Integration von bestehendem Sicherheitswissen in einen Software-Entwicklungsprozess

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Aleksander Dikanski

aus Bernau (bei Berlin)

Tag der mündlichen Prüfung:	07. November 2014
Erster Gutachter:	Prof. Dr. Sebastian Abeck
Zweiter Gutachter:	Prof. Dr. Jörn Müller-Quade

Inhaltsverzeichnis

1 Einleitung

1.1 Einführung in das Themengebiet

Trotz zahlreicher Entwicklungen im Bereich von Software-Sicherheit sind heutige Software-Systeme nur ungenügend gegenüber Bedrohungen und deren Schadensauswirkungen geschützt. Beleg hierfür sind zahlreiche Angriffe auf Software-Systeme, welche die Veröffentlichung von privaten Nutzerdaten und sensiblen Organisationsgeheimnissen umfassen [DPA11] [La11] [Sc10] [DPA11b]. Statistiken belegen, dass auch in Zukunft die Zahl der Angriffe auf Informationssysteme weiter zunehmen wird [Me11]. Dies wird durch den stärker werdenden Trend von vernetzten und verteilten Software-Systemen begünstigt, welche über Internettechnologien miteinander agieren und Daten austauschen. Durch die Kommunikation über unsichere Medien wird das Risiko eines Angriffs von Außenstehenden auf diese Systeme erhöht [VG02:2] [Sc01:16] [CN+05:3]. Allerdings wird eine solche Vernetzung und Integration von Software-Systemen immer mehr von Organisationen gefordert und gefördert, um unter anderem einen einfacheren Informationsaustausch zwischen den beteiligten Parteien zu ermöglichen [KB+07].

Organisationen sind weitestgehend damit beschäftigt, die Anzahl an technischen Schwachstellen in den eingesetzten Software-Systemen zu beheben. Der langfristige effektive Nutzen der Kombination aus Penetrationstest und Schwachstellenbehebung (engl. penetrate and patch) ist dabei zweifelhaft [Sc01] [Me11] [VG02:15ff]. Grund hierfür ist, dass durch ein solches reaktives Vorgehen nur die Symptome der Angriffe und selten die eigentlichen Ursachen behandelt werden [VG02]. Diese liegen seitens der bedrohten und angegriffenen Organisationen zumeist darin begründet, dass selten klar ist, welche Ressourcen der Software-Systeme zu schützen sind und auf welche Art und Weise dieser Schutz adäquat implementiert werden soll [Me11] [VG02].

Zudem existieren mit Gesetzen sowie industriellen Vorschriften und Standards zum Teil konkrete Vorgaben, welche Art von Ressourcen vor unberechtigtem Zugriff geschützt werden sollen. Das Bundesdatenschutzgesetz gibt beispielsweise vor, dass die Erhebung, Verarbeitung und Nutzung von personenbezogenen Daten ohne explizite Erlaubnis prinzipiell verboten ist [BDSG:§4 Abs (1)]. Im ISO-Standard 27001:2005 wiederum wird ein umfangreicher Katalog an Maßnahmen zur Gewährleistung von Informationssicherheit in Soft- und Hardware aufgezählt [ISO-27001]. Unternehmen investieren große Summen, um ihre IT-Infrastruktur so zu gestalten, dass sie entsprechende Zertifizierungen zur Erfüllung dieser Standards erhalten. Da die IT-Infrastruktur eines Unternehmens in Zeiten von verteilten und vernetzten Software-Systemen an neue Bedürfnisse angepasst werden muss [KB+07], müssen bei Anschaffung, Entwicklung oder Anpassung neuer oder existierender Informationssysteme diese Zertifizierungsvorschriften beachtet werden.

