DSL abgesehen wird. Ein offensichtliches Problem ist der Erstellungsaufwand einer DSL. Eine Kosten-Nutzen-Analyse ist schwer durchzuführen, da die Akzeptanz der Nutzer schlecht einschätzbar ist. Anwendungsdomänen-DSLs besitzen einen eher schmalen Anwendungsfokus, der den hohen Investitionsaufwand schwer rechtfertigen lässt. Zusätzlich kann sich die Weiterentwicklung und Wartung problematisch gestalten. Zum einen muss die Sprache kontinuierlich an die Domäne angepasst werden, da sie sonst keinen Mehrwert bringt. Zum anderen werden weitere Ressourcen wie Entwickler, Zeit oder Erfahrung, benötigt. Eine ständige Weiterentwicklung und Wartung ist notwendig, damit die Benutzerakzeptanz gefördert wird. Weiterhin kann die Gefahr eines Tool Lock-ins bestehen. Zwischen verschiedenen Tools zur Entwicklung einer DSL besteht meist kein Austauschmodell, weshalb man auf ein Tool festgelegt ist. Weitere Herausforderungen kann die sogenannte „DSL-Hell“ sein, wenn verschiedene DSLs für eine Domäne entwickelt werden. Ebenso können kulturelle Herausforderungen problematisch sein, wie die Akzeptanz einer neuen Sprache. Auch müssen allgemeine Probleme der Softwareentwicklung, bei der DSL-Entwicklung beachtet werden. Neben dem Wissen für die Erstellung einer DSL, spielt das Domänenwissen eine große Rolle, das durch die DSL abgebildet wird.

Die Entscheidung für oder gegen eine DSL sollte auf langfristige Sicht abgewogen werden. DSLs können eine Produktivitätssteigerung bedeuten und die Kommunikation der Domänenexperten stärken (Fowler, 2010, S. 33), dafür müssen Investitionen und anderen Herausforderungen überwunden werden. Somit ist keine allgemeingültige Aussage, ob ein Mehrwert durch die DSL Entwicklung geschaffen wird, möglich, diese Entscheidung muss kontextabhängig getroffen werden. Dabei stellt sich die Frage, warum es nicht sinnvoller ist Frameworks, Bibliotheken oder APIs zu entwickeln. Auf diese Abgrenzung soll im folgenden Abschnitt

eingegangen werden.

3.1.3 Abgrenzung zu Bibliotheken, Frameworks und APIs

Bibliotheken, Frameworks und APIs bieten eine Art der Abstraktion und kapseln Funktionen, damit dem Nutzer die Arbeit in seiner Domäne erleichtert werden kann. Dies eröffnet die Frage, ob eine eigene vollwertige Sprache sinnvoll ist und sich der Aufwand lohnt, wenn eine Umsetzung in einer GPL möglich ist. Eigens entwickelte Sprachen für eine Domäne stellen die sauberste Form einer Abstraktion dar (Voelter, Benz et al., 2013; Fowler, 2010). All das, was für APIs, Frameworks oder Bibliotheken in einer GPL benötigt wird, damit diese eingebettet und ausgeführt werden, kann durch eine DSL stark reduziert werden (Voelter, Benz et al., 2013, S. 30). Dies stellt zum einen eine Reduktion des Codes dar, zum anderen können Ressourcen geschont werden, da sich eine DSL auf bestimmte Aufgaben konzentriert. Eine DSL erlaubt eine Notation und Syntax, welche die Abstraktionen präzise ausführt und die Interaktion mit dem Programm effizient und einfach gestaltet (Voelter, Benz et al., 2013; Fowler, 2010). Dies erlaubt Nicht-Programmierern einen einfachen Einstieg in die Entwicklung von Programmen in der Domäne, was durch eine anwenderorientierte Entwicklungsumgebung, die stark an die Domäne angelehnt ist, unterstützt werden kann (Voelter, Benz et al., 2013, S. 30). Diese Entwicklungsumgebung kann zusätzlich weitere nicht triviale Analysen und Checks unterstützen, wie zum Beispiel Code Completion, Syntax Highlighting, die Markierung von Fehlern, Debugging oder Refactoring, was auf die Domäne angepasst wird (Voelter, Benz et al., 2013; Fowler, 2010). Hierbei gestaltet sich die Abgrenzung zu internen DSLs als schwierig. Jedoch lässt sich feststellen, dass wenn die zu entwickelnde Sprache von Nicht-Programmierern genutzt werden soll, eine eigenständige Sprache sinnvoll ist. Bei Nutzern mit Programmiererfahrung bietet sich hier genannte Abstraktionen an, da die Entwicklung einfacher ist und ein Verständnis der GPL vorausgesetzt werden kann.

3.1.4 Bestandteile einer DSL

Ebenso wie GPLs besitzen domänenspezifische Sprachen spezielle Bestandteile. Es spielt keine Rolle, ob es allgemeine Programmiersprachen oder domänenspezifische Sprachen sind, diese Bestandteile enthalten alle Sprachen. Ein allgemeiner Aufbau einer DSL wird in Abbildung 3.1 von Völter (2009) verdeutlicht.

Im Folgenden erläutern wir die Bestandteile und folgen Völter (2009); Fowler

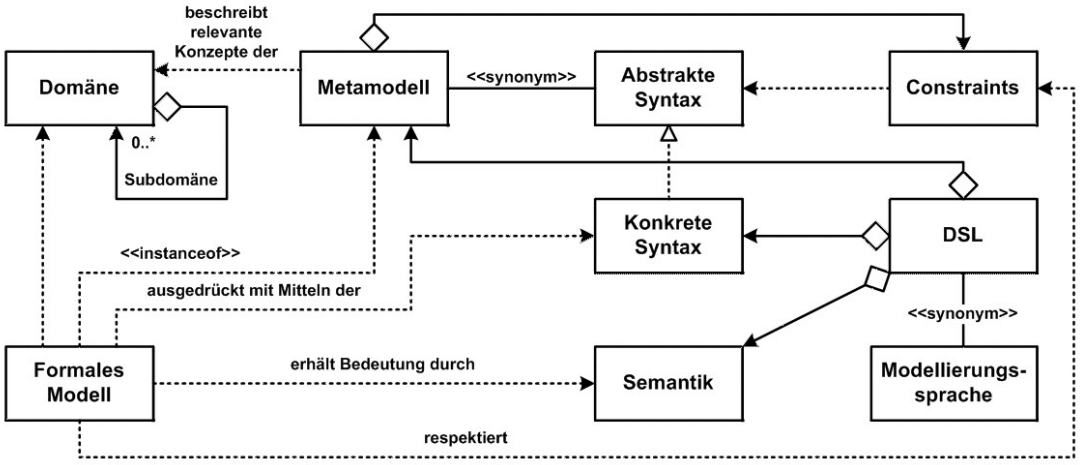

Abbildung 3.1: Zusammenhang Domäne und Bestandteile einer DSL (Völter, 2009)

(2010) und Voelter, Benz et al. (2013). Die Hauptelemente einer DSL sind die *konkrete Syntax*, das *Metamodell*, auch *abstrakte Syntax* genannt und die *Semantik*. Der offensichtlichste Bestandteil einer Sprache ist die konkrete Syntax, sie spezifiziert die Notation einer Sprache, mit der die Sprache dargestellt wird. Neben textuellen DSLs sind grafische, tabellarische oder eine Mischung dieser Darstellungen möglich.

Die abstrakte Syntax wird durch die konkrete Syntax dargestellt. Sie stellt eine Datenstruktur dar, die relevante semantische Informationen, einer Sprachinstanz, also eines Programmes präsentiert. Somit legt sie fest, was mit einer Sprache ausgedrückt werden kann, enthält aber keine Notationsdetails. Damit kann die Domäne der Sprache beschrieben und abgebildet werden. Typische Darstellung sind ein Graph oder ein Baum, wie der *Abstrakte Syntax Baum* (englisch: abstract syntax tree AST) (Jones, 2003).

Der dritte Bestandteil ist die Semantik, die Bedeutung von dem, was mit abstrakter und konkreter Syntax ausgedrückt wird. Sie unterteilt sich in die statische und dynamische Semantik. Die statische Semantik stellt eine Reihe von Beschränkungen und Typsystemregeln dar, die von einem Programm eingehalten werden müssen. Somit kann die Wohlgeformtheit eines Programms anhand der Konsistenzregeln getestet werden. Die Bedeutung eines Programms, wenn es ausgeführt wird, wird durch die Ausführungssemantik oder dynamische Semantik beschrieben. Erst mit der Semantik erlangt die Syntax eine Bedeutung.

Um eine DSL zu entwickeln, müssen zunächst die Vor- und Nachteile abgewogen werden und entschieden werden, ob daraus ein Gewinn entsteht. Des Weiteren muss über die Art der Umsetzung entschieden werden, ob es eine interne oder externe DSL wird, oder ob man sich für die Entwicklung mit einer speziellen Umgebung entscheidet. Um die Syntax einer DSL zu erstellen, ist es notwendig, die Sprache der Domäne zu kennen. Für diesen Schritt ist die Domänenanalyse bei der Erstellung einer neuen Sprache ein zentrales Element. Ebendiese Analyse soll im nächsten Abschnitt erläutert werden.

3.2 Domänenanalyse und DSL Implementierung

Die Analyse der Domäne und das Engineering sind die zentralen Schritte in der Entwicklung und Erstellung einer Domänenspezifischen Sprache. Das allgemeine Entwicklungsmodell einer DSL kann in mindestens drei Schritte untergliedert werden (Mernik et al., 2005; Hudak, 1997; Van Deursen & Klint, 1998).

1. Die Definition der Domäne, die Datensammlung und deren Analyse. Aufbauend auf dieser Analyse entwickelt sich ein Domänenmodell.
2. Die Entwicklung der DSL, die die semantischen Notationen implementiert. Inklusive der Entwicklung von Tools zur Verwendung und Unterstützung der Sprache.
3. Die Entwicklung von Anwendungen mit der DSL, also die Verwendung der Sprache selber.

Die Analyse der Domäne und die Entwicklung eines Domänenmodells können als getrennte Schritte verstanden werden, jedoch verknüpfen wir die Analyse und die Erstellung eines Domänenmodells in dieser Arbeit stark. Aus diesem Grund verstehen wir diese Punkte als einen Schritt. In dieser Arbeit liegt der Fokus auf der Erprobung einer neuen Methode zur Analyse einer Domäne und die Eignung daraus eine neue Sprache zu erstellen. Zunächst geben wir eine grundlegende Erläuterung um anschließend die Schritte der Implementierung zu beleuchten.

3.2.1 Allgemeine Erläuterung der Analyse

Das Ziel der Domänenanalyse ist ein Verständnis über die zugrunde liegenden Prinzipien und Konzepte. Um dieses Verständnis zu erhalten, ist es notwendig mehrere Schritte zu durchlaufen, um die Domäne zu analysieren (Van Deursen & Klint, 1998; Van Deursen et al., 2000). Es ist wichtig die Problemdomäne zu identifizieren und klar abzugrenzen. In einem weiteren Schritt werden Daten für die DSL gesammelt, die die Grundlage der Analyse sind. Basierend auf diesen Daten werden semantische Notationen und Operationen erstellt. Abschließend wird eine DSL entworfen, die die Anwendungen innerhalb der Domäne präzise beschreiben kann.

Quellen für die Datenanalyse kann dabei implizites oder explizites Domänenwissen sein, wie zum Beispiel technische Dokumentationen, Wissen von Domänenexperten, eventuell existierender Quellcode für ein Programm oder die Daten aus einer Kundenbefragung (Mernik et al., 2005). Das Ziel der Analyse sollte in jedem Fall eine konsistente domänenspezifische Terminologie sein und eine Semantik in einer abstrakten Form (Mernik et al., 2005). Die Analyse stellt einen kontinuierlichen Prozess der Verfeinerung des zu entwickelnden Modells dar (Prieto-Díaz, 1990). Dieser kontinuierliche Prozess unterstreicht die Verknüpfung der Analyse und der Erstellung des Domänenmodells.

Ein zentraler Punkt der Domänenanalyse ist der Einsatz eines Domänenanalysten, der mit Hilfe der Domänenexperten notwendige Informationen und Wissen aus den Daten extrahiert (Prieto-Díaz, 1990). Neighbors (1984) beschreibt den Domänenanalyst als eine Person, die für die Ausführung der Domänenanalyse verantwortlich ist. Ein Domänenanalyst muss laut Neighbors (1984) neben dem Verständnis für eine zu untersuchende Domäne, die Anwendungen und notwendigen Technologien zur Entwicklung einer Sprache verstehen, außerdem sollte er zwischen den verschiedenen beteiligten Personen vermitteln können. Die Domänenanalyse beinhaltet nicht nur die Analyse selbst, sondern bezieht den Schritt des Domänenengineerings mit ein (Van Deursen et al., 2000). Dieses Domänenengineering stellt dabei einen Ansatz zur systematischen Domänenmodellierung dar (Arango, 1989). Domänenengineering stammt aus dem Bereich der Softwarewiederverwendung und findet Verwendung bei der Entwicklung domänenspezifischer und wiederverwendbarer Bibliotheken, Frameworks oder Sprachen (Van Deursen et al., 2000). Van Deursen et al. (2000) und Mernik et al. (2005) sehen das Domänenengineering als einen Bestandteil der Analyse. Die Ziele der Analyse sind laut Mernik Mernik et al. (2005):

- Eine Definition der Domäne, die deren Rahmen festlegt.
- Die Terminologie der Domäne, die das Vokabular und Zusammenhänge enthält.
- Eine Beschreibung der Domänenkonzepte.
- Eine Beschreibung der Features der DSL, die notwendige und variable Konzepte enthält und deren Abhängigkeiten untereinander.

Die Terminologie der Domäne und deren Konzepte dienen der Entwicklung einer Sprache und deren Konstrukte, die mit einem Domänenmodell dargestellt werden. Laut Mernik (Mernik et al., 2005) lässt sich die Domänenanalyse in drei Gruppen unterteilen. Zum einen ist ein informaler Weg möglich, der kein festes Vorgehen aufweist und insbesondere durch Erfahrung in der Analyse geprägt ist. Eine weitere Möglichkeit wäre die Gewinnung von Domänenwissen aus existierendem GPL-Code durch Inspektion oder die Nutzung von Software Tools. Diese beiden Vorgehensweisen sind im Kontext dieser Arbeit uninteressant, da zum einen ein schwer nachvollziehbares Vorgehen bei der DSL-Erstellung existiert oder Code vorhanden sein müsste, was nicht der Fall ist. Die dritte Gruppe umfasst formale Vorgehensweisen, bei denen spezielle Methoden zur Domänenanalyse verwendet werden.

Mittels der gewonnenen Ergebnisse aus der Analyse kann die Implementierung einer Sprache begonnen werden, auf die wir im folgenden Abschnitt eingehen.

3.2.2 DSL Implementierung

In diesem Abschnitt gehen wir auf die Implementierung und das Design einer DSL ein. In Bezug auf die Klassifikation von DSLs entschieden wir die zu entwickelnde Sprache mit einer Language Workbench umzusetzen. Dies ermöglicht zum einen, dass Nutzer keine GPL erlernen müssten, wie dies bei internen DSLs der Fall ist (Fowler, 2010). Zum anderen müssen keine eigenen Tools entwickelt werden, die bei der Erstellung einer externen DSL nötig wären (Fowler, 2010). Zunächst wird auf grundlegende Anforderungen an DSLs eingegangen. Weiterhin wird die allgemeine Implementierung einer Sprache erläutert. Anschließend erläutern wir beispielhaft das *Meta Programming System* und *Xtext* als Language Workbenches.

Requirements für DSL

Im Folgenden werden wir wichtige Anforderungen, die an DSLs gestellt sind, erläutern. Viele der Prinzipien, die für GPLs gelten, können hierbei übernommen werden. Bei der Entwicklung DSLs existieren weitere Anforderungen (Kolovos, Paige, Kelly & Polack, 2006). Die nachfolgende Darstellung zentraler Anforderungen folgt Kolovos et al. (2006).

Als erste Anforderung sei hier die *Konformität* genannt, dies heißt, dass die Konstrukte der Sprache mit wichtigen und grundlegenden Konzepten der Domäne übereinstimmen. Die Konformität kann durch eine enge Verzahnung der Analyse und der Entwicklung einer Sprache unterstützt werden.

Ein weiterer Punkt kann die *Erweiterbarkeit* der Sprache sein. Diese ist von der Verwendung und Weiterentwicklung der Sprache abhängig. Wenn bei der Entwicklung abzusehen ist, dass die Sprache später um weitere Konstrukte und Konzepte erweitert werden soll, sollte auf eine einfache Erweiterbarkeit Wert gelegt werden.

Ebenso kann die *Integrierbarkeit* eine wichtige Rolle spielen. Für den Anwender kann es einfacher sein, wenn die Sprache in andere Sprachen oder Tools integriert werden kann und sich die Einarbeitung minimiert. Dies kann leicht durch interne DSLs erreicht werden, da diese auf GPLs und der Entwicklungsumgebungen basieren und nur neue Konzepte erlernt werden müssen.

Weiterhin soll die *Langlebigkeit* einer Sprache genannt werden. So soll sichergestellt werden, dass eine Sprache lange durch Tools und Weiterentwicklungen unterstützt wird. Dies erhöht zum einen die Benutzerakzeptanz, zum anderen machen sich durch ein langlebiges Produkt die Investitionen eher bezahlt. Das heißt, dass bereits bei der Entwicklung ein Fokus auf einen langen Lebenszyklus des Produktes gelegt werden sollte.

Des Weiteren soll als Anforderung die *Orthogonalität* einer Sprache sichergestellt sein, dies bedeutet, dass jedes Konstrukt einer Sprache exakt ein Konzept der Domäne abbildet. Ebenso sollte die Sprache Mechanismen unterstützten, um qualitative Systeme oder Ergebnisse zu ermöglichen.

Die *Qualität* eines Systems definiert sich durch dessen Zuverlässigkeit, die Sicherheit des Systems nach innen und außen, die Performance und weitere Kriterien.

Außerdem spielt die *Unterstützung* der Sprache durch entsprechende Tools und eine angepasste Entwicklungsumgebung eine wesentliche Rolle, da sie für die Akzeptanz entscheidend ist.

Als letzte zentrale Anforderung wurde die *Einfachheit* genannt. Die Konzepte einer Sprache sollten einfach gestaltet sein, damit diese für die Anwender leicht verständlich sind. Dies spielt für DSLs eine besondere Rolle, denn es soll Nutzern ermöglichen, sich in ihrer Sprache auszudrücken.

Neben den hier aufgezählten, können weitere Anforderungen wichtig sein, wie die *Skalierbarkeit* bei großen Systemen oder die *Benutzerfreundlichkeit* (Kolovos et al., 2006). Solche Anforderungen werden durch andere, wie die Einfachheit oder den Tool Support, unterstützt. Viele Anforderungen zielen darauf ab, die Nutzerakzeptanz zu stärken. Bei der Domänenanalyse sollte bereits auf die erläuterten Anforderungen eingegangen werden, um diese bei der Entwicklung optimal umsetzen zu können. Weiterhin sei darauf hingewiesen, dass sich Anforderungen schwer messen lassen und die Qualität einer DSL, sich durch die Anwenderakzeptanz widerspiegelt. Im nachfolgenden Abschnitt wird auf Aspekte der Sprachdefinition eingegangen.

Elemente der Sprachdefinition in Language Workbenches

Für die Implementierung einer Sprache lassen sich drei Bestandteile beschreiben, ein Schema für das semantische Modell, eine Umgebung um die DSL zu editieren und das Verhalten des semantischen Modells (Fowler, 2010; Voelter & Solomatov, 2010; Voelter, Benz et al., 2013). Da wir die DSL in dieser Arbeit mit einer Language Workbench umsetzen, wird die Umsetzung im Kontext dieser Workbenches erläutert. Diese Bestandteile sollen im folgenden Abschnitt näher erläutert werden und beruhen auf Ausführungen von Fowler (2010) und Volter (2011); Voelter und Pech (2012); Voelter (2013); Voelter, Benz et al. (2013). Um das semantische Schema abzubilden, wird ein Metamodell für die zu entwickelnde Sprache verwendet. Das Schema stellt die Datenstruktur und statische Bestandteile einer Sprache und deren Semantik dar. Mittels des Modells kann festgelegt werden, welche Sprachbestandteile auftreten können und wie sich deren Beziehungen gestalten. Somit bildet das semantische Schema die Grundlage für alle Konstrukte, die mittels der Sprache ausgedrückt werden können.