Zur Umsetzung von Schutzbedürfnissen in Software-Systemen existiert dabei eine Vielzahl an Technologien, Standards und Modellen. Grund hierfür ist, dass aus historischer Perspektive ein Bedarf nach Schutz vor unberechtigten Zugriff seit Einführung der ersten Software-Systeme bestand. Mit Aufkommen von neuen Angriffsmethoden wurden entsprechend immer neuere Maßnahmen zum Schutz von Software-Systemen entwickelt [WM05:3ff]. Dies hat zur Folge, dass heutzutage ein enormes Expertenwissen notwendig ist, um dieses vielfältige Sicherheitswissen korrekt anzuwenden [TJ+08]. Durch die Vielfalt des Sicherheitswissens kommt hinzu, dass zur Abwehr oder Abschwächung eines Angriffes mehrere Varianten an möglichen Umsetzungen existieren [SF+05] [CN+05]. Allerdings kann ohne Hintergrundwissen und Kontextbezug zumeist keine strukturierte Entscheidung bei der Auswahl getroffen werden [SF+05]. Ebenso ist eine mehrschichtige Lösung notwendig, um das Schutzbedürfnis von Software-Systemen umfassend zu erfüllen. Diese

Komplexität ist ein Grund für die Vernachlässigung von Sicherheitsfragestellungen bei der Entwicklung von Software-Systemen, wodurch Sicherheitsmaßnahmen erst nachträglich zu existierenden Anwendungen und Diensten hinzugefügt werden [CN+05] [MC04].

Um einen angemessenen Schutz von Software-Systemen und der durch sie verarbeitenden Ressourcen zu gewährleisten, ist es nach Meinung von Experten notwendig, Sicherheitseigenschaften eines Software-Systems während des gesamten Entwicklungsprozesses zu betrachten [Ec09:167ff] [MC04] [VG02] [Sc01]. Software-Sicherheit ist dabei eine funktionale Qualitätseigenschaft von Software-Systemen [IEEE-610.12-1990] in [FH06], durch welche ein Software-System nur Systemzustände annimmt, in welchen keine unautorisierte Informationsgewinnung oder -veränderung zulässig ist [Ec09:5]. Die durchgängige Berücksichtigung von Sicherheitseigenschaften in der Software-Entwicklung ermöglicht dabei eine verbesserte Integration von Sicherheitsfunktionalität und Fachfunktionalität [Ra02:44f]. Hierdurch lassen sich auf den Schutzbedarf eines Software-Systems angepasste Lösungen strukturiert entwickeln, wodurch das Risiko von Angriffen und Bedrohungen effektiv reduziert werden kann. Der Bedarf nach einer Evolution der Sicherheitsfunktionalität wird in dieser Herangehensweise nicht ausgeschlossen, jedoch wird diese Weiterentwicklung im Gegensatz zum oben angesprochenen reaktiven Vorgehen in einem strukturierten und methodischen Prozess durchgeführt.

Die durchgängige Betrachtung von Sicherheitseigenschaften von Software-Systemen in einem strukturierten Software-Entwicklungsprozess führt zu einer sicherheitsbasierten Software-Entwicklung (engl. security engineering [An08] [Ec09:167] [VG02]). Hierbei werden Software-Entwicklungsprozesse um Werkzeuge, Prozesse und Methoden ergänzt, mittels welchen in den Entwicklungsphasen sicherheitsrelevante Entwicklungsartefakte erzeugt werden können. Analog zur fachlichen Entwicklung ist dabei die Analyse von Sicherheitsanforderungen, der Entwurf von Sicherheitsmaßnahmen einer Sicherheitsarchitektur sowie die Implementierung von Sicherheitsmechanismen, das Testen auf Sicherheitslücken von Software-Systemen bzw. die Validierung der Sicherheitsanforderungen und die Absicherung der Ausführungsumgebung des Software-Systems relevant [An08] [MC04]. Das Ziel der Disziplin der sicherheitsbasierten Software-Entwicklung ist es somit, gegenüber Angriffen, Fehlern und Unfällen zuverlässige Software-Systeme zu entwickeln [An08:3].

Mit der Entwicklung von zunehmend komplexeren verteilten Systemen steht diese Disziplin vor der Herausforderung, mehrschichtige Sicherheitslösungen für diese heterogenen Software-Systeme strukturiert und effizient zu entwickeln. In der Strukturierung von fachlichen Software-Systemen setzen sich moderne Software-Paradigmen wie Dienstorientierung [KB+07] und Software-Produktlinien [PB+05] durch, deren zentraler Fokus die Wiederverwendung von Software-Entwicklungsartefakte und -Funktionalität zur Beherrschung der Komplexität der Systeme ist. Ein solcher Paradigmenwechsel hat für die durchgängige Entwicklung von Sicherheitslösungen noch nicht stattgefunden [Wi05], sondern wird derzeit nur für einzelne Phasen des Entwicklungsprozesses berücksichtigt [LB+02] [Jü05] [Ec09]. Hierdurch wird die Integration von Aktivitäten eines sicherheitsbasierten Entwicklungsvorgehens in eine fachliche Entwicklung erschwert.

Um das Ziel einer sicherheitsbasierten Software-Entwicklung zu erreichen, ist eine kohärente Methodik zur Entwicklung von Sicherheitsfunktionalität erforderlich, welche sich in das strukturierte Vorgehen beliebiger Entwicklungsprozesse integrieren lässt. Ein wesentlicher Beitrag hierzu liegt in der Wiederverwendung von bestehendem Sicherheitswissen in Form von sicherheitsspezifischen Software-Produkten, -Komponenten und -Rahmenwerken sowie technologieunabhängigen Sicherheitsmodellen. Hierzu ist es nötig, den Zusammenhang zwischen den abstrakten Sicherheitsmodellen und konkreten Sicherheitstechnologien eindeutig zu klären. Ebenso ist die Integration in den fachlichen Entwicklungsprozess zu definieren, welche das existierende Sicherheitswissen in ein klares

Verhältnis zum Architekturentwurf sowie zu den Sicherheitsanforderungen stellt. Dies ermöglicht eine Entscheidungshilfe bei der Entwicklung von Sicherheitslösungen für Software-Systeme.

1.2 Gegenstand dieser Arbeit

Die vorliegende Arbeit hat zum Ziel, Beiträge zur Umsetzung einer sicherheitsbasierten Software-Entwicklung zu liefern. Zu diesem Zweck wird insbesondere die Wiederverwendung und Integration von existierendem Sicherheitswissen in einem Software-Entwicklungsprozess angestrebt. Im Fokus der Untersuchung stehen dabei insbesondere die Analyse- und die Entwurfsphase eines Software-Entwicklungsprozesses. Im Folgenden werden daher die wesentlichen Aktivitäten dieser Phasen eines generischen sicherheitsbasierten Software-Entwicklungsvorgehens erläutert.

Erhebung von Sicherheitsanforderungen

Die Basis für die Implementierung von Sicherheitsfunktionen in einem Software-System ist die Erhebung und Spezifikation von Sicherheitsanforderungen [TJ+08]. Hierdurch wird die Sicherheitsfunktionalität, welche durch das Software-System benötigt wird, von vornherein bestimmt und einer nachgezogenen Implementierung von Sicherheitsfunktionalität (engl. penetrate and patch, [VG02:15]) vorgebeugt. Die Erhebung von Sicherheitsanforderungen erfordert im Gegensatz zur Erhebung von fachlichen Anforderungen die Berücksichtigung von unerwünschten Ereignissen und Zuständen, welchen im Software-System vorgebeugt werden soll. Hierzu werden sogenannte Risiko- bzw. Bedrohungsanalysen durchgeführt, durch welche diese unerwünschten Ereignisse ermittelt werden. Der Ablauf dieser Analysen lässt sich dabei prinzipiell in die folgenden Schritte unterteilen [WM05:23ff] [VG02:34ff] [TJ+08].

Zunächst werden die zu schützenden Ressourcen eines Software-Systems spezifiziert, indem die Ressourcen des Software-Systems identifiziert und deren Wert bestimmt wird. Dies führt zur Spezifikation von Schutzzielen, wie zum Beispiel Integrität und Vertraulichkeit, welche sicherheitsrelevante Qualitätseigenschaften der Ressourcen darstellen. Anschließend werden Bedrohungen bestimmt, deren Auswirkungen den Wert der Ressourcen reduzieren. Ebenso wird die Eintrittswahrscheinlichkeit sowie der durch die Auswirkungen verursachte Schaden der Bedrohungen ermittelt, wodurch das Bedrohungsrisiko spezifiziert wird. Abhängig vom Risiko der Bedrohungen und dem Wert einer Ressource werden anschließend geeignete Sicherheitsanforderungen spezifiziert. Diese stellen Ansprüche an das Software-System dar, das Risiko der Bedrohung zu reduzieren oder den Auswirkungen der Bedrohungen entgegenzuwirken. Damit verfolgen sie das Ziel, den Wert und die Schutzziele der Ressourcen zu erhalten.