Ein weiterer wichtiger Punkt, der definiert werden muss, ist ein Editor für die DSL, der von den Endanwendern genutzt werden kann. Zum einen existieren Freitexteditoren, auf den wie üblich beliebiger Text eingegeben werden kann. Dieser freie Text wird anschließend durch einen Generator in Sprachkonstrukte übersetzt. Demgegenüber stehen projektionale oder strukturelle Editoren. Diese Editoren stellen das semantische Modell als eine Projektion dar und arbeiten so-

mit direkt auf diesem Modell. Die Eingaben der Endanwender können direkt auf syntaktische Fehler und Inkonsistenzen mit dem Modell überprüft werden. Mit Hilfe dieser projektionalen Editoren kann das semantische Modell in grafische, tabellarische Darstellungen oder Diagramme übertragen werden und die Erstellung von Programmen zusätzlich unterstützen.

Der dritte Punkt, das Verhalten des semantischen Modells, bezieht sich auf die Ausführung des geschriebenen DSL-Codes. Es ist notwendig, DSLs ausführbar zu machen oder sie in andere Sprachen zu übertragen. Für diese Fälle ist es erforderlich einen Interpreter oder einen Code-Generator zu entwickeln. Language Workbenches unterstützen beide Möglichkeiten. Der geschriebene DSL-Code kann direkt durch einen Interpreter ausgeführt werden. Die häufigste Variante ist die Übertragung des DSL-Codes in eine GPL oder Beschreibungssprache. Zum einen ist eine Texttransformation, zum anderen eine Model-to-Model Transformation möglich (Voelter, Benz et al., 2013). Bei der Model-to-Model Transformation wird das semantische Modell in ein Modell der Zielsprache durch einen Generator übertragen (Voelter, Benz et al., 2013). Dafür ist jedoch ein Modell der Zielsprache notwendig.

Die hier aufgezählten Aspekte der Sprachdefinition während der Implementierung sind bei der Erstellung einer domänenspezifischen Sprache zu berücksichtigen. Language Workbenches bieten einen einheitlichen Weg der Erstellung einer Sprache und integrieren die Aspekte der Sprachdefinition in sich. Nachfolgend wird auf zwei ausgewählte Language Workbenches eingegangen. Gründe für deren Auswahl waren zum einen, dass beides Open Source Anwendungen sind und zum anderen, dass beide gut dokumentiert sind. Außerdem bieten beide die Möglichkeit der Verwendung von Plug-Ins, die Austauschbarkeit der konkreten Syntax, Tooling-Support für Editoren, Modularisierung, Wiederverwendung und weitere Möglichkeiten. Neben diesen hier aufgeführten Language Workbenches existieren weitere, dafür sei auf folgende Publikationen verwiesen (Fowler, 2010; Erdweg et al., 2013).

Meta Programming System

In diesem Abschnitt stellen wir eine Charakterisierung des Meta Programming Systems MPS von JetBrains dar. MPS bietet eine eigenständige Entwicklungsumgebung, die auf die Entwicklung, Implementierung und Verwendung neuer Sprachen fokussiert ist. Eine Besonderheit ist die Verwendung eines sogenannten projektionalen Editors. Bei diesen Editoren wird die konkrete Syntax als eine

Projektion des ASTs dargestellt. Somit schreibt der Nutzer im eigentlichen Sinne keinen Text, sondern ändert den AST. MPS vermittelt den Eindruck eines normalen Texteditors. Dieses Konzept erfordert eine Umgewöhnungs- und Einarbeitungszeit, da es von gewohnter Programmierung abweicht. Weitere Besonderheiten sind die einfache Erweiterbarkeit und Kombination von Sprachen, die dazu beitragen komplexe Sprachen abzubilden. (Volter, 2011; Jetbrains, 2014a)

Da alle Schritte auf dem AST getätigt werden und somit alle Entwicklungsschritte im Modell vorliegen, entfällt das Parsing des Textes. Die Arbeit auf dem AST erspart die Definition der Grammatik einer Sprache, da durch den Benutzer der Sprache direkt Knoten im AST erstellt werden. Jeder Knoten des AST stellt ein Konstrukt oder eine Anweisung dar, die im Quellcode in der Sprachinstanz vorkommt. Je nach Güte der entwickelten Editoren für eine Sprache können diese den Endnutzer unterstützen und somit den Aufwand für das Erlernen der Verwendung einer neuen Sprache minimieren. Die Generierung von Ziel-Codes erfolgt durch die Model-to-Model Transformation, in dem der AST in ein Model der Zielsprache übersetzt wird. Bei MPS ist die Hauptsprache Java, aber es existieren weitere Implementierungen, wie zum Beispiel XML. Neben der Model-to-Model Transformation ist eine Textgenerierung möglich, um Knoten des AST in beliebigen Text zu übersetzten. Zusätzlich kann ein Interpreter für eine Sprache erstellt werden.(Voelter, Benz et al., 2013; Jetbrains, 2014a)

Die größte Herausforderung bei MPS ist die Umstellung auf die projektionale Editierung, da es ein ungewohntes Konzept darstellt (Volter, 2011; Jetbrains, 2014a). Zusätzlich ist immer die Verwendung der Entwicklungsumgebung notwendig, da durch die Projektion des AST keine Verwendung eines einfachen Texteditors möglich ist. Die Roadmap sieht dafür einen webbasierten Editor, sowie eine Integration in Eclipse vor (Pech, 2014).

Xtext

Im Gegensatz zu MPS setzt Xtext auf das Editieren von Freitext. Xtext ist ein Framework und basiert auf dem Eclipse Modeling Project (The Eclipse Foundation, o. J.). Dieser Abschnitt gibt einen Überblick über das Eclipse-Plug-In Xtext von itemis (The Eclipse Foundation, 2013; Itemis AG, o. J.). Wir folgen dabei Ausführungen vonVoelter, Benz et al. (2013).

Xtext stellt ein Framework zur Erstellung textueller domänenspezifischer Sprachen dar. Neben Syntax Highlighting und Code-Vervollständigung können Constraints geprüft und Debugging unterstützt werden. Für die Codegenerierung wird

die DSL standardmäßig in eine Java-artige Sprache übersetzt, ansonsten sind auch beliebige Zielsprachen umsetzbar. Die Darstellung des Datenmodells wird mit EMF/Ecore ermöglicht. Durch EMF wird die Überführung der Sprache in einen AST ermöglicht. Die Syntax in Xtext wird mit einer EBNF-artigen Notation entwickelt. Durch die Verwendung von Freitextprogrammierung ist die Verwendung von Parsern notwendig. Neben der Überführung in anderen Code, kann auch ein Interpreter für DSL entwickelt werden. Durch die Eclipseumgebung, kann Xtext mit weiteren Add-ons unterstützt werden.

Durch die Verwendung von Parsern kann nicht jede kontextfreie Grammatik geparst werden, was zu Problemen bei der Modularisierung von Sprachen führen kann. Vorteilhaft ist die große User-Community, die Unterstützung bei der Entwicklung bietet.

Abschließende Bemerkungen zur Implementierung

Für die Implementierung von Domänenspezifischen Sprachen ist die Entwicklung von drei Bestandteilen notwendig. Die Entwicklung eines semantischen Modells einer Sprache, ein Editor für die Verwendung der Sprache und ein Generator zur Überführung und Ausführbarmachung der mit der DSL geschriebenen Anwendungen. In dieser Arbeit wird die Implementierung der Sprache mit einer Language Workbench realisiert. Als Tool wurde MPS gewählt. Diese Entscheidung basiert zum einen auf der guten Dokumentation. Zum anderen ermöglicht der projektionale Editor von MPS dem Endanwender umfangreiche Unterstützungen bei der Entwicklung von Skripten mit der DSL. Des Weiteren kann sich der Entwickler auf die Definition des semantischen Modells konzentrieren und wird bei der Entwicklung des Editors und des Generators durch MPS unterstützt.

3.2.3 Vorgehensmodelle

Wir gehen in diesem Abschnitt auf Techniken zur Entwicklung von DSLs ein. Diese Methoden beschreiben formale Vorgehensweisen der Domänenanalyse (siehe Abschnitt 3.2) und gehen oft darüber hinaus.

In der Literatur werden häufig FODA (Feature-Oriented Domain Analysis - Merkmalsorientierte Domänenanalyse) (Kang et al., 1990), FAST (Family-Oriented Abstractions, Specification and Translation – Familienorientierte Abstraktionen, Spezifikationen und Translationen) (Weiss et al., 1999) und DARE (Domain Analysis and Reuse Environment - Domänenanalyse und Wiederverwendungs Umge-

bung) (W. Frakes, Prieto, Fox et al., 1998) als verwendete Methoden beschrieben (Alonso, 2002; W. B. Frakes & Kang, 2005; Mernik et al., 2005; Lisboa et al., 2010). Neben diesen Methoden existieren weitere wie zum Beispiel IDEF0 (Integration Definition for Function Modeling - Integrationsdefinition für Funktionsmodellieren) zur Verwendung für die Domänenanalyse (Imran, Foping, Feehan & Dokas, 2010). Weitere Methoden werden von Mernik et al. (2005) und Van Deursen et al. (2000) benannt. Diese Techniken stammen aus den Domänenengineering und damit dem Produktlinienengineering (W. B. Frakes & Kang, 2005). Jedoch kann deren Vorgehen bei der Domänenanalyse, auch der Erstellung einer DSL dienen und ist somit relevant für diese Arbeit. Um ein Verständnis für die hier genannten Methoden und deren Funktionsweise zu erhalten, gehen wir in den folgenden Abschnitten auf FAST, FODA und IDEF0 als beispielhafte Methoden ein und erläutern deren Vorgehensweise. DARE betrachten wir nicht näher, da es eine Softwareumgebung benötigt, für die wir keine aktuellen Implementierungen gefunden haben.

FAST – Family-Oriented Abstractions

FAST stellt einen Softwareentwicklungsprozess dar, der Architekturprinzipien von Produktlinien verwendet. Im Folgenden erläutern wir Eigenschaften dieser Methode zur Entwicklung einer DSL und folgen Coplien et al. (1998); Weiss et al. (1999) und Mernik et al. (2005). Die Verwendung von Architekturen der Softwareproduktlinien im Erstellungsprozess neuer Software ist zentral für FAST. Mit Hilfe dieser Architekturprinzipien wird eine Plattform für Softwareproduktfamilien geschaffen, die die Gemeinsamkeiten und Unterschiede der Produkte erfasst. Das Ziel dieses Prozesses ist es die Entwicklungskosten zu minimieren und die Zeit von der Entwicklung bis zur Einführung eines Produktes zu verkürzen. Zentrale Schritte sind das Domänenengineering und das Anwendungsengineering, zusätzlich wird die Wirtschaftlichkeit des Prozesses für eine Softwareproduktfamilie berechnet. Das Ziel des Domänenengineerings und insbesondere der enthaltenen Domänenanalyse ist es Gemeinsamkeiten und Unterschiede der Produkte einer Softwarefamilie zu identifizieren. Die Beschreibung der Domäne wird mit Strukturdiagrammen erreicht, um die Grenzen und die Beziehungen der Domänen nach außen zu beschreiben. Ausgehend von dem Domänenmodell, das durch die Domänenanalyse entsteht, kann eine DSL abgeleitet werden.

FODA - Feature-Oriented Domain Analysis

Als eine weitere Methode im Zusammenhang mit der Domänenanalyse beschreiben wir, Kang et al. (1990) und Mernik et al. (2005) folgend, FODA. Als drei grundlegende Schritte der Analyse werden die Kontextanalyse, die Modellierung der Domäne, die die Probleme und Konstrukte der Domäne beschreibt und die Modellierung der Softwarearchitektur zur Abdeckung der Domänenkonstrukte beschrieben. Eine zentrale Rolle bei der Kontextanalyse und der Domänenmodellierung spielt der Domänenanalyst, der gemeinsam mit Domänenexperten, unter der Verwendung von Strukturdiagrammen, Featuremodellen und Entity Realationship Modellen eine Domäne analysiert. Zunächst benötigt FODA die Erstellung eines Featuremodells. Dies enthält auf der einen Seite Gemeinsamkeiten, also Features die notwendig sind und in allen Daten vorhanden sind und auf der anderen Seite variable Features, die nicht in allen Ausgangsdokumenten vorhanden sind. Ein gemeinsames Feature ist in allen Instanzen des Konzeptes vorhanden. Ein variables Feature kann entweder optional oder alternativ sein und muss somit nicht vorhanden sein. Die Aufgabe des Featuremodells ist es eine hierarchische Zusammensetzung von allen notwendigen, alternativen oder optionalen Features zu ermöglichen. Des Weiteren enthält es Regeln, wie die Features zusammenarbeiten und zusammengesetzt werden können, sogenannte Constraints. Um aus dem gewonnenen Featuremodell eine DSL zu erstellen, ist es zunächst notwendig, die variablen und die notwendigen Features zu betrachten und dementsprechend einzubringen. Somit wird festgelegt, welche Features grundlegend in jeder Instanz der DSL festgelegt sein müssen, damit ein Programm ausgeführt werden kann. Die variablen Features sind ebenso im Ausführungsmodell der DSL enthalten, es wird jedoch erst bei der Erstellung einer Instanz festgelegt, ob das entsprechende Feature benötigt wird. Der zentrale Punkt bei FODA ist die Erstellung eines zur Domäne passenden Featuremodells, aus welchen notwendige und variable Features für die DSL abgeleitet werden.

IDEF0 - Integration Definition for Function Modeling

Als dritte Methode stellen wir IDEF0 nach Imran et al. (2010) vor. Der Fokus der Methode liegt auf dem Design, der Konstruktion und der Einführung einer neuen Sprache und weniger auf der Analyse. IDEF0 wurde entwickelt, um Aktivitäten, Prozesse und Aktionen einer Organisation oder eines Systems zu modellieren. Wichtige Schritte bei diesem Prozess sind eine Bedarfsanalyse, Nutzungsanalyse, Definition der Anforderungen, funktionale Analyse, das Systemdesign, Wartung

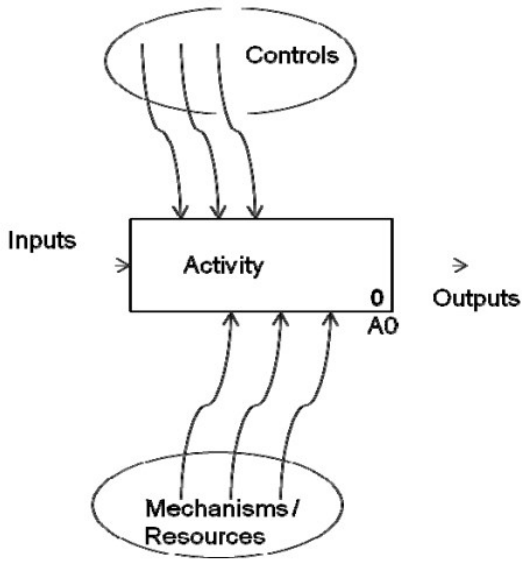

Abbildung 3.2: IDEF0 Sprachmodell (Imran et al., 2010)

und eine Basis für kontinuierliche Weiterentwicklung. Ebenso soll die Beteiligung von Domänenexperten durch einfache grafische Darstellung erhöht werden. Diese Darstellungen enthalten Aktivitäten, Eingaben, Ausgaben, Beschränkungen oder Kontrollen, sowie Mechanismen oder Ressourcen eines Systems (siehe Abb. 3.1). Da diese Diagramme hierarchisch strukturiert sind, kann die Problemdomäne stetig weiter verfeinert werden, solang bis eine ausreichende Beschreibung erreicht wurde. Die Strukturierung teilt sich in Einzelaktivitäten bis zu kompletten Prozessen, die aus mehreren Einzelaktivitäten bestehen. So kann eine ganze Domäne beschrieben und aus dieser Analyse eine Sprache entwickelt werden. Die Anforderungen, die an die Domäne gestellt werden, werden bei diesem Verfahren durch die Anforderungen der Domänenexperten abgedeckt und in die oben beschriebene Notation übersetzt.

Herausforderungen der Erstellung einer Sprache

Die Analyse der Domäne ist der zentrale Schritt, um neue Sprachen entwickeln zu können. Dementsprechend stellt sie die Entwickler vor entsprechende Herausforderungen, auf die hier eingegangen wird. Zunächst hängt die Qualität der Analyse stark von den Erfahrungen und dem Können des Domänenanalyst ab (Van Deursen et al., 2000; Neighbors, 1984). Neighbor (Neighbors, 1984) erachtet den Analyst als das zentrale Organ der Analyse, der neben Erfahrungen in der Softwareentwicklung, Domänewissen besitzen und ebenfalls ein Domänenexperte

sein sollte. Die hier vorgestellten Vorgehensmodelle setzen voraus, dass die Experten wissen, was für deren Domäne entscheidend ist (Kang et al., 1990). Das bedeutet, dass die Experten ein klares Bild über die notwendigen Features und Konzepte ihrer Domäne besitzen, nur dann kann die Domäne erfolgreich analysiert werden. Eine weitere Herausforderung ist die Verwendung von formalen Vorgehensmodellen. Wie Mernik et al. (2005) beschreibt, werden primär informelle Methoden zur Entwicklung neuer Sprachen verwendet. Der informelle Weg stellt dabei kein nachvollziehbares Vorgehen bei der Entwicklung dar und erschwert die Validierung der Sprache. Weiterhin ist die Definition des Abstraktionslevel ein Problem (Lind, 2003; Lintern, 2006), denn es muss entschieden werden, wie detailliert die Domäne dargestellt wird.

3.3 Zusammenfassung

In diesem Kapitel gaben wir einen allgemeinen Überblick zu DSLs. Ein Definition wurde beschrieben, allgemeine Vor- und Nachteile erläutert und wichtige Bestandteile einer solchen Sprache dargestellt. Die Hauptbestandteile jeder Sprache sind deren Metamodell, die konkrete Syntax und die Semantik. Anschließend wurde auf die Analyse der Domäne eingegangen. Die Analyse der Domäne wurde als ein zentrales Element der DSL-Entwicklung beschrieben. Ziele der Analyse sind die Domänendefintion, die Terminologie der Domäne, Domänenkonzepte und die Features der DSL. Anschließend wurde auf die Implementierung einer domänenspezifischen Sprache eingegangen. Zum einen werden Anforderungen an eine Sprache gestellt, die durch die Implementierung abgedeckt werden sollten. Zum anderen enthält die Implementierung drei wesentliche Elemente, ein Schema für das semantische Modell, eine Entwicklungsumgebung für die Sprache und einen Generator. Da die in dieser Arbeit zu entwickelnde Sprache, mit einer Language Workbench entwickelt wird, wurden beispielhaft MPS und Xtext vorgestellt. Abschließend wurde auf Vorgehensweisen bei der Entwicklung eingegangen. So lässt sich zwischen formalen und informalen Methoden unterscheiden. Als formale Methoden wurden FAST, FODA und IDEF0 näher erläutert. Diesen Methoden und weiteren Methoden ist gemeinsam, dass sie neben der Analyse Wege zur Implementierung einer Sprache vorstellen.

4 Überblick der Domäne

Im folgenden präsentieren wir einen Überblick, über die zu untersuchende Domäne, die Befundung medizinischer Bilddaten. An dieser Stelle gehen wir nicht näher auf die Befundung an sich ein. Zum einen werden bestimmte Schritte durch die in dieser Arbeit durchgeführten Analyse näher erläutert, zum anderen übersteigt dies den Rahmen dieser Arbeit. In dieser Arbeit wird die Domäne beispielhaft an der Befundung mit syngo.via (siehe Abschnitt 4.2) von Siemens untersucht (Siemens Healthcare AG, 2014c). Ebenso könnten Systeme anderer Hersteller eine Grundlage sein, da sie ähnliche Funktionen bieten. Wir gehen in dieser Arbeit speziell auf Applikationen zur Befundung des Blutkreislaufes, der Herzkranzgefäße und onkologischer Fragestellungen ein. Zur Analyse der Domäne führen wir Interviews mit Anwendern dieser Applikationen durch. So wird im Folgenden zunächst ein allgemeiner Überblick der Domäne erörtert. Anschließend wird syngo.via dargestellt und ein Überblick über die Applikationen zur Untersuchung des Blutkreislaufes, der Herzkranzgefäße und der Onkologie gegeben. Die Applikationen werden in dieser Arbeit alternativ als Tasks bezeichnet.

4.1 Allgemeines zur Domäne

Bildgebende Maßnahmen mittels CT oder MRT bilden heute ein wichtiges Werkzeug in der klinischen Diagnostik. Das zentrale Element in der zu untersuchenden Domäne ist die Vorgehensweise bei der Befundung und Erstellung einer Diagnose mit Hilfe der zuvor genannten Aufnahmen. Bevor eine elektronische Befundung dieser Aufnahmen möglich war, musste der Arzt die Bilder manuell auswerten. Mittlerweile werden von einigen Anbietern Systeme angeboten, die eine digitale Befundung und Auswertung von CT-, MRT- oder PET-Aufnahmen ermöglichen (Brant & Helms, 2012). Das Problem der Befundung ist, große Datenmenge in Form von Aufnahmen, in kurzer Zeit nach Auffälligkeiten zu durchsuchen. In vie-

len Fällen läuft diese Befundung nach einem ähnlichen Schema ab. Aus diesem Grund kann die Befundung mit vordefinierten Abläufen automatisiert und beschleunigt werden. Siemens stellt in syngo.via ein System zur Verfügung, *Rapid Results*, was die Definition kleinerer Abläufe bereits jetzt ermöglicht. In dieser Arbeit wird eine DSL erstellt, um allgemeine Abläufe der Befundung mit grundlegenden Konzepten, in ausgewählten Applikationen, beschreiben zu können.

4.2 Überblick zu syngo.via

syngo.via stellte eine Auswertestation dar, die medizinisch, diagnostische Applikationen für das Betrachten, die Manipulation, die 3D-Visualisierung und den Vergleich von medizinischen Aufnahmen ermöglicht (Siemens Healthcare AG, 2014c). Zu diesem Zweck wird eine Unterstützung von CT-, MRT-, PET- und SPECT-Datensätzen angeboten, die es ermöglichen anatomische und funktionale Ansichten darzustellen. Die Integration der Software in die Client-Server-Architektur bietet einer Klinik die Möglichkeit, die Aufnahmen eines Scanners direkt mit syngo.via zu laden, die entsprechende Befundung durchzuführen und die Aufnahmen und Ergebnisse anschließend im PACS zu archivieren. Das PACS (Picture Archiving and Communication System) stellt ein digitales Bildarchivierungs- und Kommunikationssystem dar, das die Langzeitarchivierung medizinischer Daten ermöglicht. Weiterhin unterstützt syngo.via Radiologen oder medizinisch-technisches Personal bei der Automatisierung der Befundungsschritte und ermöglicht zusätzlich automatisierte Unterstützung bei der Erstellung einer Diagnose. Eine Reihe von verschiedenen Applikationen bieten dem Anwender Funktionen, die für die Befundung klinischer Probleme zugeschnitten sind. Im Rahmen dieser Arbeit beschränken wir uns auf die Applikationen Coronary-, Oncology- und Vascular-Analysis, die in den folgenden zwei Abschnitten näher erläutert werden.

4.3 Befundung der Herzkranzgefäße und des Blutkreislaufes

In syngo.via wird für Untersuchungen des allgemeinen Blutkreislaufes die Vascular Analysis und für die Herkranzgefäße die Coronary Analysis zur Verfügung gestellt (Siemens Healthcare AG, 2014a, 2014d). Für beide Analyseapplikationen werden ähnliche Funktionen angeboten, da beide der Diagnostik von Blutgefäßen dienen. Mit der Vascularanalyse können allgemeine Gefäße untersucht werden

(siehe beispielhaft Abb. 4.1), während die Koronaranalyse eine Fokussierung auf die Herzkranzgefäße darstellt. In beiden Applikationen werden Hauptgefäße segmentiert, damit der Anwender sofort auf diese zugreifen kann. Die Segmentierung beschreibt hierbei die Extraktion einzelner Gefäße und deren Darstellung aus einer Aufnahme. Vordergründig werden mit diesen Applikationen Stenosen, also Gefäßverschlüsse und Gefäßerweiterungen, sogenannte Aneurysmen, diagnostiziert (Brant & Helms, 2012).

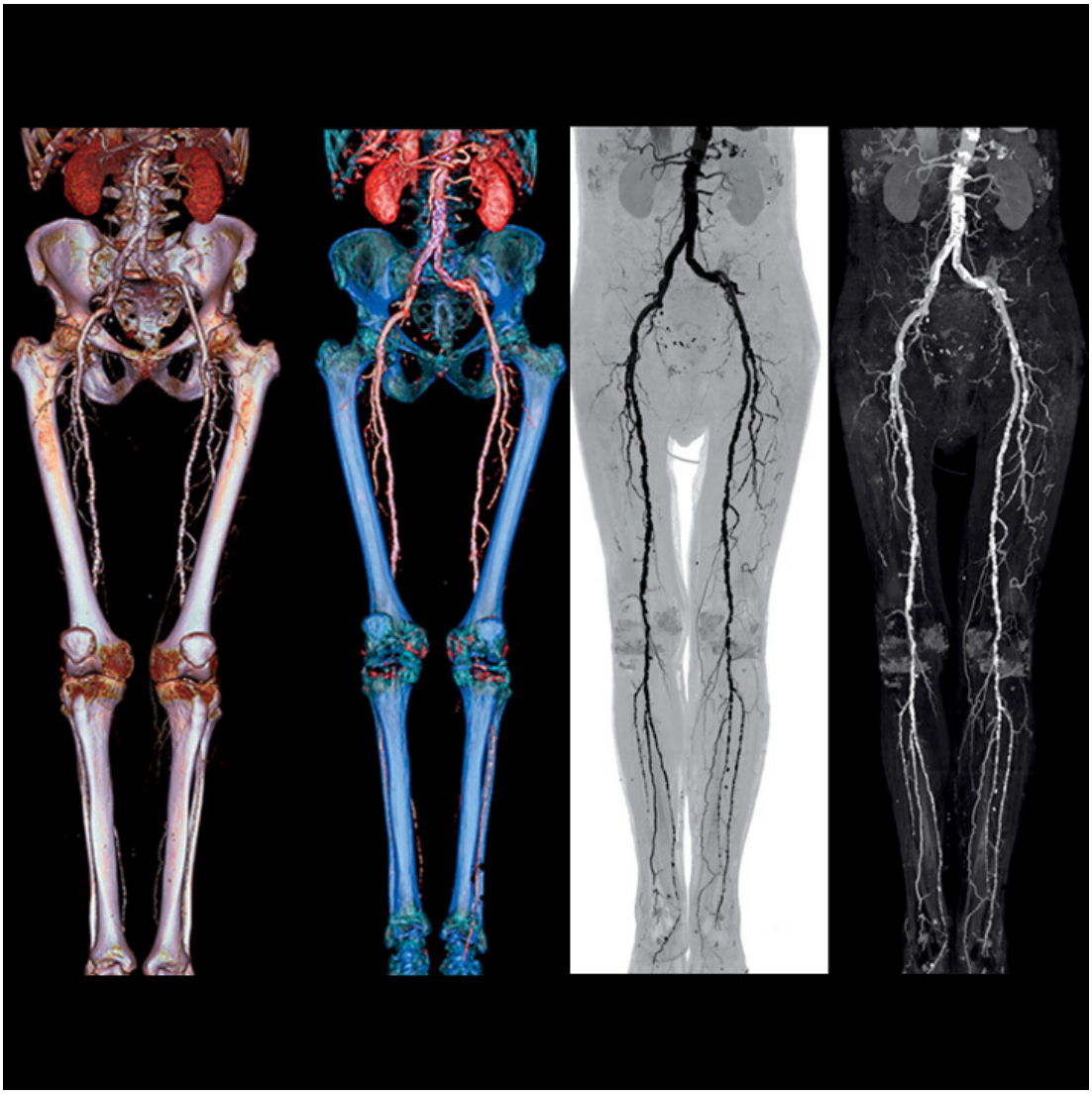

Abbildung 4.1: Vascular Imaging - Darstellung von Blutgefäßen – Angiography (Radiology of Stauferklinikum, 2014)

4.4 Onkologische Befundung

Mit Hilfe der Oncology Applikation werden primär Segmentierungen und Evaluierungen von Läsionen, also krankhaften Veränderungen, der Lunge, Leber, der Lymphknoten und anderen Organen durchgeführt (Siemens Healthcare AG, 2014b). Der Nutzer kann mit dieser Applikation Befunde erstellen, die auf sogenannten Raumforderungen beruhen. Raumforderungen sind beispielsweise gutartige und bösartige Tumore oder Nekrosen. Neben der Erstbefundung können diagnostizierte Läsionen über die Zeit verfolgt werden, um somit den Erfolg der Therapie zu evaluieren und gegebenenfalls anzupassen.

5 Analyse der qualitativen Anwenderdaten

In diesem Kapitel gehen wir neben der Datensammlung auf die Analyse der gewonnenen Daten ein. Mit diesem Kapitel wollen wir eine nachvollziehbare Ausführung der Analyse ermöglichen. So sollen spezifische Kategorien und Codes, die durch die Analyse ermittelt wurden, beispielhaft dargestellt werden, um zum einen die Analyse selbst bildhaft zu erläutern und zum anderen ein Verständnis für die Domäne und deren Zusammenhänge zu ermöglichen. Zunächst wird das Studiendesign zur Analyse näher erläutert, um anschließend auf die erfassten Daten einzugehen. Im dritten Abschnitt wird der Schwerpunkt auf die Analyse der Rohdaten gesetzt.

5.1 Studiendesign

Ein wesentlicher Schritt für die Gewinnung der Daten war die Erstellung einer zentralen Frage, die dem Anwender gestellt wurde (Eisenhardt, 1989; Yin, 2010). In Absprache mit Siemensmitarbeitern und ausgehend von der Domäne wurde dabei folgende Frage entwickelt:

> „Wie gehen die Endanwender bei der Erstellung einer Diagnose, unter der Verwendung einer bestimmten Applikation vor und warum ist dieses Vorgehen für sie notwendig?“

Ausgehend von dieser Frage wurden semi-strukturierte Interviews durchgeführt und durch theoretisches Sampling entsprechende Interviewpartner ausgewählt. Für das Sampling wurden Interviewpartner und Siemensmitarbeiter nach Empfehlungen für weitere Interviewpartner befragt, die bestimmte Gebiete, die in der Analyse aufgetaucht sind, näher erläutern können. Einschränkungen bei dem theoretischen Sampling mussten durch den vorgegebenen zeitlichen Rahmen der

Tabelle 5.1: Punktesystem

Punkte	Beschreibung
+	Grundlegende Kenntnisse
++	Fortgeschrittene Kenntnisse
+++	Starkes Domänenwissen, durch entsprechenden Kundenkontakt oder selbst Endanwender

Interviewpartner hingenommen werden und Interviewpartner wurden dann nach ihrer zeitlichen Verfügbarkeit interviewt. Bei der Interviewdurchführung wurde die zentrale Frage in oben genannter Art und Weise gestellt. Weitere Fragen innerhalb der Interviews wurden dem Kontext entsprechend gestellt. Somit konnten wir während der Interviews Themen vertiefen oder aus vorherigen Interviews aufgreifen. Entsprechend der guten fachlichen Praxis der Forschung mit Grounded Theory, wurde nicht auf bereits existierende Theorien und Ausführungen zu dieser Domäne eingegangen (Yin, 2010, 2014).

5.2 Daten

Im Fachbereich der Onkologie wurden drei Interviews und im Bereich der koronaren und vaskulären Analyse fünf Interviews durchgeführt. Weiterhin zeigt Tabelle 5.2 die Rollen der Interviewpartner sowie eine Einschätzung ihres individuellen Domänenwissens. Zusätzlich verwendeten wir bei unserer Analyse eine textuelle Beschreibung der typischen Abläufe bei der Durchführung onkologischer Befundungen, die von der Siemens Healthcare AG bereitgestellt wurden. Durch diese unterschiedlichen Quellen gewinnt auch die Triangulation in dieser Analyse an Bedeutung. Jedoch liegt der Schwerpunkt der Datengewinnung auf den Interviews. Die Interviewpartner sind dabei Siemensangestellte, deren Bezug zum Anwender jeweils unterschiedlich ausgeprägt ist, abhängig vom Kontakt zum Anwender. Zur Einordnung des Domänenwissens wurden die Interviewpartner befragt, inwieweit sie sich unter dem Gesichtspunkt der Forschungsfrage als Domänenexperten verstehen. Ihre Aussagen wurden zu einem besseren Verständnis in einem Punktesystem abgebildet (siehe Tabelle 5.1).

Mit Hilfe der Einordnung konnte im weiteren Verlauf eine Unterstützung bei der Entscheidung von Anforderungen getroffen werden, in Bezug darauf, ob diese in die zu entwickelnde Sprache integriert werden sollen. Wie der Tabelle 5.2 zu entnehmen ist, korrelieren die Selbsteinschätzung der Experten über ihr Domänen-

Tabelle 5.2: Interviewpartner und Rollen

Interview	**Rolle des Interviewpartners**	**Domänenapplikation**	**Einschätzung des Domänenwissens**
Interview1-27-05	Projektleiter	Coronary/ Vascular	++
Interview2-10-065	Test Engineer (manuelle Tests), vormals MTA	Coronary/ Vascular	+++
Interview3-11-06	Architekt	Coronary/ Vascular	++
Interview4-23-06	Produkttraining für Ärzte, MTAs	Coronary/ Vascular	+++
Interview5-26-06	Clinical Coach	Coronary/ Vascular	++
Interview6-16-07	Projektleiter	Oncology	+
Interview7-22-07	Test Engineer	Oncology	++
Interview8-31-07	Produkt Coach	Oncology	+++

wissen mit deren Kontakt zu den Anwendern und dem Wissen, zu den Applikationen. Dabei kann auch innerhalb einer Rolle, der Kontakt zu den Anwendern unterschiedlich ausgeprägt sein. So ist bei Produkt Coaches, die die Anforderungen der Anwender ermitteln und bei Produkttrainern ein ausgeprägter Kontakt zu Kunden vorhanden, der sich in deren Selbsteinschätzung widerspiegelt. Bei allen anderen Rollen hängt der Kontakt zu den Anwendern stark von den jeweiligen Tätigkeiten ab und kann somit nicht direkt spezifiziert werden.

Die geführten Interviews dauerten zwischen 30 und 90 Minuten und wurden persönlich in Deutsch durchgeführt. Die Transkription der Daten wurde persönlich vom Autor durchgeführt, was eine frühe Einarbeitung und Erschließung der Domäne ermöglichte (Strauss & Corbin, 1998). Weiterhin legten wir Wert auf die parallele Ausführung der Datenanalyse und der Datensammlung, um die Fragen der Interviews entsprechend anpassen zu können. Ursprünglich war eine Trennung der Endanwender- und Architekteninterviews vorgesehen. Zum einen konnte im Verlauf der Datensammlung festgestellt werden, dass nur wenige qualitative Unterschiede zwischen den erfassten Daten bestehen. Primär konnten wir quantitative Abweichungen feststellen, die auf der Nähe zu den Endanwendern beruhen. Zum anderen ist die Trennung der Interviews nicht mehr notwendig, da aus

Architektensicht keine DSL erstellt wird und somit keine Vergleichsmöglichkeit besteht.

5.3 Datenanalyse

Die Analyse der zuvor beschriebenen Daten wurde durch einen iterativen Prozess ausgeführt, indem neu zu analysierende Daten dem konstanten Vergleich mit bereits analysierten Daten unterzogen wurden. Die Auswertung und Gewinnung der Daten wurden mit einer explorativen Fallstudie in Verbindung mit Grounded Theory durchgeführt. Der Prozess der Analyse folgt dabei dem Ansatz von Strauss und Corbin (1998), den wir in Kapitel 2 darstellten. Eine vorbereitende Analyse wurde nicht durchgeführt, jedoch existierte bereits eine mögliche Unterteilung der Domäne in vaskuläre, koronare und onkologische Applikationen, die im Verlauf der Analyse bestätigt wurde. Als Software zur Durchführung der Analyse wurde MAXQDA[1] verwendet. Neben der Kodierung der Daten wurden kontinuierlich Memos verfasst und weiter verfeinert. Diese Memos dienen auf der einen Seite dem Verständnis der entstandenen Codes, zum anderen ermöglichen sie die Beschreibung des Zusammenhangs der verschiedenen Codes. Während der Analyse haben wir insbesondere *In-Vivo-Codes* (Glaser & Strauss, 1967) erstellt. Solche Codes entsprechen entweder direkt der Begrifflichkeit der Interviewten oder entlehnen sich aus diesen Begrifflichkeiten (Glaser & Strauss, 1967).

Durch die Analyse der Daten ergab sich auch die Einordnung der zu untersuchenden Domäne in ihren Kontext. Das Aktivitätsdiagramm (siehe Abb. 5.1) stellt einen typischen Workflow zur Befundung dar. So werden zunächst alle notwendigen Patientendaten erfasst und durch entsprechende klinische Aufnahmen unterstützt. Die Auswertung der klinischen Aufnahmen stellt das Untersuchungsgebiet in dieser Arbeit dar und kann somit als die Domäne bezeichnet werden. Eine sogenannte Interventionsplanung schließt sich an die Auswertung der Daten an und beschreibt eine Intervention beim Patienten mit konservativen oder operativen Methoden (Brant & Helms, 2012), die jedoch nicht Gegenstand dieser Arbeit sind.