Entwurf einer Sicherheitsarchitektur

Ausgehend von den festgelegten Sicherheitsanforderungen werden in der anschließenden Entwurfsphase geeignete Sicherheitsmaßnahmen abgeleitet und entsprechend modelliert. Analog zum Entwurf der fachlichen Funktionalität wird dabei das Abstraktionsniveau der Maßnahmen und Modelle iterativ verfeinert. Ausgehend von einer Sicherheitsarchitektur, in welcher die bereitgestellte Sicherheitsfunktionalität zusammengefasst wird, wird anschließend der Feinentwurf der einzelnen Sicherheitsmaßnahmen modelliert. Der Feinentwurf ist dabei Ausgangspunkt für die Umsetzung der Modelle in lauffähige Software-Systeme in der Implementierungsphase. Dabei werden die Maßnahmen iterativ verfeinert, wobei sich das Abstraktionsniveau der Sicherheitsmaßnahmen an dem Abstraktionsniveau der fachlichen Funktionalität orientiert.

Die Sicherheitsarchitektur beschreibt die Sicherheitsfunktionalität, welche in Form von Schnittstellen den Komponenten der fachlichen Architektur bereitgestellt wird. Hierbei wird eine Zentralisierung der Sicherheitsfunktionalität bevorzugt, um eine Trennung der Zuständigkeiten [Pa72] zwischen fach-

licher und sicherheitsrelevanter Funktionalität zu ermöglichen. Aufgrund der querschnittlichen Natur der Sicherheitsmaßnahmen, kann diese Trennung nicht immer konsequent durchgesetzt werden. In diesem Fall ist die Abhängigkeit zwischen fachlicher Funktionalität und den Diensten der Sicherheitsarchitektur zu minimieren.

Neben der Sicherheitsfunktionalität sind Sicherheitsrichtlinien ein weiterer wichtiger Bestandteil einer Sicherheitsarchitektur. Diese beschreiben dabei wesentliche Bedingungen zu einer Sicherheitsmaßnahme, welche durch die Sicherheitskomponenten verarbeitet und umgesetzt werden. Hierbei werden vor allem gesetzliche Rahmenbedingungen und Sicherheitsvorschriften der Organisation zum Schutz von Ressourcen auf realisierbare Regeln und Bedingungen abgebildet. Für die Umsetzung der Richtlinien eignen sich bekannte Sicherheitsmodelle, welche bewährte Strukturen zur Organisation der Richtlinien spezifizieren.

Abhängig von den in der Anforderungsanalyse spezifizierten Sicherheitsanforderungen müssen diese in den folgenden Phasen auf geeignete Entwurfsmodelle abgebildet werden, welche Maßnahmen zur Umsetzung der Anforderungen darstellen. Zur Spezifizierung von Sicherheitsmaßnahmen in der Analyse- und der Entwurfsphase gelten Sicherheitsmuster als annerkanntes Werkzeug, um strukturiert bekannte und erprobte Lösungen zu typischen und wiederkehrenden Sicherheitsproblemen einzusetzen [CN+05] [SF+05].

Auswahl geeigneter Sicherheitstechnologien

In der anschließenden Implementierungs- und Testphase werden die Sicherheitsmuster durch konkrete Technologien umgesetzt und gegenüber den Sicherheitsanforderungen validiert. Bei der Umsetzung von Sicherheitsfunktionalität ist es empfehlenswert, existierende Sicherheitsprotokolle und -standards einzusetzen sowie existierende Sicherheitsprodukte und -dienste zu verwenden, welche diese implementieren. Grund hierfür ist zum einen, dass die Funktionalität von existierenden Sicherheitsprodukten aufgrund der langjährigen Anwendung unter realen Bedingungen ausreichend getestet und somit stabil umgesetzt ist. Zum anderen wird durch den Einsatz von Sicherheitsstandards die Interoperabilität zwischen verschiedenen Sicherheitsprodukten bzw. fachlichen Software-Systemen gewährleistet. Werden Modelle des fachlichen sowie des Sicherheitsentwurfs direkt implementiert, so sollten Vorgaben zur Programmierung von fehlerfreiem Quellcode angewandt werden, um Schwachstellen und Verwundbarkeiten auf Quellcodeebene zu vermeiden [HL09].