Für die Analyse der Domäne in dieser Arbeit stellen die Codes, die mit MAXQDA ermittelt wurden, zentrale Anforderungen der Nutzer dar. So entlehnt sich die Benennung der entstanden Codes direkt aus dem qualitativen Datenmaterial, um zu gewährleisten, dass die zu entwickelnde Sprache auf der eigentlichen Sprache der

[1] http://www.maxqda.com/

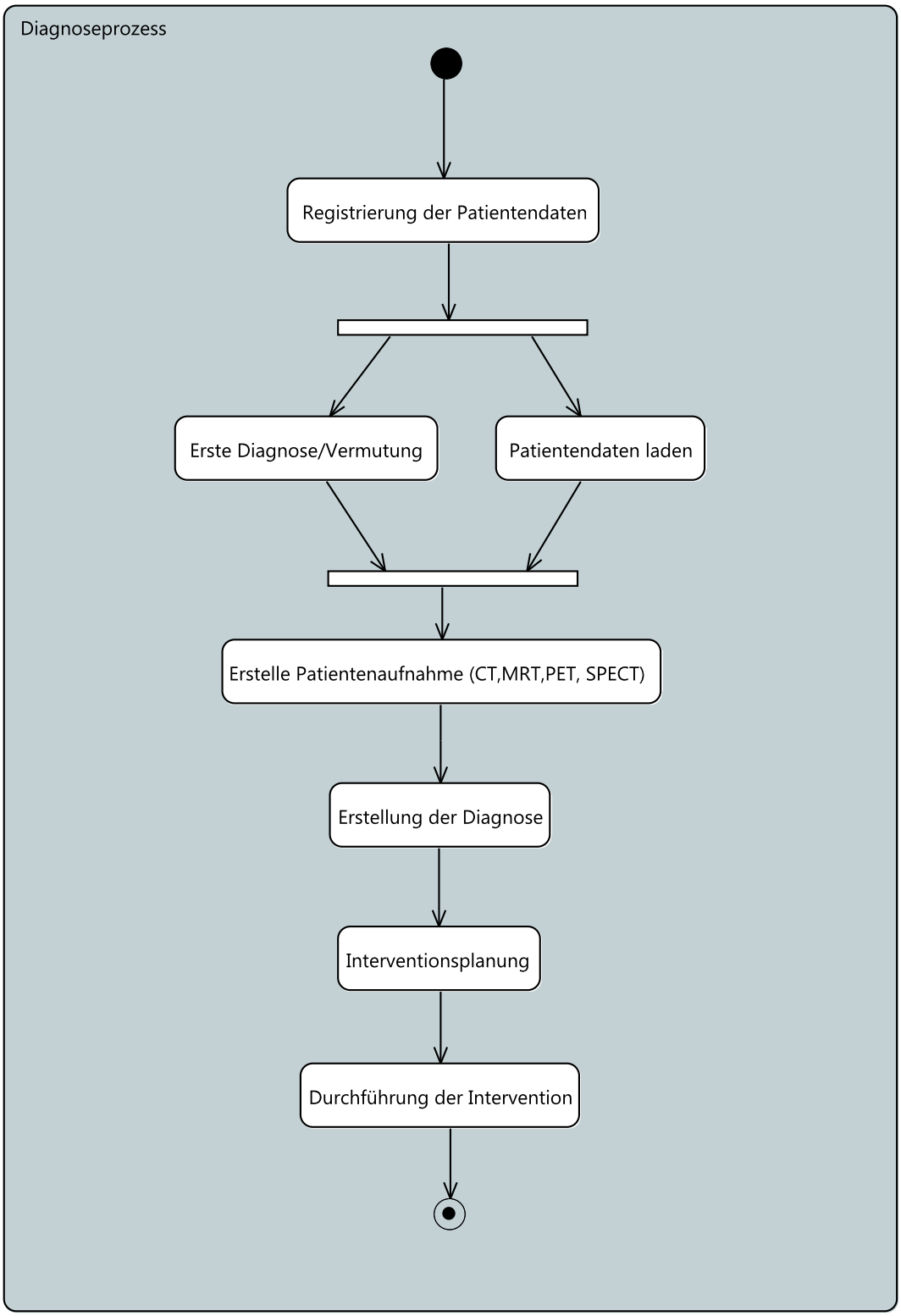

Abbildung 5.1: Diagnoseprozess

Domänenexperten basiert. Obwohl die Interviews selbst in Deutsch durchgeführt wurden, wurden zentrale Elemente der Domäne in Englisch ausgeführt, so dass auch die Kodierung ausschließlich in Englisch erfolgte. Grundlegend konnte festgestellt werden, dass ein diagnostischer Workflow dabei Elemente einer einzelnen Applikation beinhalten kann, jedoch auch eine Kombination von mehreren Applikationen möglich ist. Dieser Workflow stellt somit die zentrale Komponente der zu untersuchenden Domäne dar und verknüpft alle weiteren Konzepte (siehe Abb. 5.2). Die ermittelten Konzepte der Analyse lassen sich in Funktionen und Eigenschaften unterteilen. Funktionen stellen dabei Schritte dar, die die Anwender nutzen, um etwas zu untersuchen. Eigenschaften von Funktionen oder des Layouts werden von den Anwendern genutzt, um die Diagnose entsprechend ihren Vorstellungen durchführen zu können.

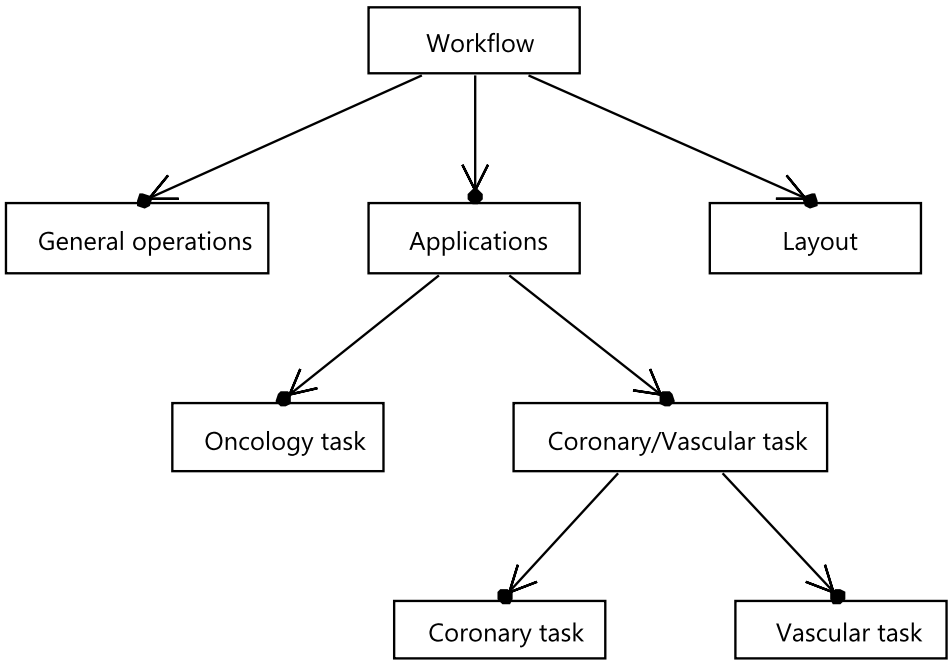

Abbildung 5.2: Domänenübersicht

5.3.1 Hauptapplikationen

Durch die Analyse konnte eine Einteilung der drei bereits erwähnten Applikationen bestätigt werden. So konnten jeweils spezifische Anforderungen für vaskuläre, koronare und onkologische Applikationen festgestellt werden, die sich durch spezielle Eigenschaften und Funktionen unterscheiden. Wie im Kapitel 4 beschrieben,

konnte durch die Analyse eine starke Ähnlichkeit der vaskulären und der koronaren Applikation gezeigt werden. So unterscheiden sie sich insbesondere beim sogenannten Preprocessing, welches spezifische Vorverarbeitungen der medizinischen Bilddaten darstellt. Dies spiegelt sich zum Beispiel durch die unterschiedliche Segmentierung von Gefäßen wieder. Während in der koronaren Applikation ausschließlich Herzkranzgefäße erkannt werden, ermöglicht das Preprocessing in der vaskulären Applikation die Erkennung von Gefäßen im gesamten Körper. Die so erkannten Gefäße können dem Anwender sofort präsentiert werden und er kann nach einer Verifikation sofort mit der Erstellung einer Befundung beginnen. Für onkologische Diagnosen ist jedoch eine andere Vorgehensweise notwendig, was durch die Analyse widergespiegelt werden konnte. Ausgehend von dieser Erkenntnis wird veranschaulicht, dass spezifische Funktionen und Eigenschaften eine Unterstützung bei onkologischen Fragestellungen bieten.

5.3.2 Generelle Eigenschaften

Neben den Eigenschaften der spezifischen Applikationen konnten Eigenschaften extrahiert werden, die alle Applikationen betreffen. Solche Eigenschaften sind zum einen *generelle Operationen* und zum anderen das *Layout*. In der Kategorie generelle Operationen befinden sich Operationen und Funktionen, die in allen Applikationen getätigt werden. Ein sehr häufig genanntes Beispiel dafür ist das Messen einer Distanz, was in den meisten der Interviews benannt wurde. Neben dem Messen einer Distanz wurden weitere Eigenschaften der Anwender benannt und konnten in den meisten Interviews festgestellt werden. Mit der Kategorie Layout wird die Darstellung der zu untersuchenden Aufnahmen und deren Eigenschaften beschrieben. So definiert das Layout eine Zusammensetzung aus verschiedenen Segmenten, die die Aufnahmen mit unterschiedlichen Filtern und Eigenschaften, wie der Ausrichtung oder dem Zoom darstellen. Einem Segment ist dabei jeweils zu einer bestimmten Zeit genau ein Filter mit spezifischen Eigenschaften zugeordnet, die einem Anwender die Möglichkeit geben, sich die Aufnahmen seinen Vorstellungen entsprechend darstellen zu lassen.

5.4 Zusammenfassung

Dieses Kapitel stellte die qualitative Datenanalyse der zu untersuchenden Daten dar. Als erstes wurde auf das Design der Studie eingegangen. Neben den üblichen Grundlagen zur Durchführung einer qualitativen Datenanalyse wurde

die zentrale Frage für die Interviews formuliert. Anschließend wurde auf die zu untersuchenden Daten eingegangen, die sich primär auf Interviews konzentrierten, jedoch durch bereitgestellte Dokumente auch Triangulation ermöglichen. So konnte durch die Analyse festgestellt werden, dass zwischen den Interviews weniger thematische Unterschiede, als vielmehr Unterschiede im Inhalt zu bemerken waren. Je mehr Kontakt ein Interviewpartner mit dem Endanwender hatte, oder selbst einer war, umso mehr konnte im Interview zur zentralen Frage beigetragen werden. Die Durchführung der Analyse fand mit MAXQDA statt und ermöglichte zum einen den kontextuellen Einblick in die Domäne, genauer wie sich der diagnostische Workflow einbettet und zum anderen die Einteilung der Domäne in die drei beschriebenen Applikationen. Neben den beschrieben Applikationen konnten weitere Eigenschaften ermittelt werden, die generellen Operationen und das Layout.

6 Ableitung des Domänenmodells

Um die analysierten Daten in eine domänenspezifische Sprache zu übertragen, ist es notwendig ein Modell der Domäne zu erstellen, das die Zusammenhänge darstellt. Aus diesem Grund werden wir in diesem Kapitel eine Möglichkeit der Umsetzung eines Domänenmodells aufzeigen. Zunächst wird ein Featuremodell erstellt, das alle notwendigen Features der Domäne enthält. Für die Implementierung der Sprache wird zusätzlich ein Klassendiagramm verwendet, das die Darstellung struktureller Zusammenhänge ermöglicht. Abschließend wird auf Sprachmerkmale der Domäne eingegangen.

6.1 Featuremodell

Die Attribute der Domäne lassen sich unter anderem in Funktionen und Eigenschaften unterteilen. Diese Attribute wurden von den Domänenexperten in Interviews genannt und beschrieben. Eine Möglichkeit unterschiedliche Bestandteile eines Systems, sogenannte Features, darzustellen wird durch sogenannte Featuremodelle ermöglicht, wie sie in Kapitel 5 beschrieben wurden. Ein Einsatzszenario für Featuremodelle ist die Entwicklung von Softwareproduktlinien, denn ihre kompakte Darstellung macht sie im Kontext dieser Arbeit zu einem sinnvollen Werkzeug (Kang et al., 1990; Mernik et al., 2005). Diese Features stellen Aspekte, Eigenschaften oder Charakteristiken eines Systems oder einer Domäne dar, die von den Anwendern wahrgenommen und benötigt werden (Kang et al., 1990). Ein Featuremodell erlaubt somit die hierarchische Komposition aller Features, die für die Endanwender der Domäne relevant sind (Kang et al., 1990). Neben den Features können zusätzlich deren Abhängigkeiten voneinander dargestellt werden. Die Entscheidung fiel zunächst auf das Featuremodell, da es geeignet ist, dass Codesystem der Analyse abzubilden und somit eine iterative Weiterentwicklung

möglich war. Die Aufteilung der Features in notwendige und optionale Features bezieht sich auf die spezifischen Produkte eine Softwareproduktlinie, während sie sich speziell in dieser Arbeit auf die Einteilung der spezifischen Instanzen dieser DSL bezieht. Eine Instanz stellt ein Ablauf dar, der durch die Anwender erstellt werden kann. Ansonsten folgen wir der Notation der Featuremodelle.

Durch die Analyse konnte eine hierarchische Komposition von Codes erstellt werden, die als Features verstanden werden können. Die entstandenen Codes stellen eine Beschreibung von dem dar, was durch die Anwender getan wird. Somit war es grundlegend möglich, die Codes der Analyse in ihrer hierarchischen Komposition direkt zu übernehmen. Für das Featuremodell wurden die Codes so übertragen, dass sie eine bestimmte Tätigkeit beschreiben, die ein Endanwender typischerweise ausführt. Ein Großteil der Features, bezüglich des diagnostischen Workflows, konnte durch ein direktes Mapping aus dem Codesystem übertragen werden. Wenige Codes, die in den Interviews genannt oder beschrieben und nach Rücksprache mit den Domänenexperten als nicht praktikabel oder notwendig angesehen wurden, sind nicht in das Domänenmodell übernommen wurden. Hierfür wurde ausschließlich mit Experten Rücksprache gehalten, die ein großes Domänenwissen besitzen, wie dies in Kapitel 5.2 beschrieben wurde. Neben den direkt übernommenen Konzepten und Codes, leiten sich einige Bezeichnungen von Features aus den Memos ab, die während der Analyse entstanden sind.

Der *Diagnosis Workflow* unterteilt sich ebenso wie bei der Analyse in *Layout*, *General operations*, *Oncology task* und *Vascular/Coronary task*. Diese Komposition erstreckt sich dabei parallel zum Baum des Codesystems. Somit ist eine einfache Übertragung der Konzepte aus der Analyse gegeben. Diese einfache Übertragbarkeit führt zum einen dazu, dass Features der Domäne direkt aus der Analyse entlehnt sind und somit die Domänensprache direkt aufgreifen. Zum anderen wird eine direkte *Traceability* der Features gewährleistet (Rupp et al., 2009). Das bedeutet, dass ein Feature in die Daten zurückverfolgt werden kann und in der Analyse verankert ist. Neben der Traceability wird durch dieses Vorgehen die Konformität der Domäne sichergestellt. Das Vorgehen beim Übertragen der Konzepte und Codes wird beispielhaft an den generellen Operationen in Tabelle 6.1 dargestellt.

Der größte Anteil der Features stellt Funktionen dar, die Anwender zur Befundung nutzen. Einige Funktionen besitzen Eigenschaften, die von den Endanwendern nach ihren Bedürfnissen angepasst werden können. Der Großteil der Features in Tabelle 6.1 stellt Funktionen dar, die von den Endanwendern ausgewählt

Tabelle 6.1: Generelle Operationen

Codes des Codesystems	**Feature im Featuremodell**
blowup	Perform segment blowup
panning	-
Windowing	Align windowing
Align segments /bookmark	Align segment
Report	Show and edit report
Rotate	Rotate view
Dataset	-
scrolling/ reading	Scroll through segments
mark with arrowtool	Mark with arrow
PACS	-
Results/ Findings	-
finish evaluating/ reading	Finish evaluation
Printing	Print image
create segment snapshots	Create snapshot
ranges	*Create series of images (abstract)*
marker	Mark point of interest
mark ROIs	Mark ROI
measure distance	Measure distance
measure angle	Measure angle
change CT preset	Change CT preset

werden können und deren Bezeichnung sich aus dem Codesystem ableiten. Eine Ausnahme stellt hier beispielsweise der Code *ranges* dar. Dieser Code beschreibt ein Tool, was von syngo.via angeboten wird und es ermöglicht, Bildserien zu erstellen. Diese Verallgemeinerung konnte aus den Interviews der Anwender entnommen werden und führt zur produktunabhängigen Benennung *Create series of images*. Diese Benennung beschreibt eine Tätigkeit, die von den Anwendern ausgeführt werden kann. Die Übertragung der Einträge der generellen Operationen aus dem Codesystem wird zusätzlich mit dem Ausschnitt aus dem Featuremodell in Abbildung 6.1 dargestellt.

Aufgrund der engen Verzahnung des Codesystems mit dem Featuremodell, konnte eine kontinuierliche Entwicklung des Featuremodell parallel zur Analyse durchgeführt werden. Somit basieren alle Features direkt auf Konzepten, die aus der Analyse hervor gegangenen sind und damit auf der Sprache der Domäne. Neben den Features wird die Möglichkeit der Erstellung von Constraints gegeben. Beispielsweise können bestimmte Segmentfilter nur in bestimmten Applikationen verwendet werden. Wie zum Beispiel der CPR Filter (Curved Planar Reformation (Kanitsar, Wegenkittl, Fleischmann & Groller, 2003)), der ausschließlich in der Coronary- und Vaskular-Applikationen Verwendung findet. Weitere Constraints werden im Featuremodell dargestellt und beschreiben Einschränkungen, die in der Umsetzung der Sprache Beachtung finden müssen. Das erstellte Featuremodell ist im Anhang A dargestellt.

6.2 Klassenmodell

Neben dem dargestellten Featuremodell wurde zur Darstellung struktureller Eigenschaften der Domäne ein Klassendiagramm gewählt (siehe Anhang B). Mit Hilfe eines Klassendiagramms wird die Definition und Erfassung von Wissen ermöglicht, welches für die spätere Implementierung notwendig ist (Balzert & Balzert, 2009). Neben den Primärdaten der Domäne enthält ein Klassendiagramm, kontextuelles Wissen, wie beispielsweise die Struktur. Dieses Wissen kann für die spätere Weiterentwicklung nützlich sein. Ebenso wie das Featuremodell basiert das Klassendiagramm direkt auf dem Codesystem, das durch die Analyse gewonnen werden konnte. Das Klassendiagramm ermöglicht die Ergänzung des Featuremodells um Zusammenhänge, Kardinalitäten und inhaltliche Dinge, die ansonsten nicht dargestellt werden können.

Mit Hilfe des Klassendiagramms wird der Aufbau eines *Diagnosis workflow* dargestellt. Dieser Workflow beinhaltet ein oder mehrere *Specific tasks*, die zum einen

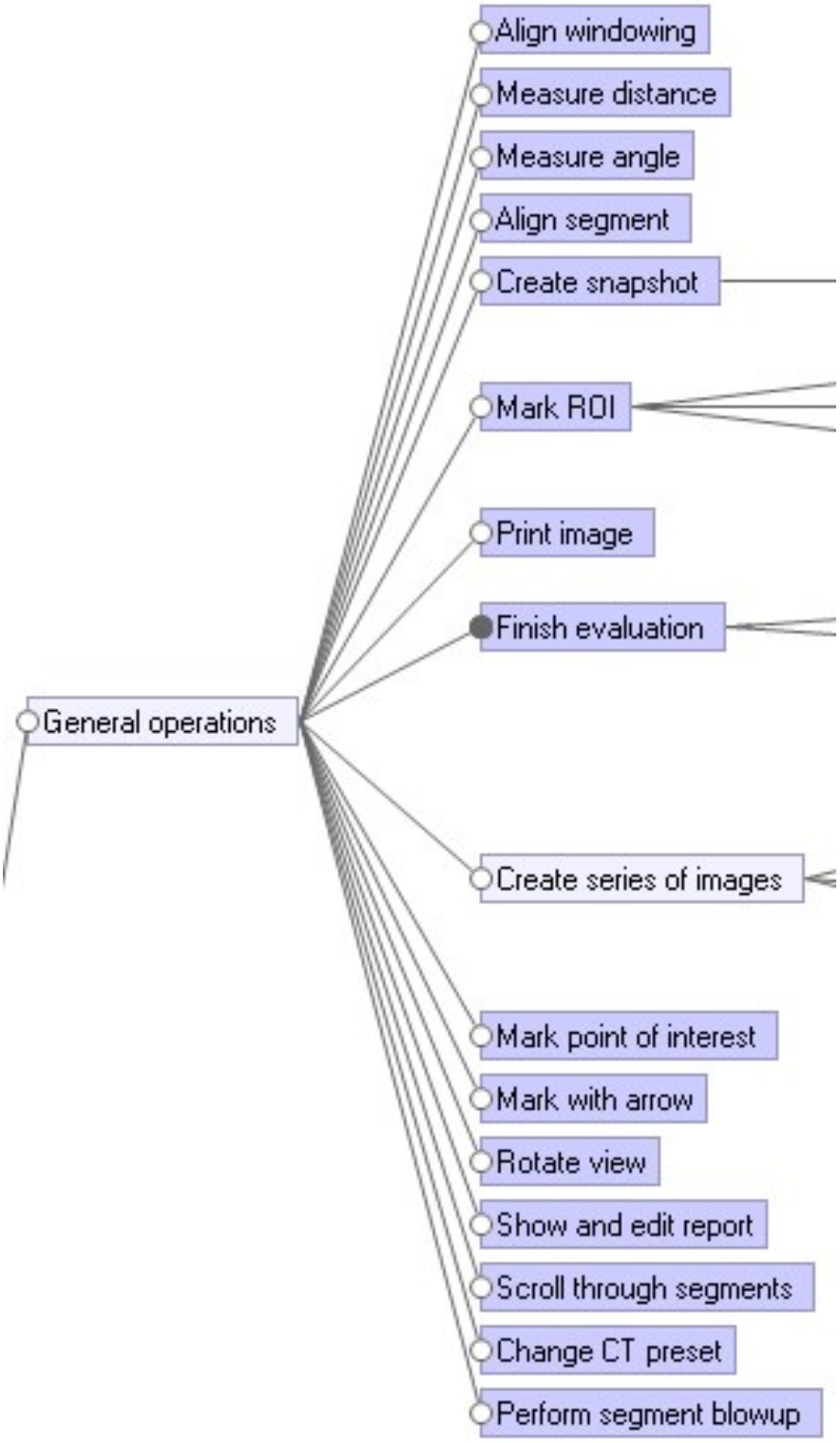

Abbildung 6.1: Ausschnitt Featuremodell der generellen Operationen

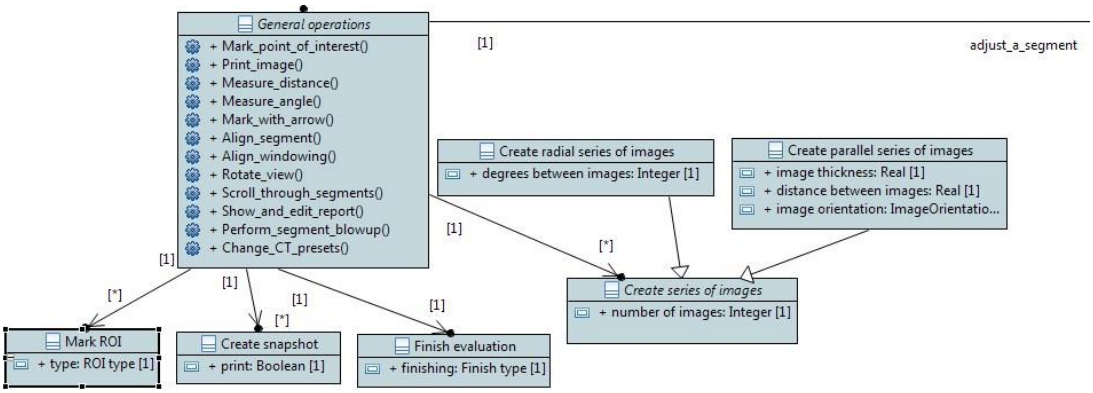

Abbildung 6.2: Klassendiagramm generelle Operationen

General operations und zum anderen Operationen der drei bekannten Applikationen verwenden. In einer Applikation können verschiedene Operationen ausgewählt werden, die im Klassendiagramm aufgrund der Übersichtlichkeit nicht näher dargestellt werden. Neben den Tasks ist dem *Diagnosis Workflow* ein *Layout* zugeordnet, das über die generellen Operationen vom Endanwender angepasst werden kann. Dieses Diagramm basiert, ebenso wie das Featuremodell, auf dem Codesystem der Analyse.

Mit dem Klassendiagramm wird eine Differenzierung der Features in Nutzerrelevante Operationen und deren Eigenschaften ermöglicht. Beispielhaft sei dies an der Klasse *Mark ROI* erklärt. Diese Klasse stellt eine generelle Operation dar, die in allen Applikationen ausgewählt werden kann, um interessante Regionen zu markieren. Diese Markierung kann auf unterschiedliche Weisen durchgeführt werden, weshalb ein Attribut *type* hinzugefügt wurde. Mit der Enumeration *ROI type* kann der Anwender auswählen, wie er etwas markieren möchte. Benutzerrelevante Operationen oder Funktionen werden entweder als eigene Klasse dargestellt oder sind als Funktion einer der Applikationsklassen oder den generellen Operationen zugeordnet. Zur besseren Übersicht und bei Generalisierungs-Spezialisierungs-Beziehungen haben wir uns für die Erstellung eigener Klassen von Operationen entschieden. Die Attribute wurden entweder als Übergabeparameter von Funktionen deklariert oder bei eigenen Klassen entsprechend als Attribut aufgeführt. Dieses Vorgehen ist auf alle anderen Operationen und deren Attribute übertragbar. Die Differenzierung der Features ergibt sich aus der Analyse, insbesondere deren Memos. Beispielhaft sei auf den Ausschnitt des Klassendiagramms verwiesen, der die generellen Operationen darstellt (siehe Abb. 6.2).

6.3 Sprachliche Merkmale

Durch die Ableitung des Featuremodells und des Klassendiagramms aus der Interviewanalyse, ergeben sich deren Bestandteile aus der Sprache der Domänenexperten. Sie stellt somit eine imperative Fachsprache dar, die es Anwendern ermöglicht diagnostische Abläufe zu beschreiben. Dementsprechend wurden alle Elemente des Featuremodells und Klassendiagramms ähnlich der Codes aus der Analyse benannt, dies erzeugt eine starke Verknüpfung zwischen der Analyse und der zu entwickelnden DSL. Aus diesem Grund sind sprachliche Begriffe den Funktionen eindeutig zuzuordnen und geben damit direkt Auskunft über diese Funktionen. Für die Beschreibung der Operationen, die von Endanwendern ausgewählt werden können, wurde für Verben der Imperativ gewählt. Dies verdeutlicht die Aufforderung, die der Nutzer ausdrücken möchte. Dem Anwender wird eine beschränkte Kommunikation mit der Sprache ermöglicht, die sich auf vorgegebene Sprachbefehle und Ausdrücke der Domäne eingrenzt. Durch die Beschreibung der Abläufe erzeugt die Sprache eine festgelegte Chronologie. Die Befehle, die der Anwender beschrieben hat, werden in der festgelegten Reihenfolge ausgeführt. Allen Befehlen gemein, ist die prägnante Ausdrucksweise, damit für die Anwender eine einfache Verständlichkeit ermöglicht wird. Weiterhin konnten bestimmte Wörter als spezifische Sprachmerkmale identifiziert werden, die wiederholt bei bestimmten Features auftauchen. Dazu zählt das Messen bestimmter Eigenschaften, wie beispielsweise *measure distance*, welches durch das Wort *measure* dargestellt wird. Weitere zentrale Merkmale sind *mark*, um bestimmte Bildbereiche unterschiedlich zu markieren, *show*, damit etwas nach den Vorstellungen des Anwenders angezeigt wird oder *create*, um ein Ergebnis zu erzeugen, was für die Befundung relevant ist. Zusätzlich relevant ist der Begriff *align*, da mit dieser Anweisung alle Segmente ausgerichtet werden, was durch das Klassendiagramm verdeutlicht wird. Ebenso sei hier auf die produktunabhängige Benennung der Elemente des Domänenmodells hingewiesen, um zu gewährleisten, dass ein Element direkt beschreibt, was damit getan wird.

6.4 Zusammenfassung

In diesem Kapitel konnten wir beispielhaft die Überführung des Codesystems aus der Analyse in das Featuremodell, anhand der generellen Operationen, zeigen. Die einfache Übertragung der Codes in Features wurde dargestellt. Durch die einfache Übertragung ist eine gute Traceability gewährleistet, denn es besteht eine starke

Verknüpfung zwischen den Features im Featuremodell auf der einen Seite und den Elementen des Codesystems auf der anderen Seite. Für die Darstellung von Zusammenhängen oder Kardinalitäten zwischen den Funktionen und damit von Features wurde ein Klassendiagramm gewählt. Die Kombination des Featuremodells und Klassendiagramms stellt das eigentliche Domänenmodell dar und bietet die Möglichkeit der Implementierung einer Sprache. Weiterhin wurde in diesem Kapitel auf relevante Merkmale der Sprache eingegangen, die für die Anwender von Bedeutung sind.

7 Erstellung der DSL

In diesem Kapitel gehen wir auf die Erstellung der DSL mittels JetBrains MPS ein. Zunächst stellen wir Grundlagen der Sprecherstellung mit MPS dar. Darauf folgt die Überführung des Domänenmodells in die Sprache. Anschließend wird der Schwerpunkt auf die prototypische Umsetzung in syngo.via gesetzt.

7.1 Grundlagen MPS

Für die Überführung ist es notwendig, zunächst Grundlagen zur Erstellung einer Sprache mit MPS zu umreißen.

Wie im Abschnitt 3.2.2 beschrieben, setzt MPS auf die Verwendung von projektionalen Editoren. Zunächst wird das Schema der DSL erstellt, was auch als Metamodell oder abstrakte Syntax bezeichnet wird, bei MPS geschieht dies in Form von Sprachkonzepten. Die Instanz der abstrakten Syntax, in Form eines Programmes wird als AST bezeichnet. In einem zweiten Schritt wird die konkrete Syntax erstellt in Form eines projektionalen Editors. Bei diesem Ansatz wird der AST direkt durch Aktionen im Editor erstellt und die konkrete Syntax wird durch Projektionsregeln dargestellt. Der Anwender interagiert somit über die Projektionsregeln des Editors mit dem AST. Aus diesem Grund ist es möglich, konkrete Syntax in Form von Text, Symbolen, Tabellen oder Grafiken darzustellen. (Voelter, Benz et al., 2013; Jetbrains, 2014a)

Grundlegende Elemente bei der Erstellung der Struktur einer Sprache sind die sogenannten *Concepts*, die im nachfolgenden als Konzepte bezeichnet werden. Die Instanz eines Konzeptes ist ein Knoten des AST. Ein Konzept definiert Struktur, Syntax, Typsystem und die Semantik seiner Instanzen. Die Summe aller Konzepte und deren Beziehungen untereinander ergeben die Struktur einer Sprache. Um allgemeine Eigenschaften für Konzepte zu definieren, existiert die Möglichkeit

sogenannter *Concept Interfaces*. Ein Konzept kann ein oder mehrere Konzept Interfaces implementieren. Diese Analogie zu anderen Programmiersprachen setzt sich bei der Erweiterung von Konzepten fort. Weiterhin besteht die Möglichkeit, die Instanz eines Konzepts im AST als *Root-Knoten* zu definieren. Für Konzepte können Attribute, sogenannte *Properties*, festgelegt werden, die einen bestimmten Typ besitzen wie primitive, enumerations oder selbstdefinierte Typen. Außerdem können *Children* festgelegt werden. Sie definieren Aggregationsbeziehungen zwischen dem Konzept und seinen möglichen Bestandteilen, inklusive deren Kardinalität. Mit diesen wird die hierarchische Struktur des AST festgelegt. (Jetbrains, 2014b; Campagne, 2014)

Für jedes erstellte Konzept kann ein Editor erstellt werden, der die konkrete Syntax darstellt. Er ermöglicht, die Anzeige und Bearbeitung von AST-Knoten. Weiterhin können *Constraints* festgelegt werden, die die Beziehungen der Konzepte ergänzen oder einschränken. Um Verhaltensweisen von Konzepten vorzudefinieren, besteht die Möglichkeit der Erstellung von *Behaviors*, damit wird es beispielsweise ermöglicht, Attribute vorzudefinieren. Die Constraints und Behaviors sind als Teil der statischen Semantik zu verstehen. (Campagne, 2014)

7.2 Implementierung der Sprache

Als Grundlage für die Implementierung der DSL in dieser Arbeit verwendeten wir das erstellte Domänenmodell (siehe Kapitel 6). Die ermittelten Features konnten in die beschriebenen Sprachkonzepte von MPS überführt werden. Mit Hilfe des Klassendiagramms wurden unter anderem die Kardinalitäten zwischen den Konzepten und ihren Children bestimmt. Beispielhaft soll das Vorgehen, mittels der generellen Operationen, und den darüber liegenden hierarchischen Features vorgestellt werden, wie dies bei der Erstellung des Domänenmodells in Kapitel 6 durchgeführt wurde.

Durch die Analyse konnte festgestellt werden, dass der *Diagnoseworkflow* aus mindestens einer, aber auch mehreren Applikationen bestehen kann. Aus diesem Grund wurde ein Konzept *DiagnosisWorkflow* angelegt, dessen Instanzen Rootelemente sind. Weiterhin wurden Konzepte für die drei bekannten Applikationen angelegt, der *CoronaryTask*, der *VascularTask* und der *OncologyTask*. Die Benennung der Konzepte orientiert sich hierbei am Domänenmodell. Die Instanzen der Konzepte der Applikationen können hierbei selbst wieder Rootelemente darstellen, da ein Workflow nur aus einer einzelnen Applikation bestehen kann. Ebenso können die Applikationen Children des Konzepts *DiagnosisWorkflow* sein. Da-

mit innerhalb des Konzeptes *DiagnosisWorkflow* die beschriebenen Applikationen ausgewählt werden können, sind die Children dieses Konzeptes vom Typ *TaskInterface*. Dieses *TaskInterface* wird von den drei beschriebenen Konzepten der Applikationen implementiert. Somit wird ermöglicht, dass innerhalb eines Workflows die entsprechenden Tasks ausgewählt werden können. Gleichzeitig stellt das *TaskInterface* eine Notwendigkeit bei der Implementierung dar, existiert jedoch nicht im Domänenmodell. Für die Applikationen wurde implementierungsbedingte Interfaces erstellt, die von dem Konzept *GeneralOperations* implementiert werden. Mit Hilfe dieser Art der Implementierung wird es ermöglicht, dass die generellen Operationen in den entsprechenden Tasks ausgewählt werden können. Damit diese Operationen ausgewählt werden können, implementiert das Konzept *GeneralOperations* das *CoronaryInterface*, *OncologyInterface* und *VascularInterface*. Das Konzept *GeneralOperations* stellt eine Spezialisierung des *BaseConcepts* dar, welches grundlegende Eigenschaften eines Konzeptes mitbringt.

In Tabelle 7.1 wird die Überführung der generellen Operationsfeatures in die entsprechenden Konzepte dargestellt. Eine Besonderheit bei der Implementierung musste beim Feature *Align segment* beachtet werden. Da für die Segmentfilter wie MIP (Maximum intensity projection (Hofer, 2010)) oder VRT (Volumen Rendering Technik (Hofer, 2010)) unterschiedliche Eigenschaften gelten, wurde für jeden Filter ein eigenes Konzept erstellt. So können die Anwender ein Segment mit einem bestimmten Filter justieren. Ansonsten konnte auch hier eine direkte Übertragung der Features in die DSL ermöglicht werden. Damit ist sichergestellt, dass die Elemente der Analyse in der DSL auffindbar sind. Für die Implementierung wurden weitere Konzepte angelegt, wie zum Beispiel das *TaskInterface*, das für den Anwender nach außen nicht sichtbar ist und kein Einfluss auf diese Konzepte genommen werden kann. Weitere Besonderheiten, wie die Erstellung von abstrakten Konzepten, ermöglicht die direkte Übertragung abstrakter Features, wie dies beim Konzept *CreateSeriesOfImages* geschehen ist. Die konkrete Implementierung wird erst durch Konzepte der nächst niedrigeren Hierarchiestufe aus dem Featuremodell übernommen.

Für die konkrete Implementierung eines Konzepts soll im Folgenden das Konzept *MarkROI* (siehe Abb. 7.1) näher erläutert werden. Dieses Konzept ist nicht abstrakt und stellt eine Funktion dar, die vom Anwender ausgewählt werden kann. Da es das Konzept *GeneralOperations* erweitert, kann es in jedem Task ausgewählt werden. Zunächst wurde hier der Alias *Mark a region of interest* gewählt, den der Anwender bei Verwendung angezeigt bekommt. Weiterhin besitzt das Konzept die Eigenschaft *ROIType*. In der Analyse konnte festgestellt werden, dass *ROI* freihändig, mit einem Kreis oder Polygonal, markiert werden

Tabelle 7.1: Überführung der generellen Operationen aus dem Featuremodell in die Implementierung

Feature im Domänenmodell	**Konzept in MPS**
Align windowing	AlignWindowing
Align segment	AlignMIPSegment AlignMIPThinSegment AlignMPRSegment AlignMPRThickSegment AlignVRTSegment AlignVRTThinSegment
Show and edit report	ShowAndEditReport
Rotate view	RotateView
Scroll through segments	ScrollThroughSegments
Mark with arrow	MarkWithAnArrow
Finish evaluation	FinishEvaluation
Print image	PrintImage
Create Snapshot	CreateAnSnapshot
Create series of images (abstract)	CreateSeriesOfImages (abstract)
Mark point of interest	MarkAnPointOfInterest
Mark ROI	MarkROI
Measure distance	MeasureAnDistance
Measure angle	MeaseureAnAngle
Change CT preset	ChangeCTPreset

können. Somit hat der Anwender die Auswahl zwischen drei Typen. Für diesen Zweck wurde ein Enumerationsdatentyp *ROITypes* erzeugt, der dem Endanwender eine Auswahlmöglichkeit zwischen den Typen anbietet. In anderen Konzepten wurden andere Datentypen wie Ganzzahl-, Gleitkommazahl-, oder Stringtypen verwendet. Alle weiteren Konzepte, wurden in ähnlicher Art und Weise umgesetzt.

```
concept MarkROI extends    GeneralOperations
                implements <none>

 instance can be root: false
 alias: Mark a region of interest
 short description: <no short description>

 properties:
 ROIType : ROITypes
```

Abbildung 7.1: Konzept *MarkROI*

Als beispielhafte Darstellung für die Konzepte der drei Tasks, soll das Konzept des *OncologyTask* beleuchtet werden. Die Erstellung der Abläufe innerhalb der Tasks ist immer ähnlich. Das Konzept *OncologyTask* implementiert das *TaskInterface*, damit es im Konzept *DiagnosisWorkflow* zur Verfügung steht. Die Besonderheiten der Implementierung der Applikationen liegen in deren Kinderkonzepten (siehe Abb. 7.2). So besitzt jede Applikation ein spezifisches Preprocessing, das Vorberechnungen für die Diagnose vornimmt. Bei dem Konzept *OncologyTask* ist dies das Konzept*PreprocessingOncology*. Der Anwender entscheidet, ob ein bestimmter Schritt durchgeführt wird oder nicht. Des Weiteren muss beim Abschluss einer Befundung entschieden werden, ob die Daten lokal im System gespeichert oder archiviert werden. Dies wird mit dem Konzept *FinishEvaluation* realisiert. Das *OncologyInterface* wird zum einen durch die *GeneralOperations* implementiert, zum anderen wird es durch die Operationen, die spezifisch für onkologische Diagnosen sind, umgesetzt. Somit hat der Anwender die Auswahl zwischen generellen und onkologischen Operationen. Eine Übersicht über alle Konzepte der DSL befindet sich im Anhang (siehe Anhang C).

Eine Instanz des *OncologyTask* wird in Abbildung 7.3 dargestellt. Der Task erhält einen vom Anwender gewählten Namen für die Eindeutigkeit. Die Preprocessing Optionen und die Beendigung der Evaluierung werden durch das System vorgegeben, wobei der Anwender zwischen den Optionen entscheiden kann. Die beiden

```
children:
oncologyOperations     : OncologyInterface[1..n]
preprocessingOncology  : PreprocessingOncology[1]
finishOncology         : FinishEvaluation[1]
```

Abbildung 7.2: Konzept *OncologyTask*

anderen Einträge *mark a region of interest* und *create finding with:* stellen eine mögliche Auswahl des Anwenders dar und sind Instanzen der Konzepte *MarkROI*, welches vorgestellt wurde, sowie von *CreateFindingWithSegmentation*. Das zweite Konzept stellt eine spezifische Operation für onkologische Befundungen dar, die nur in dieser Task ausgewählt werden kann und beschreibt die Segmentierung einer onkologischen Läsion.

```
Oncology Task ForMasterThesis {
  preprocessing options - activate:
    remove table: true
    arterial enhancement fraction for liver: false
    automated lung segmentation with LungCAD: false

  mark a region of interest: Circle

  create finding with:  general segmentation

  finish the evaluation and: store in local system
}
```

Abbildung 7.3: Instanz *OncologyTask*

7.3 Prototyp

Neben der prototypischen Umsetzung der Sprache, gehen wir in diesem Abschnitt auf eine Möglichkeit zur Verkürzung des Befundungsprozesses ein.