Zum Testen von sicherheitsrelevanter Funktionalität können anschließend sogenannte Penetrationstests [Ec09:194] [An08:885] [VG02:38ff] angewendet werden, um Bedrohungen für Software-Systeme zu simulieren und so mögliche Schwachstellen zu finden und zu beheben. Statische Analysen wiederum helfen beim Auffinden von Verwundbarkeit im Quellcode, indem typische fehlerbehaftete Sprachkonstrukte identifiziert werden [MC04].

Resümee

Der beschriebene Ansatz zur Entwicklung von Sicherheitsfunktionalität für ein Software-System stellt ein generisches Vorgehen dar, welches sich an ein ebenfalls generisches Vorgehen zur Entwicklung von fachlicher Funktionalität orientiert [So07]. Durch diese Orientierung an die funktionale Entwicklung wird die Integration der Entwicklung von Sicherheitsfunktionalität in das Entwicklungsvorgehen verfolgt. Der Fokus dieser Arbeit richtet sich auf die Unterstützung der Analysephase sowie der Entwurfsphase eines sicherheitsbasierten Entwicklungsvorgehens, weshalb diese Phasen beschrieben wurden. Für die Beiträge dieser Arbeit zur Entwurfsphase sind dabei die existierenden Sicherheitsstandards und -produkte relevant, weshalb diese in der Implementierungsphase ebenfalls beschrieben wurden. Das generische Entwicklungsvorgehen steht repräsentativ für zahlreiche spezifischere Ansätze, bei welchen die Integration von Sicherheitsfunktionalität problematisch ist. Die

Problemstellungen, welche die Beiträge der Arbeit motivieren, werden im folgenden Abschnitt besprochen.

1.3 Problemstellungen

Die durchgängige Berücksichtigung von Sicherheitseigenschaften eines Software-Systems mittels der Integration von Methoden zur sicherheitsbasierten Software-Entwicklung in einen Entwicklungsprozess stellt ein grundlegendes Software-technisches Problem dar [Ec09:168]. Sicherheit beschreibt zum einen eine funktionale Anforderung an ein Software-Systems und gilt zum anderen als eine qualitätsbezogene Software-Eigenschaft [So07] [FH06]. Aufgrund dessen liegt Sicherheitsfunktionalität meist querschnittlich zur fachlichen Funktionalität, weshalb für jede fachliche Funktion deren Sicherheitseigenschaften berücksichtigt werden müssen. Eine Betrachtung von querschnittlicher Funktionalität ist insbesondere für Sicherheit unumgänglich. Eine zumindest logische Trennung von Fachfunktionalität und querschnittlicher Funktionalität in der Entwicklung ist jedoch wünschenswert [Pa72]. Auch in der späteren Implementierung ist eine solche Trennung aufgrund von Wartbarkeitsanforderungen und Verwaltung von Sicherheitsfunktionalität vorteilhaft.

Zudem steht die Entwicklung der fachlichen Funktionalität im Hauptfokus der Software-Entwicklung, weshalb Sicherheitsfunktionalität aus Kosten- und Zeitgründen häufig vernachlässigt wird. Grund hierfür ist, dass es selbst für erfahrene Software-Architekten und -Entwickler schwierig ist, Sicherheitsfunktionalität in einem strukturierten und methodischen Vorgehen zu entwickeln. Denn aufgrund der Komplexität der Sicherheitsdomäne wird zusätzlich zu Kenntnissen über die fachliche Domäne, Wissen über die Konzepte der Sicherheits-Domäne benötigt. Im Folgenden werden Problemstellungen vorgestellt, welche sich aus der Vernachlässigung von bestehendem Sicherheitswissen in einem generischen sicherheitsbasierten Entwicklungsvorgehen ergeben.