7.3.1 Prototypische Umsetzung

Für die prototypische Verwendung der Sprache ist es notwendig, die Sprache zu übersetzen. In MPS existieren Generatoren, die eine Model-to-Model Transfor-

mation ermöglichen. Mit dieser Transformation werden Instanzen der Sprache, der AST eines spezifischen Programms, in die Zielsprache übertragen. Die sogenannte Base Language von MPS stellt Java dar. Jedoch können weitere Sprachen als Ziel der beschriebenen Transformation mittels eines Generators dienen. Zentrale Elemente der Übersetzung sind die *Mapping Configuration* und die sogenannten *Templates*. Mit Hilfe der Mapping Configuration wird bestimmt, wie die Knoten aus dem AST auf entsprechende Templates übertragen werden. Die Templates stellen Modelle dar, die in der Zielsprache verfasst sind. Innerhalb der Mapping Configuration wird mit Regeln festgelegt, wie die Konzepte in Templates übertragen werden. In dieser Arbeit sind für die Übertragung der Konzepte zwei Klassen von Regeln notwendig. Zum einen die sogenannte *Root mapping rule*, mit deren Hilfe definiert wird, welches Template für einen Inputknoten eines Konzept aufgerufen wird. Als Ergebnis wird ein Rootknoten in der Zielsprache erzeugt. Zum anderen *Reduction rules*, die es ermöglichen, Instanzen von Konzepten mittels der aufgerufenen Templates in Zielknoten zu übersetzen. Sie stellen keine Rootknoten dar. (Jetbrains, 2014b)

In syngo.via existiert ein System, das es ermöglicht, eine kleine Anzahl von Operationen automatisiert durchführen zu lassen. Die in dieser Arbeit vorgestellte DSL bietet eine weitaus größere Anzahl von Operationen, die von den Endanwendern ausgewählt werden können. Der hier verwendete Prototyp baut auf diesem existierenden System auf. Dabei basiert das existierende System auf XML-Protokollen. Für die Erstellung des Generators wurde von der Siemens Healthcare AG die Dokumentation für das sogenannte Rapid Results Protokoll zur Verfügung gestellt. Da dies produktrelevanten Code darstellt, wird in dieser Arbeit in Auszügen auf den Generator eingegangen. Weiterhin wird nicht auf den Zielcode eingegangen, da eine Ausführung nur mit syngo.via möglich ist. MPS bietet, neben der Unterstützung von Java, die Unterstützung für die Übersetzung der Konzepte in XML an. Die im vorherigen Abschnitt beschriebenen Sprachkonzepte wurden mittels XML-Templates in das Zielprotokoll übersetzt. Der *DiagnosisWorkflow* und die drei Tasks wurden in Rootknoten übersetzt, die dementsprechend auch ein XML-Wurzelelement enthalten.

Alle weiteren Konzepte wurden mittels Reduction rules in entsprechende Templates übersetzt. Soweit es möglich war, wurde auf die existierenden Operationen des Rapid Results Protokoll zurückgegriffen. Operationen, die nicht implementiert sind, wurden mit Platzhaltern übersetzt. Somit können einfache Abläufe innerhalb von syngo.via getestet werden, die von den Anwendern mittels der DSL beschrieben wurden. In Abbildung 7.4 wird ein möglicher Ablauf mit MPS definiert. Zunächst sollen die segmentierten Gefäße vom Anwender verifiziert werden,

um anschließend für ein Gefäß Länge, Stenosegrad und Durchmesser zu erfassen. Weiterhin wurde eine Serie von Bildern angelegt, eine Distanz und ein Winkel gemessen. Das Preprocessing wurde auf den Standardeinstellungen belassen und die durch den Ablauf erzeugten Ergebnisse sollen lokal im System abgelegt werden. Die ersten vier Konzeptinstanzen *verify detected vessel findings*, *measure length*, *measure stenosis level* und *measure vessel diameter* sind in den Tasks Coronary und Vascular auswählbar. Die drei weiteren Konzeptinstanzen sind in allen Tasks auswählbar, da sie zu den generellen Operationen zählen.

Abbildung 7.5 stellt die Übersetzung in syngo.via dar. Zusätzlich ist hier ein Schritt *Select Layout*, der sicherstellt, dass zu Beginn der Befundung ein einheitliches Layout vorliegt, da dies durch die Anwender geändert worden sein kann. Des Weiteren werden automatisch die gewählten Gefäße selektiert. Der Anwender wird mit Hilfe dieser definierten Schritte durch die Befundung geführt. Somit wird ein definiertes Ergebnis geschaffen, da die Schritte in fester Reihenfolge ausgeführt werden. Die Eingabe bestimmter Parameter ermöglicht es, die Ergebnisse nach den eigenen Vorstellungen zu definieren. Die Schritte können entweder automatisiert und ohne weitere Eingriffe das Anwenders durchgeführt werden oder es wird eine Bestätigung der Ergebnisse durch den Anwender benötigt. Die Art der Umsetzung hängt von der Implementierung und dem Zielsystem ab. Diese Bestätigung ist durch die medizinische Notwendigkeit gegeben, da gewisse Ergebnisse eine Verifizierung durch den Anwender benötigen. Alle in der Sprache definierten Abläufe werden in entsprechender Art und Weise in syngo.via umgesetzt, sofern dies vom System ermöglicht wird. Für onkologische Workflows bestand im Zeitraum der Umsetzung keine Automatisierungsmöglichkeit, weshalb die Automatisierung nur für koronare und vaskuläre Workflows umgesetzt wurde.

7.3.2 Zukünftige Weiterentwicklung der Sprache

Neben der Erstellung des vorgestellten Prototypen (siehe Abschnitt 7.3.1) sind weitere Entwicklungen denkbar, die die Befundung unterstützen können. Durch die Analyse konnte festgestellt werden, dass mit dem Preprocessing notwendige Berechnungen für die Befundung durchgeführt werden (siehe 7.2). Beispielhaft sei hier die Segmentierung, also Detektierung, der Blutgefäße in der koronaren und vaskulären Applikation genannt. Die entwickelte DSL ermöglicht es, Workflows zu beschreiben, die mehr als eine Applikation abdecken. Für jede Applikation muss ihr spezifisches Preprocessing durchgeführt werden, wenn Bilddaten neu geladen werden, was die Befundung verzögert. Durch die Möglichkeit, Workflows

```
Vascular Task OverviewVascularTask {
  preprocessing options - activate:
    remove bones: true
    remove table: true

  verify detected vessel findings

  measure a distance

  measure an angle

  mark a region of interest: Circle

  measure length for: Runoff right

  measure stenosis level:
      number of reference marker: 2
      diameter type: effective Diameter
      choose vessel: Runoff right

  measure vessel diameter for: A. Renalis right

  create a parallel series of images:
      distance between slices: 25.0
      image thickness: 5.0
      number of images: 25
      image orientation: left to right

  finish the evaluation and: store in local system
}
```

Abbildung 7.4: Taskinstanz

mit mehreren Applikationen zu erstellen, kann das Preprocessing vorgezogen und somit die Befundungszeit verkürzt werden.

In Abbildung 7.6 wird der Ablauf des Preprocessing dargestellt. Für den ersten

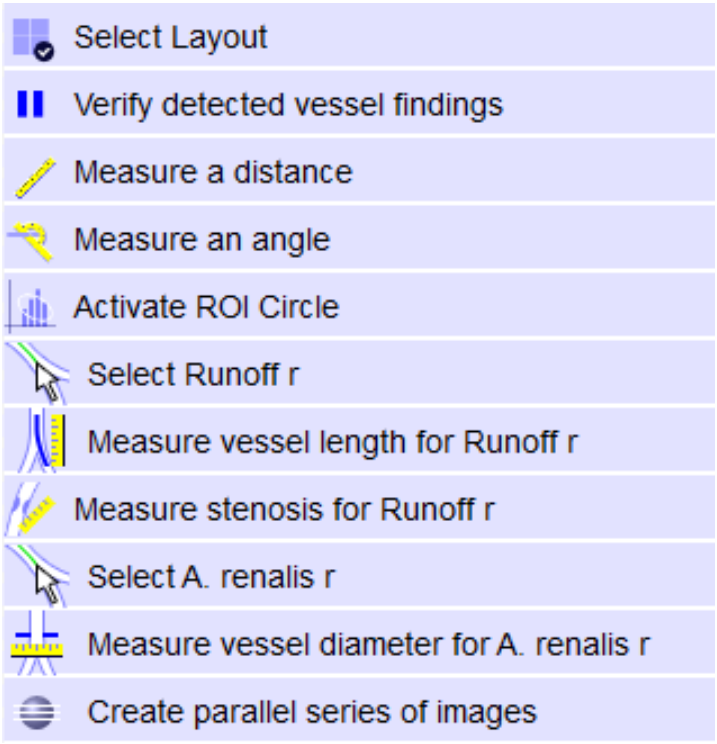

Abbildung 7.5: Überführte Befehle in syngo.via

Task muss bei neuen Datensätzen auf das Preprocessing gewartet werden, bevor die Befundung starten kann. Setzt sich der Workflow aus mehreren Tasks zusammen, kann das Preprocessing für den zweiten Task parallel zur Befundung durch den Anwender des ersten Tasks durchgeführt werden. Somit kann die Wartezeit zwischen den Tasks minimiert oder ausbleiben. In einem Interview[1] wird beschrieben, dass die Befundung eines Tasks in circa zwei Minuten abgeschlossen sein soll. Durch die hier vorgestellte Möglichkeit, die sich aus der Analyse ergab, könnte die Befundungszeit bei der Verwendung mehrerer Applikationen gesenkt werden. Weiterhin empfiehlt sich die Erweiterung der DSL um zusätzliche Applikationen.

7.4 Zusammenfassung

Zu Beginn legten wir Grundlagen für die Implementierung einer Sprache mit MPS dar. Ein wesentlicher Vorteil dieser Implementierung ist, dass keine explizite Erstellung einer Grammatik notwendig ist, da das semantische Modell der Sprache durch Konzepte und deren Verknüpfungen erstellt wird. Durch diese Art

[1] Interview7-22-07-2014-e

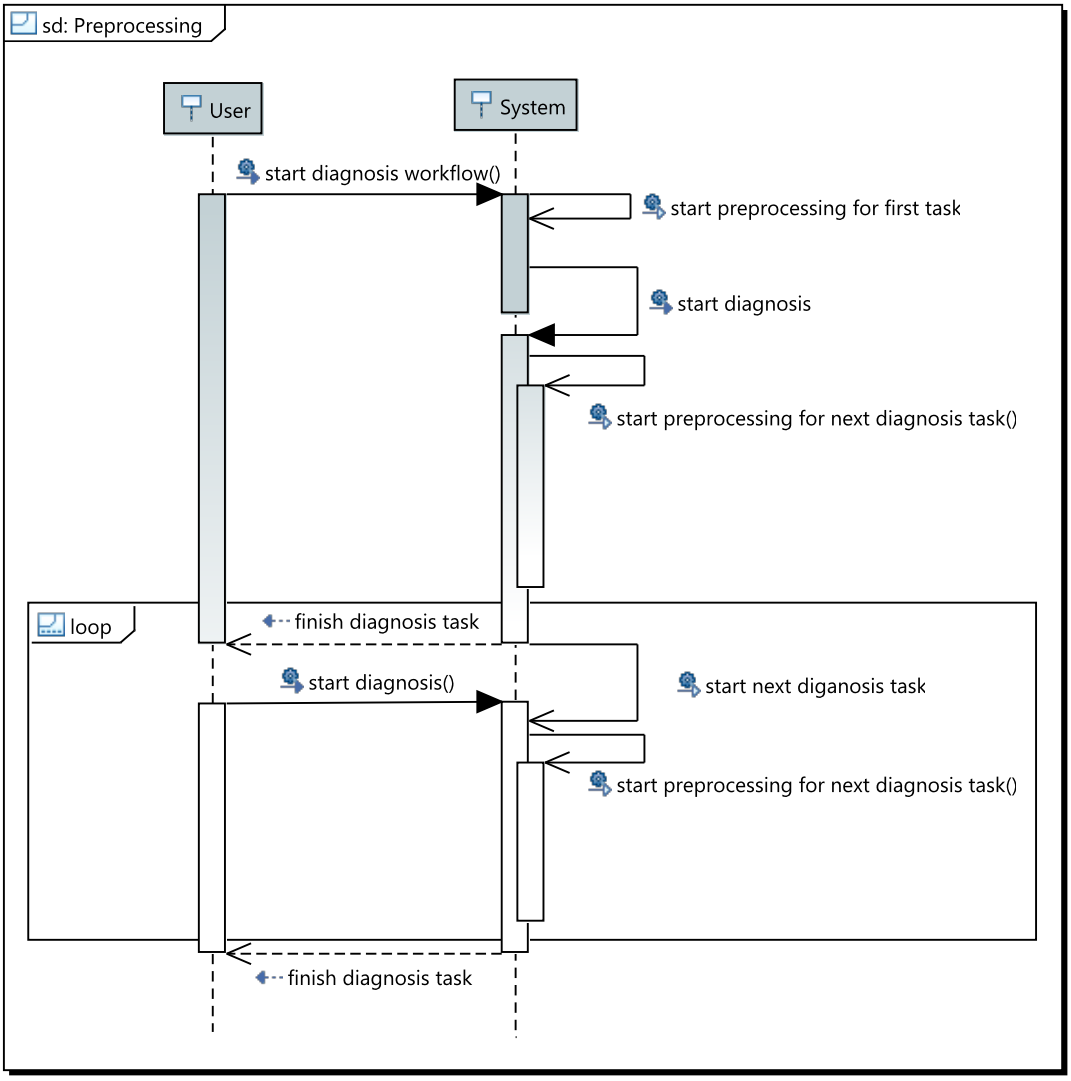

Abbildung 7.6: Sequenzdiagramm Preprocessing

der Sprachimplementierung war eine einfache Übertragung des Domänenmodells in die Sprache möglich. Insbesondere hat hierbei das Featuremodell dazu beigetragen, denn die Features konnten überwiegend direkt in Konzepte übertragen werden. Beispielhaft erläuterten wir das Vorgehen in diesem Kapitel an den generellen Operationen, wie dies bei der Erstellung des Domänenmodells getan wurde. Durch dieses Vorgehen kann eine gute Traceability gewährleistet werden, denn eine direkte Rückverfolgung der Konzepte in das Codesystem der Analyse ist möglich. Anschließend wurde auf die prototypische Umsetzung der Sprache eingegangen. Die Umsetzung basiert auf syngo.via von Siemens und zeigt die Möglichkeiten der Sprache.

8 Bewertung der Sprache

Im Folgenden wird die erstellte DSL bewertet. Wir gehen zunächst auf die Anforderungen einer Sprache ein, die im Abschnitt 3.2.2 detailliert erläutert wurden. Dies dient dazu die Konformität der DSL bezüglich der allgemein definierten Anforderungen zu überprüfen. In einem zweiten Schritt gehen wir auf qualitatives Feedback von Domänenexperten eingehen.

8.1 Allgemeine Bewertung der Sprache

Eine zentrale Anforderung an domänenspezifische Sprachen ist die sogenannte Konformität, die die Übereinstimmung von Konzepten der domänenspezifischen Sprache mit zentralen Konzepten der Domäne beschreibt. Zusätzlich wird dies durch die Orthogonalität einer Sprache gefordert. Sie besagt, dass jedes Konzept einer Sprache genau einem Konzept der Domäne entspricht. Dies stellt eine stärkere Anforderung als die Konformität dar und wird ebenfalls durch die Art und Weise der Erstellung der DSL in dieser Arbeit unterstützt. Des Weiteren ist eine gute Traceability zwischen den Konzepten der Sprache und der Domäne möglich. Wir können davon ausgehen, dass ein hoher Grad an Konformität und Orthogonalität zwischen der erstellten DSL und der Domäne vorhanden ist, denn alle Konzepte der Sprache bauen auf der qualitativen Analyse der Interviews und somit der Sprache der Experten auf.

Eine weitere Anforderung an Sprachen ist deren Erweiterbarkeit. In der Domäne dieser Arbeit ist auf drei Tasks, Coronary, Vascular und Oncology, eingegangen worden. Neben diesen Tasks bietet syngo.via weitere an, die einen anderen Schwerpunkt bei der Diagnose aufweisen. So ist davon auszugehen, dass eine einfache Erweiterbarkeit der Sprache notwendig ist. Durch die Implementierung der Sprache mit JetBrains MPS ist eine einfache Erweiterbarkeit gegeben, denn es müssen entsprechende Konzepte erstellt werden, ohne die Grammatik neu zu

Tabelle 8.1: Bewertung Anforderungen
Sehr gut ⇔ Gut ⇔ Akzeptabel ⇔ Schlecht ⇔ Sehr schlecht

Anforderungen	**Umsetzung**
Konformität	Sehr gut
Orthogonalität	Sehr gut
Unterstützung durch Tools	Sehr gut
Integrierbarkeit	Schlecht
Erweiterbarkeit	Sehr gut
Langlebigkeit	Akzeptabel
Einfachheit	Sehr gut (soweit hier beurteilbar)
Qualität	Gut

definieren. Problematisch ist die Integrierbarkeit der Sprache. Der projektionale Editor macht es notwendig, dass die Sprache mit MPS geöffnet wird, durch die freie Verfügbarkeit der Entwicklungsumgebung ist dies unproblematisch. Weiterhin ist die Langlebigkeit einer Sprache grundlegend für deren Verwendung. Durch den langen Zeitraum der Weiterentwicklung von MPS (Dmitriev, 2004; Jetbrains, 2014a) und die aktive Entwicklung mit MPS wie *mbeddr* für die Entwicklung eingebetteter Software (Voelter, Ratiu, Kolb & Schaetz, 2013) oder der Bugtracker *Youtrack* (Erdweg et al., 2013), ist davon auszugehen, dass MPS weiterentwickelt wird und die Langlebigkeit der DSL anzunehmen ist. Eine abschließende Beurteilung lässt sich über die Langlebigkeit nicht treffen.

Die Qualität einer Sprache wird maßgeblich durch die eingesetzte Entwicklungsumgebung beeinflusst und wird aus diesem Grund von uns nicht weiter betrachtet. Ein weiterer Punkt war die Unterstützung der Sprache durch Tools. MPS stellt hierbei ein Tool dar, was neben der Definition einer Sprache eine breite Unterstützung bei der Erstellung von Anwendungen mit der Sprache bietet (siehe Kapitel 7). Beispielhaft sei hier der projektionale Editor genannt, der die Anwender bei der Erstellung von Programmen oder Abläufen unterstützt und falsche Eingaben somit reduziert werden können. Eine grafische Umsetzung der Sprache wäre realisierbar, da das semantische Modell der Sprache, was durch die Konzepte realisiert ist, beibehalten werden kann und die Präsentation durch die Editoren eine Anpassung ermöglichen. Als letzter Punkt sei hier die Einfachheit der Sprache genannt. Dies ergibt sich zum einen aus der Konformität und der Orthogonalität einer DSL, wenn sie sich an der Sprache der Domänenexperten orientiert, zum anderen ist hierbei die Akzeptanz und die Bewertung der Experten notwendig, auf die im folgenden Abschnitt eingegangen wird. Da die Verknüpfung zwischen der

DSL und der Domäne als hoch zu betrachten ist, können wir von der Einfachheit der DSL ausgehen. Zusätzlich wird durch die Einfachheit, die Benutzerfreundlichkeit der Sprache unterstützt. In Tabelle 8.1 haben wir eine Übersicht der hier aufgeführten Anforderungen dargestellt.

8.2 Bewertung durch Domänenexperten

In einem zweiten Schritt haben wir die Bewertung der DSL durch Domänenexperten erfasst. Zusätzlich fließen neben der Bewertung von interviewten Experten, Bewertungen von nicht interviewten Experten ein. Feedbacks wurden zum einen mündlich und zum anderen schriftlich gegeben.

Grundlegend wurde ein positives Feedback gegeben. Insbesondere wurden die einfachen Konzepte der Sprache hervorgehoben. Es wurde eine intuitive Verwendung der Sprache beschrieben, insofern man mit der Verwendung des Editors vertraut ist. Weiterhin wurde von den Domänenexperten eine gute Verständlichkeit der DSL zum Ausdruck gebracht. Die hier aufgeführten Punkte führen uns zur Annahme, dass die Anforderung der Einfachheit dieser DSL als gegeben anzusehen ist. Die Möglichkeit taskübergreifende Workflows zu erstellen, wurde ebenfalls als positiv hervorgehoben, da dies neue Möglichkeiten bei der Befundung erlaubt. Der Umgang mit MPS wurde zunächst als gewöhnungsbedürftig beschrieben. Ein Großteil der Experten ist mit textuellen Editoren von anderen Entwicklungsumgebungen vertraut und musste sich auf die projektionale Editierung mittels MPS einstellen. Die entsprechende Unterstützung seitens des Editors wurde als positiv empfunden, da Workflows erstellt werden können, die konform zur Sprache sind. Dieses Feature ist MPS zuzuschreiben. Die DSL wurde nicht als überladen wahrgenommen und laut der Feedbacks fand eine Fokussierung auf zentrale Domänenelemente statt. Somit kann die Konformität der DSL bestätigt werden. Der breite Funktionsumfang, der durch die Sprache zur Verfügung gestellt wird, ist ebenfalls positiv eingeschätzt worden. Ebenso wurde die Traceability der Konzepte der DSL als positiv gewertet, alle Konzepte sind in den Interviews der Anwender und somit in deren beschriebenen Abläufen auffindbar. Dies ist auf die enge Verzahnung der Erstellung der DSL mit der Analyse zurückzuführen. Die Analyse der Domäne beruht auf Interviews mit Anwendern von syngo.via, trotz dieser Grundlage wurden von den Experten die Unabhängigkeit der DSL bestätigt. Dies sollte Anwendern anderer Plattformen einen einfachen Einstieg in diese DSL ermöglichen. Zudem wurde positiv hervor gehoben, dass die Abläufe offline gestaltet werden können, also unabhängig von einer Zielplattform sind.

Diesem positiven Feedback stehen Verbesserungsvorschläge und Kritik gegenüber. Von einzelnen Experten wurden Anregungen zu weiteren Funktionen, also Konzepten gegeben, die aus ihrer Sicht für die Sprache notwendig sind. Vorgeschlagene Funktionalitäten waren zum Beispiel die Auswahl vordefinierter Layouts und die Möglichkeit Makros zu definieren um gleiche Abläufe immer wieder verwenden zu können. Weiterhin könnte die Auswahl konkreter Layoutsegmente die Verwendung unterstützen und die Verwendung technischer Winkel zur Ausrichtung der Segmente als zusätzliche Funktion eingerichtet werden. Ebenso wurde es in Betracht gezogen, die DSL für weitere Applikationen zu vervollständigen. In dieser Arbeit wurde die Analyse auf drei Applikationen beschränkt. Durch die Möglichkeiten von MPS ist eine einfache Erweiterbarkeit der DSL gegeben. Zum einen könnten neue Funktionen und zum anderen komplette Applikationen mit eigenen Funktionen integriert werden. Die Erweiterbarkeit der DSL wurde im Feedback der Architekten als leicht eingestuft. Die Erweiterbarkeit ist der Entwicklungsumgebung MPS zuzuschreiben. Neben den aufgeführten Punkten, wurde von technischen Domänenexperten die Anregung gegeben, Iterationen einzubauen, um zum Beispiel für alle Gefäße den Grad der Stenose zu messen, ohne dies einzeln aufführen zu müssen. Dabei stellt sich die Frage, ob nichttechnische Anwender wie Ärzte die Funktion als hilfreich betrachten würden. Diese Frage stellt sich auch bei Vorbedienungen oder Verzweigungen, die genannt wurden. Verzweigungen wären beispielsweise dann sinnvoll, wenn Gefäße nicht automatische detektiert werden könnten und man einen alternativen Ablauf vorgibt. Solche komplexeren Abläufe wurden jedoch von den Experten nicht beschrieben, könnten aber als Grundlage für weitere Forschung dienen. Einige Experten fänden eine grafische DSL als hilfreich. Solch eine Umsetzung wäre denkbar, da die Konzepte der DSL durch grafische Abbildungen ergänzt werden könnten. Schwierig gestaltet sich noch die Importierbarkeit der erstellten Abläufe in die Zielplattform syngo.via, dies ist jedoch von der jeweiligen Umsetzung der Zielplattform abhängig und wird hier nicht weiter betrachtet.

Die hier aufgeführten Erweiterungsvorschläge und konstruktiven Hinweise zur Verbesserung der DSL lassen mehrere Schlüsse auf die Ausführung des Verfahrens zu. Der Wunsch nach weiteren Funktionalitäten könnte ein Anlass sein ,weitere Interviews in diesem Bereich durchzuführen. Eventuell würde sich hierfür ein iteratives Vorgehen empfehlen. Die Erstellung der DSL in dieser Arbeit kann als erste Iteration verstanden werden, weiteres Feedback könnte in Form von Interviews durch das in dieser Arbeit vorgestellte Vorgehen in die DSL einfließen. Das gewonnene Feedback könnte somit als Grundlage für weitere Fragen dienen, um Punkte zu beleuchten, die in den ersten Interviews nicht benannt wurden.

8.3 Zusammenfassung

Zusammenfassend kann durch die Beurteilung der Sprache anhand der Anforderungen und der Feedbacks ein positives Bild der DSL festgestellt werden. Der einfache, verständliche Aufbau der DSL, der durch klare Konzepte erreicht wurde, die auf der Analyse der Interviews beruhen, wird als positiv bewertet. Einige Schwächen und Verbesserungsvorschläge der DSL wurden von den Experten aufgezeigt. Diese könnten durch weitere Interviews in die DSL integriert werden. Weiterhin würde sich ein iteratives Vorgehen empfehlen, um die DSL weiterzuentwickeln.

9 Fazit und Ausblick

In diesem Kapitel stellen wir eine Zusammenfassung der vorliegenden Arbeit vor. Weiterhin geben wir einen Überblick des durchgeführten explorativen Prozesses zur Erstellung einer domänenspezifischen Sprache. Abschließend geben wir einen Ausblick auf mögliche Forschung und weiterführende Arbeiten.

9.1 Fazit und Zusammenfassung

Diese Arbeit stellt zunächst wichtige Grundlagen zum Verständnis dar. Dafür haben wir im Kapitel 2 Ausführungen zur Durchführung qualitativer Datenanalyse gegeben. Wir konzentrierten uns auf die Darlegung wichtiger Prozesse der qualitativen Datenanalyse und folgten der Grounded Theory nach Glaser und Strauss. Als zentrale Konzepte seien hier der permanente Vergleich, das theoretische Sampling, die theoretische Sättigung und die Triangulation genannt. Das Framework der Fallstudienanalyse nach Eisenhardt wurde ebenso vorgestellt und verwendet ähnliche Konzepte wie die Grounded Theory. Im Kapitel DSL (Kapitel 3) haben wir grundlegende Eigenschaften und Bestandteile domänenspezifischer Sprache dargestellt. Für die Entwicklung einer Sprache konnten drei zentrale Schritte herausgearbeitet werden, wie die Analyse der Domäne und die Erstellung eines Domänenmodells, die Entwicklung der DSL und zugehöriger Tools, sowie die Verwendung der Sprache. Zur Entwicklung der Sprache gaben wir einen Überblick über die Entwicklungsumgebung MPS von JetBrains. Weiterhin wurden Vorgehensweisen zur Erstellung domänenspezifischer Sprachen dargestellt, die in der Literatur erwähnt werden. Es wurden Herausforderungen umrissen, die bei der Erstellung von DSL bestehen. Als Herausforderungen seien hier die Abhängigkeit vom Domänenanalyst benannt, die klaren Anforderungen der Anwender und die informalen Vorgehensweisen.

Im Kapitel 4 haben wir die hier zu untersuchende Domäne umrissen und Konzep-

te der medizinischen Bildgebung dargestellt. Die Durchführung der qualitativen Analyse der gewonnenen Daten durch Anwenderinterviews wurde im Kapitel 5 beschrieben. Während der Interviews wurden Anwender zur Durchführung von Befundungen befragt. Durch die Analyse konnte eine Einordnung der Domäne in ihren Kontext gefunden werden. Weiterhin wurde die Aufteilung der Domäne in ihre Applikationen und Eigenschaften dargestellt. Diese Aufteilung der Analyse konnte in Kapitel 6 zur Erstellung eines Domänenmodells verwendet werden. Das Domänenmodell untergliedert sich in ein Featuremodell und ein Klassendiagramm. Wie wir zeigen konnten ist eine enge Verzahnung zwischen der Analyse und der Erstellung des Domänenmodells vorhanden, so dass sich alle Elemente des Domänenmodells auf die Analyse zurück führen lassen und somit in den Anwenderinterviews fundiert sind. Auf diese Weise war eine direkte Übertragung des Codesystems in das Domänenmodell möglich.

Aus dem erstellten Domänenmodell wurde in Kapitel 7 die DSL mit Hilfe von MPS entwickelt. Auch hier war eine direkte Übertragung der Funktionen und Eigenschaften aus dem Domänenmodell möglich. Somit basieren die sogenannten Sprachkonzepte der DSL jeweils auf den analysierten Daten. In diesem Kapitel wurde die einfache Übertragung des Domänenmodells in die Sprache gezeigt. Außerdem wurde auf die prototypische Umsetzung eingegangen und diese an dem Beispiel von syngo.via der Siemens Healthcare AG erläutert. Die gewonnene DSL wurde im Kapitel 8 bewertet. Einerseits ist auf ausgewählte Anforderungen einer Sprache eingegangen worden, bei denen gezeigt wurde, dass der Großteil der Anforderungen als gut oder sehr gut erfüllt betrachtet werden kann. Andererseits wurde auf qualitatives Feedback von Domänenexperten eingegangen, welches die verständlichen und einfachen Konzepte der Sprache und die daraus resultierende intuitive Verwendung der Sprache hervorhebt. Insgesamt wurde die gewonnene DSL von den Anwendern positiv aufgenommen.

Der in dieser Arbeit vorgestellte Prozess kann in folgende Teilschritte unterteilt werden:

1. Analyse der Daten mit Hilfe von QDA
2. Ableitung eines Domänenmodells aus dem Codesystem
3. Erstellung der DSL und entsprechender Tools
4. Verwendung der Sprache und Gewinnung von Feedback

Die Schritte eins und zwei könnten wie in Kapitel 3 beschrieben, zusammengefasst werden, da eine enge Verzahnung zwischen der Analyse und dem Domänenmo-

dell besteht. Für die bessere Übersichtlichkeit wurden zwei separate Schritte aufgeführt. Wie in Abschnitt 8.2 ausgeführt, können diese Schritte als ein iterativer Prozess verstanden werden, um gewonnenes Feedback in die Weiterentwicklung einfließen zu lassen. Mit Hilfe des Feedbacks können neue Fragestellungen aufgeworfen werden und durch weitere Interviews und deren Analyse einfließen.

Weiterhin können durch diesen hier durchgeführten Prozess Herausforderungen (siehe Abschnitt 3.2.3) bei der Entwicklung solcher Sprachen minimiert werden. Es ist im Sinne der Domänenanalyse kein erfahrener Domänenanalyst mehr notwendig. Für die Analyse muss ausschließlich der Prozess der QDA verstanden sein und somit sind keine oder wenige Vorkenntnisse der Domäne notwendig. Zusätzlich stellt die Analyse bei unserem Prozess einen formalen Weg dar. In vielen Fällen wurde die Analyse auf eine eher informale Art und Weise durchgeführt und war von den Erfahrungen des Domänenanalysten abhängig. Die Anwender müssen sich nicht über notwendige Anforderungen im klaren sein, da diese aus der Analyse entsprechender Interviews und weiterer Daten gewonnen werden. Dies lässt sich dadurch erklären, dass Anwender nicht mehr danach gefragt werden, was sie benötigen, sondern sie erklären durch „Wie“- und „Warum“-Fragen ihre Prozesse. Dies grenzt auch den hier erprobten Prozess von den in Abschnitt 3.2.2 vorgestellten Vorgehensweisen ab. Weiterhin ist die DSL in den Interviews durch die qualitative Analyse fundiert. Zusätzlich kann die Traceability der Sprachkonzepte gewährleistet werden, da jedes Sprachkonzept auf einem Code des Codesystems basiert.

Neben diesen Vorteilen dieses Prozesses sehen wir neue Herausforderungen. Durch die geführten Interviews entsteht ein große Datenmenge, die verarbeitet und analysiert werden muss. Weiterhin muss ein Verständnis der qualitativen Datenanalyse vorhanden sein. Problematisch kann das festlegen eines Abstraktionslevels sein, was durch das Verfahren nur bedingt übernommen wird. Während der Analyse entstanden Codes und übergeordnete Kategorien, die wir nicht einheitlich in Konzepte der DSL übersetzen konnten.

Durch die positive Bewertung der Sprache durch Domänenexperten und der erfolgreichen Umsetzung der DSL bewerten wir diesen, in dieser Arbeit vorgestellten Prozess, positiv. Dieser Prozess zeichnet sich durch sein formale und nachvollziehbare Durchführung aus und überwindet Herausforderungen, die bei der Erstellung einer DSL entstehen.

9.2 Ausblick

Nachdem wir in dieser Arbeit ein neues Verfahren zur Entwicklung domänenspezifischer Sprachen durchführen konnten, sind weitere Tests in der Praxis vorstellbar. Eventuell wäre ein Vergleich existierender Vorgehensweisen zu Erstellung domänenspezifischer Sprachen mit dem hier vorgestellten Prozess denkbar. Dafür könnten analysierte Domänen mit diesem Verfahren untersucht werden, um einen Vergleich der DSL zu erhalten. Denkbar ist eine automatisierte Überführung des Codesystems in das Domänenmodell und zusätzlich in eine DSL. Dafür wäre Kennzeichnung von Funktionen und Eigenschaften hilfreich, um diese trennen zu können, was mit einem Featuremodell nicht möglich ist. Wie wir zeigen konnten war eine direkte Übertragung der Codes aus der Analyse in das Klassendiagramm möglich und dieser Schritt könnte automatisiert werden. Weiterhin sollte die iterative Ausführung des Prozesses getestet werden. Wir verstehen den Prozess in dieser Arbeit als erste Iteration, weitere Iterationen könnten das Feedback von Experten einarbeiten und die Sprache weiterentwickeln.

Anhang

A Featuremodell

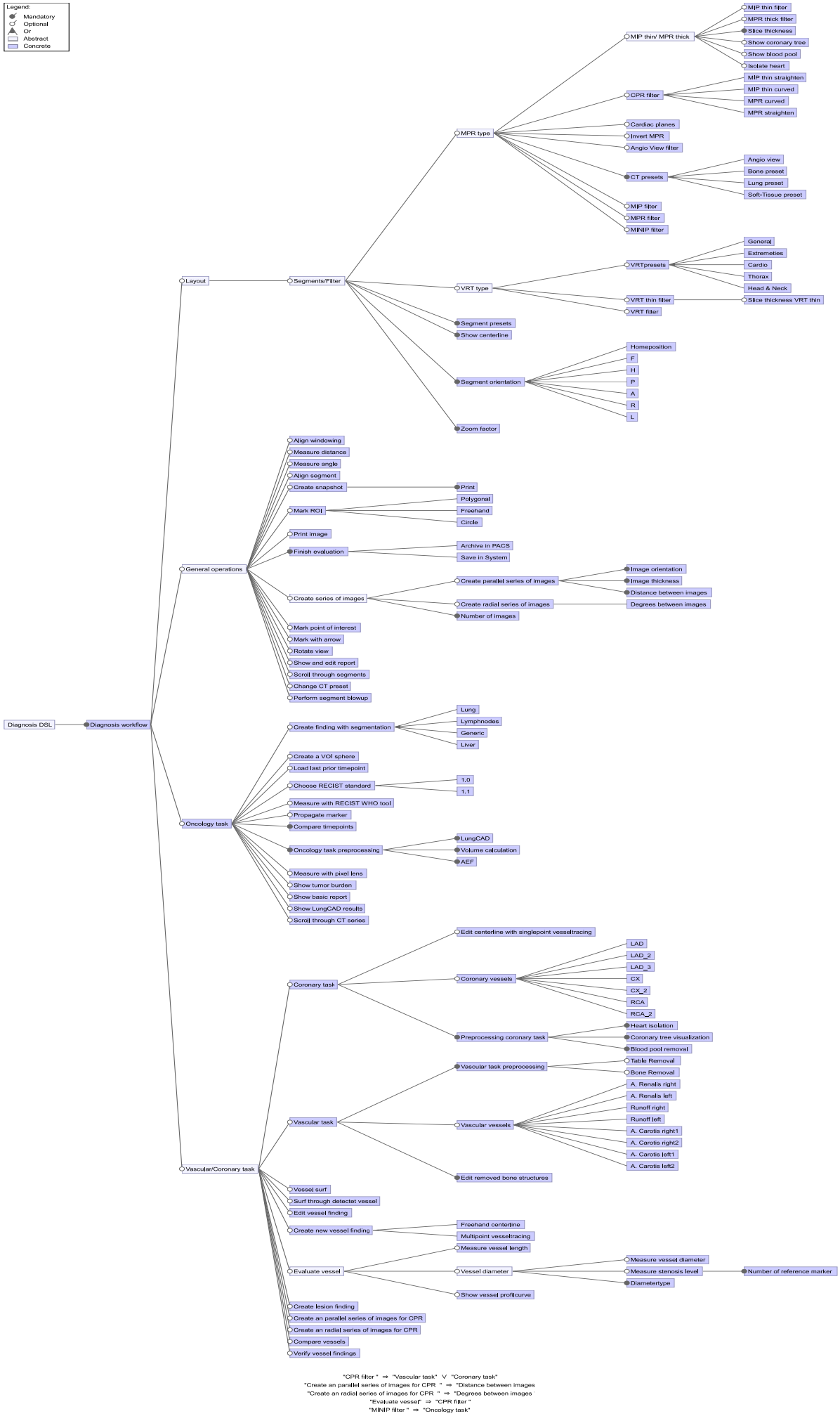

Abbildung 1: Featuremodell

B Klassendiagramm

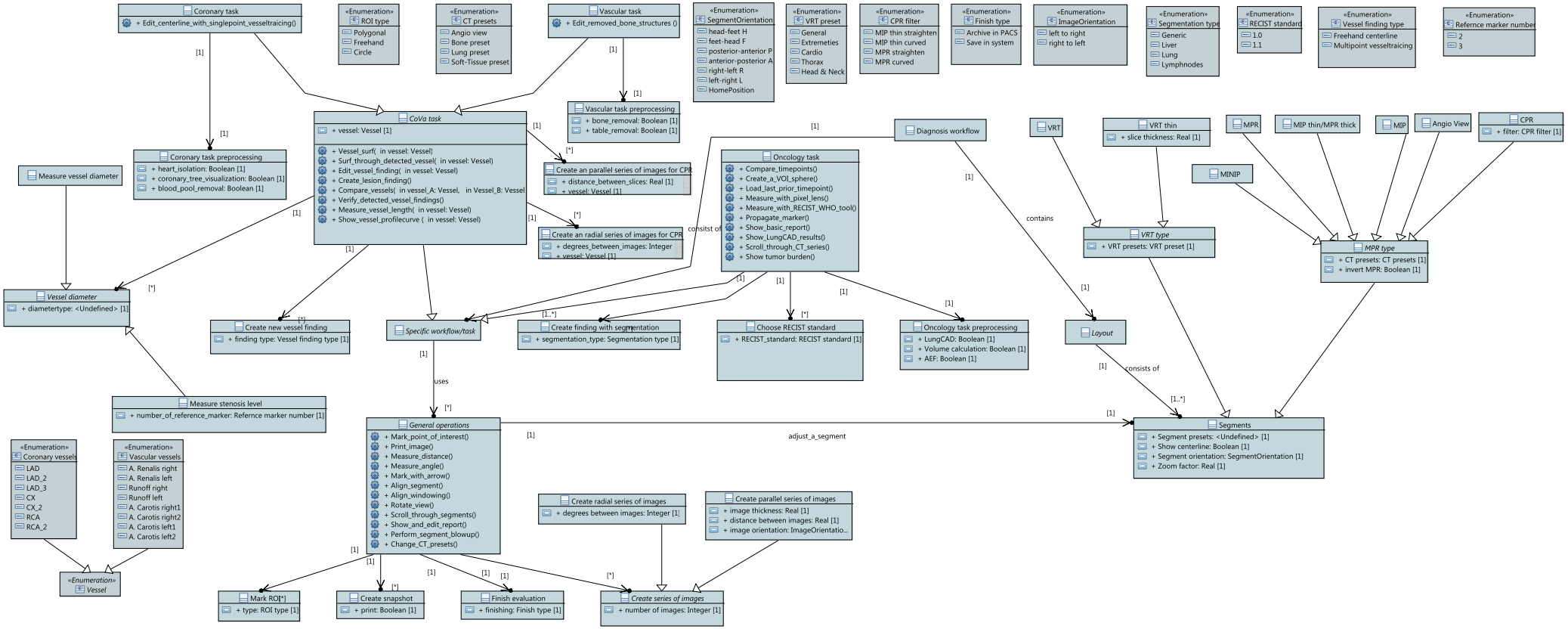

Abbildung 2: Klassendiagramm

C Konzeptübersicht

Konzept	Erläuterung
Diagnosis Workflow	Ermöglicht die Erstellung eines Workflows, der aus der Kombination unterschiedlicher Tasks besteht.
Coronary Task	Erlaubt die Erstellung eines koronaren Workflows.
Oncology Task	Erlaubt die Erstellung eines onkologischen Workflows.
Vascular Task	Erlaubt die Erstellung eines vaskulären Workflows.
General Operations – Verfügbar in allen Tasks	
Align MIP segment	Setzt für ein Segment ein MIP Filter und setzt weitere Eigenschaften.
Align MIP thin segment	Setzt für ein Segment den MIP thin Filter und weitere Eigenschaften.
Align MPR segment	Setzt für ein Segment den MPR Filter und weitere Eigenschaften.
Align MPR thick segment	Setzt für ein Segment den MPR thick Filter und weitere Eigenschaften.
Align VRT segment	Setzt für ein Segment den VRT Filter und weitere Eigenschaften.
Align VRT thin segment	Setzt für ein Segment den VRT thin Filter und weitere Eigenschaften.
Align windowing	Erlaubt es die Helligkeit und den Kontrast eines Segments einzustellen.
Change CT presets	CT Presets sind vordefinierte windowing Werte. Es besteht die Auswahl zwischen angio (Angiographie), bone (Knochen), lung (Lunge) oder soft-tissue (Weichteilgewebe).
Create a parallel series of images	Erstellt eine parallele Serie von Bilden in einem Segment.
Create a radial series of images	Erstellt eine radiale Serie von Bilden in einem Segment.
Create a snapshot	Erstellt eine Schnappschuss eines Segments.

Finish evaluation	Diese Konzept kann nicht gewählt werden. Es ist das letzte Konzept in jedem Task oder Workflow und ermöglicht es Daten lokal abzulegen oder zu archivieren.
Mark a point of interest	Erlaubt es einen Punkt zu markieren.
Mark a region of interest (MarkROI)	Erlaubt es eine Region zu markieren. Freihändig, mit einem Kreis oder Polygonal.
Mark with an arrow	Erlaubt es eine Punkt mit einem Pfeil zu markieren.
Measure an angle	Erlaubt es einen Winkel zu messen.
Measure a distance	Erlaubt es eine Distanz zu messen.
Print image	Sendet ein Bild eines Segments zum Druck.
Rotate view	Aktiviert die Segmentrotation und erlaubt es, die Sicht eines Segments zu rotieren.
Scroll through segments	Aktiviert eine Funktion um durch Segmente zu scrollen. Es erlaubt nach Läsionen zu suchen, bis entweder Läsionen gefunden wurden, oder keine Segmente mehr zu untersuchen sind.
Show and edit reports	Zeigt den aktuellen klinischen Report zur Befundung und erlaubt die Anpassung des Reports.
Coronary/Vascular operations – Verfügbar in koronaren und vaskulären Tasks	
Align Angio View segment	Setzt für ein Segment den Angio View Filter und weitere Eigenschaften.
Align CPR segment	Setzt für ein Segment den CPR Filter und weitere Eigenschaften.
Create a parallel CPR series of images	Erzeugt einen Stapel von Schnittbildern entlang des ausgewählten Gefäßpfades. Es ermöglicht, den Gefäßstatus zu dokumentieren.
Create a radial CPR series of images	Speichert einen Stapel von Bildern mit mehreren CPR Ansichten des angezeigten Gefäßes aus verschiedenen Blickrichtungen. Es ermöglicht, den Gefäßstatus zu dokumentieren.

Create a lesion finding	Erzeugt einen Fund zu einer Läsion.
Create new centerline for vessel	Erlaubt es eine neue Centerline für ein Gefäß zu erzeugen. Die Segmentierung kann freihändig oder unterstützt vom System durchgeführt werden.
Measure length of a vessel	Aktiviert eine Funktion um die Länge oder den Abschnitt eines ausgewählten Gefäßes zu messen.
Measure stenosis level of a vessel	Aktiviert eine Funktion um den Stenosegrad eines ausgewählten Gefäßes zu ermitteln.
Measure vessel diameter	Aktiviert eine Funktion um den Gefäßdurchmesser eines ausgewählten Gefäßes zu ermitteln.
Show centerlines	Zeigt alle segmentierten Centerlines in den Gefäßen an.
Verify detected vessel findings	Diese Funktion gibt den Hinweis, dass der Anwender alle detektierten Gefäße überprüfen sollte, um sicherzustellen, dass die Segmentierung korrekt ausgeführt wurde.
Surf through vessels	Aktiviert eine Funktion um durch Gefäße zu scrollen. Es erlaubt nach Läsionen zu suchen, bis entweder Läsionen gefunden wurden, oder keine Gefäße mehr zu untersuchen sind.
Vascular operations – Verfügbar im vaskulären Task	
Edit removed bone structures	Aktiviert eine Funktion um segmentierte Knochenstrukturen zu editieren, die durch das Preprocessing segmentiert wurden.
Oncology operations – Verfügbar im onkologischen Task	
Align MINIP segment	Setzt für ein Segment den MINIP Filter und weitere Eigenschaften.
Choose RECIST standard	Erlaubt es einen RECIST Standard zu wählen. Es besteht die Auswahl zwischen Standard 1.0 und 1.1.

Compare timepoints	Erlaubt es verschiedene Zeitpunkte zu vergleichen. Dabei ist es mögliche die aktuelle mit der letzten Befundung zu vergleichen.
Create a VOI sphere	Erlaubt es ein Volumen des Interesses mit einer Kugel zu markieren.
Create a finding with RECIST WHO tool	Aktiviert eine Funktion, um eine Läsionsfund mit dem RECIST Standard zu erstellen.
Create Finding With Segmentation/ Perform segmentation	Erlaubt es Läsionen zu segmentieren. Diese Segmentierung ist für Leber, Lunge, Lymphknoten und generelles Gewebe möglich.
Evaluate a VOI	Erlaubt es ein Volumen des Interesses zu bewerten.
Load the last prior timepoint	Lädt den letzten befundeten Zeitpunkt.
Measure pixel lens	Erlaubt es Abweichungen der Grauwerte in einem bestimmten Bereich zu messen.
Propagate marker	Aktiviert eine Funktion um Markierte Bereiche zu propagieren und mit anderen Zeitpunkten zu verknüpfen.
Show basic report	Zeigt den Basis Report bei onkologischen Untersuchungen an.
Show LungCAD results	Zeigt die segmentierten Läsionen an, die durch den LungCAD Algorithmus im Preprocessing segmentiert wurden.
Show tumor burden	Zeigt die Tumorlast bei einer Befundung an.
Start scrolling through series	Erlaubt es durch segmentierte Zeitserien zu scrollen.

Tabelle 1: Konzeptübersicht

Literaturverzeichnis

Alonso, O. (2002). Generation of text search applications for databases. an exercise on domain engineering. In *Software reuse: Methods, techniques, and tools* (S. 179–193). Springer.

Arango, G. (1989). Domain analysis: From art form to engineering discipline. *ACM Sigsoft software engineering notes*, *14* (3), 152–159.

Balzert, H. & Balzert, H. (2009). Prinzipien. *Lehrbuch der Softwaretechnik: Basiskonzepte und Requirements Engineering*, 25–51.

Benbasat, I., Goldstein, D. K. & Mead, M. (1987). The case research strategy in studies of information systems. *MIS quarterly*, 369–386.

Brant, W. E. & Helms, C. A. (2012). *Fundamentals of diagnostic radiology*. Lippincott Williams & Wilkins.

Campagne, F. (2014). *The mps language workbench: Volume i* (Bd. 1). Fabien Campagne.

Charmaz, K. (2008). Grounded theory as an emergent method. In I. P. L. . S. N. H.-B. (Eds.) (Hrsg.), *Handbook of emergent methods* (S. 81—110). Gilford Press. Zugriff auf `http://www.sxf.uevora.pt/wp-content/uploads/2013/03/Charmaz_2008-b.pdf` (Online; abgerufen am 24.11.2014)

Coplien, J., Hoffman, D. & Weiss, D. (1998). Commonality and variability in software engineering. *Software, IEEE*, *15* (6), 37–45.

Corbin, J. & Strauss, A. (1994). Grounded theory methodology. *Handbook of qualitative research*, 273–285.

Corbin, J. & Strauss, A. (2008). *Basics of qualitative research: Techniques and procedures for developing grounded theory*. Sage.

DiCicco-Bloom, B. & Crabtree, B. F. (2006). The qualitative research interview. *Medical education*, *40* (4), 314–321.

Dmitriev, S. (2004). *Language Oriented Programming: The Next Programming Paradigm.* `http://www.jetbrains.com/mps/docs/Language_Oriented_Programming.pdf`. ([Online; abgerufen am 23.11.2014])

Duden. (2014). *Domäne.* http://www.duden.de/rechtschreibung/Domaene. ([Online; abgerufen am 24.11.2014])

Eisenhardt, K. M. (1989). Building theories from case study research. *Academy of management review*, *14* (4), 532–550.

Eisenhardt, K. M. & Graebner, M. E. (2007). Theory building from cases: opportunities and challenges. *Academy of management journal*, *50* (1), 25–32.

Erdweg, S., van der Storm, T., Völter, M., Boersma, M., Bosman, R., Cook, W. R., ... others (2013). The state of the art in language workbenches. In *Software language engineering* (S. 197–217). Springer.

Fowler, M. (2005). Language workbenches: The killer-app for domain specific languages. *Martin Fowler*.

Fowler, M. (2010). *Domain-specific languages*. Pearson Education.

Frakes, W., Prieto, R., Fox, C. et al. (1998). Dare: Domain analysis and reuse environment. *Annals of Software Engineering*, *5* (1), 125–141.

Frakes, W. B. & Kang, K. (2005). Software reuse research: Status and future. *IEEE transactions on Software Engineering*, *31* (7), 529–536.

Glaser, B. G. & Strauss, A. (1967). The discovery grounded theory: strategies for qualitative inquiry. *Aldin, Chicago*.

Glaser, B. G. & Strauss, A. L. (1965). Discovery of substantive theory: A basic strategy underlying qualitative research. *American Behavioral Scientist*, *8* (6), 5–12.

Hancock, B., Ockleford, E. & Windridge, K. (1998). *An introduction to qualitative research*. Trent focus group Nottingham.

Hofer, M. (2010). *Ct-kursbuch: Ein arbeitsbuch für den einstieg in die computertomographie*. Didamed-Verlag.

Hudak, P. (1997). Domain-specific languages. *Handbook of Programming Languages*, *3*, 39–60.

Imran, S., Foping, F., Feehan, J. & Dokas, I. M. (2010). Domain specific modeling language for early warning system: using idef0 for domain analysis. *Int. J. Comput. Sci*, *7* (5), 10–17.

Itemis AG. (o. J.). *About Xtext.* http://xtext.itemis.com/xtext/language=en/36527/about-xtext. ([Online; abgerufen am 09.09.2014])

Jetbrains. (2014a). *Meta Programming System.* http://www.jetbrains.com/mps/. ([Online; abgrufen am 23.11.2014])

Jetbrains. (2014b). *MPS User's Guide MPS 3.0 Documentation.* https://confluence.jetbrains.com/display/MPSD30/MPS+User's+Guide. ([Online; abgerufen am 23.11.2014])

Jones, J. (2003). Abstract syntax tree implementation idioms. In

Proceedings of the 10th conference on pattern languages of programs (plop2003). Zugriff auf `http://www.hillside.net/plop/plop2003/Papers/Jones-ImplementingASTs.pdf`

Kang, K. C., Cohen, S. G., Hess, J. A., Novak, W. E. & Peterson, A. S. (1990). Feature-oriented domain analysis (foda) feasibility study.

Kanitsar, A., Wegenkittl, R., Fleischmann, D. & Groller, M. E. (2003). Advanced curved planar reformation: Flattening of vascular structures. In *Proceedings of the 14th ieee visualization 2003 (vis'03)* (S. 7).

Kolovos, D. S., Paige, R. F., Kelly, T. & Polack, F. A. (2006). Requirements for domain-specific languages. In *Proc. of ecoop workshop on domain-specific program development (dspd)* (Bd. 2006).

Lacey, A. & Luff, D. (2009). *Qualitative data analysis.* The NIHR RDS EM / YH.

Laws, K. & McLeod, R. (2004). Case study and grounded theory: Sharing some alternative qualitative research methodologies with systems professionals. In *Proceedings of the 22nd international conference of the systems dynamics society.*

Lind, M. (2003). Making sense of the abstraction hierarchy in the power plant domain. *Cognition, Technology & Work*, *5* (2), 67–81.

Lintern, G. (2006). Foundational issues for work domain analysis. In *Proceedings of the human factors and ergonomics society annual meeting* (Bd. 50, S. 432–436).

Lisboa, L. B., Garcia, V. C., Lucrédio, D., de Almeida, E. S., de Lemos Meira, S. R. & de Mattos Fortes, R. P. (2010). A systematic review of domain analysis tools. *Information and Software Technology*, *52* (1), 1–13.

Mernik, M., Heering, J. & Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM computing surveys (CSUR)*, *37* (4), 316–344.

Merriam, S. B. (1998). *Qualitative research and case study applications in education. revised and expanded from"case study research in education.".* ERIC.

Merriam, S. B. et al. (2002). Introduction to qualitative research. *Qualitative research in practice: Examples for discussion and analysis*, 3–17.

Neighbors, J. M. (1984). The draco approach to constructing software from reusable components. *Software Engineering, IEEE Transactions on* (5), 564–574.

Pech, V. (2014, 4). *Jetbrains mps roadmap (draft).* Zugriff auf `https://confluence.jetbrains.com/display/MPS/MPS+public+roadmap` (Online; abgerufen am 19.11.2014)

Prieto-Díaz, R. (1990). Domain analysis: An introduction. *ACM SIGSOFT*

Software Engineering Notes, *15* (2), 47–54.

Radiology of Stauferklinikum, G., Mutlangen. (2014). *Vascular imaging - run-off ct angiography.* Zugriff auf http://www.healthcare.siemens.co.uk/siemens_hwem-hwem_ssxa_websites-context-root/wcm/idc/groups/public/@global/@imaging/@ct/documents/image/mdax/oda5/~edisp/ct_somatom-perspective_clinical-images_vascular_run-off-ct-angiography-00881448/~renditions/ct_somatom-perspective_clinical-images_vascular_run-off-ct-angiography-00881448~8.jpg (Online; abgerufen am 24.11.2014)

Rupp, C., Simon, M. & Hocker, F. (2009). Requirements engineering und management. *HMD Praxis der Wirtschaftsinformatik*, *46* (3), 94–103.

Siemens Healthcare AG. (2014a). *Coronary Analysis.* https://www.healthcare.siemens.com/computed-tomography/options-upgrades/clinical-applications/syngo-ct-coronary-analysis. ([Online; abgerufen am 24.11.2014])

Siemens Healthcare AG. (2014b). *Oncology Analysis.* http://www.healthcare.siemens.com/computed-tomography/clinical-imaging-solutions/ct-oncology-engine. ([Online; abgerufen am 24.11.2014])

Siemens Healthcare AG. (2014c). *syngo.via.* http://www.healthcare.siemens.com/medical-imaging-it/clinical-imaging-applications/syngovia. ([Online; abgerufen am 24.11.2014])

Siemens Healthcare AG. (2014d). *Vascular Analysis.* https://www.healthcare.siemens.com/computed-tomography/options-upgrades/clinical-applications/syngo-ct-vascular-analysis. ([Online; abgerufen am 24.11.2014])

Sommerville, I. & Sawyer, P. (1997). *Requirements engineering: a good practice guide.* John Wiley & Sons, Inc.

Strauss, A. (2004). Methodologische grundlagen der grounded theory. *Strübing, J./Schnettler, B.(Hg.): Methodologie interpretativer Sozialforschung. Klassische Grundlagentexte. Konstanz: UVK*, 427–452.

Strauss, A. & Corbin, J. M. (1998). *Basics of qualitative research : Techniques and procedures for developing grounded theory.* Sage Publications, Inc.

The Eclipse Foundation. (o. J.). *Eclipse Modeling Project.* http://eclipse.org/modeling/. ([Online; abgerufen am 09.09.2014])

The Eclipse Foundation. (2013). *Xtext Documentation.* http://www.eclipse.org/Xtext/documentation/2.5.0/Xtext%20Documentation.pdf. ([Online; abgerufen am 23.11.2014])

Urquhart, C., Lehmann, H. & Myers, M. D. (2010). Putting the 'theory'back into grounded theory: guidelines for grounded theory studies in information

systems. *Information systems journal*, *20* (4), 357–381.

Van Deursen, A. & Klint, P. (1998). Little languages: Little maintenance? *Journal of software maintenance*, *10* (2), 75–92.

Van Deursen, A., Klint, P. & Visser, J. (2000). Domain-specific languages: An annotated bibliography. *Sigplan Notices*, *35* (6), 26–36.

Völter, M. (2009). *Werkzeuge für domänenspezifische Sprachen.* `http://www.heise.de/developer/artikel/Werkzeuge-fuer-domaenenspezifische-Sprachen-227190.html`. ([Online; abgerufen am 23.11.2014])

Voelter, M. (2013). Language and ide modularization and composition with mps. In *Generative and transformational techniques in software engineering iv* (S. 383–430). Springer.

Voelter, M., Benz, S., Dietrich, C., Engelmann, B., Helander, M., Kats, L. C., ... Wachsmuth, G. (2013). *Dsl engineering-designing, implementing and using domain-specific languages.* dslbook.org.

Voelter, M. & Pech, V. (2012). Language modularity with the mps language workbench. In *Software engineering (icse), 2012 34th international conference on* (S. 1449–1450).

Voelter, M., Ratiu, D., Kolb, B. & Schaetz, B. (2013). mbeddr: Instantiating a language workbench in the embedded software domain. *Automated Software Engineering*, *20* (3), 339–390.

Voelter, M. & Solomatov, K. (2010). Language modularization and composition with projectional language workbenches illustrated with mps. *Software Language Engineering, SLE*, 16.

Volter, M. (2011). From programming to modeling-and back again. *Software, IEEE*, *28* (6), 20–25.

Weiss, D. M. et al. (1999). Software product-line engineering: a family-based software development process.

Yin, R. K. (2010). *Qualitative research from start to finish.* Guilford Press.

Yin, R. K. (2014). *Case study research: Design and methods.* Sage publications.