Problemstellung PS1: Strukturierte Entwicklung von Sicherheitsfunktionalität

Die Beschreibung des generischen Entwicklungsvorgehens im vorherigen Abschnitt stellt eine informelle Beschreibung der wichtigsten Aktivitäten eines sicherheitsbasierten Entwicklungsvorgehens dar. Zur strukturierten Entwicklung von Sicherheitsfunktionalität bedarf es einer präziseren Spezifikation von Aktivitäten, welche in den jeweiligen Phasen durchgeführt werden, sowie von Entwicklungsartefakten, welche das Ergebnis einer Entwicklungsphase darstellen. Die Aktivitäten beschreiben dabei die Schritte, welche zur Konstruktion von sicherheitsrelevanten Entwicklungsartefakten für die einzelnen Phasen durchgeführt werden müssen. Die Entwicklungsartefakte stellen wiederum die Eingabe für darauffolgende Phasen dar.

Ein sicherheitsbasiertes Entwicklungsvorgehen sollte dabei bereits existierende heterogene Ansätze zur Entwicklung von Sicherheitsfunktionalität berücksichtigen und in das Vorgehen einordnen. Bei diesen Ansätzen steht zumeist die Betrachtung von einzelnen Entwicklungsphasen im Vordergrund, welche durch bestimmte Modellierungswerkzeuge, zum Beispiel zur Modellierung von Sicherheitsanforderungen [SO05] [BF+08] oder -maßnahmen [SF+05] erweitert werden. Eine Einordnung der verschiedenen Ansätze in den Entwicklungsprozess findet jedoch in den existierenden Ansätzen nicht statt, wodurch der Austausch zwischen den Phasen und den Modellierungswerkzeugen erschwert wird. Es existieren zwar bereits erste Ansätze, welche eine solche Einordnung durch die Einführung eines Domänenmodells für sicherheitsbasierte Software-Entwicklung anvisieren [FH06] [MF+10], diese sind jedoch ebenfalls nur auf einzelne Phasen beschränkt und somit nicht für eine durchgehende Betrachtung im Entwicklungsprozess geeignet.

Weiter ist zu beachten, dass das sicherheitsbasierte Entwicklungsvorgehen unabhängig von einem konkreten Software-Entwicklungsprozess spezifiziert wird, wodurch der flexible Einsatz in

verschiedenen Entwicklungsprozessen ermöglicht wird. Allerdings ist dabei zu beachten, dass das sicherheitsbasierte Entwicklungsvorgehen nicht starr ist, sondern sich an die Gegebenheiten eines fachlichen Software-Entwicklungsprozesses anpassen lässt. Hierzu müssen die Abhängigkeiten zu einem fachlichen Entwicklungsprozess im Detail spezifiziert sowie die Interaktionen zwischen den Aktivitäten definiert werden.

Ebenso darf das Vorgehen nicht auf die Entwicklung von Sicherheitsfunktionalität für bestimmte fachliche Domänen eingeschränkt sein. Vielmehr soll im allgemeinen Fall eine Vielzahl an fachlichen Domänen unterstützt werden. Sollte dennoch eine Fokussierung auf die Entwicklung von Sicherheitsfunktionalität einer spezifischen fachlichen Domäne benötigt werden, so sollte dies durch entsprechende Modifikationen durch das Entwicklungsvorgehen unterstützt werden.

Problemstellung PS2: Berücksichtigung von existierendem Sicherheitswissen

Um eine effiziente sicherheitsbasierte Software-Entwicklung durchzuführen ist es vorteilhaft, existierendes Sicherheitswissen zur Lösung von Sicherheitsproblemen einzusetzen. Bei Sicherheitswissen handelt es sich um bewährte Lösungsverfahren, welche zu einem realen Sicherheitsproblem dokumentiert wurden. Der Einsatz von existierendem Sicherheitswissen führt im Gegensatz zu Eigenentwicklungen zu stabileren Sicherheitslösungen für ein Software-System. Sicherheitswissen in Form von abstrakten Sicherheitsmodellen unterstützt dabei die Spezifizierung von Sicherheitsrichtlinien, während Sicherheitsmuster bei der Modellierung von sicherheitsrelevanter Funktionalität helfen. Technische Sicherheitsstandards ermöglichen zudem eine Interoperabilität zwischen und den Austausch von zugrundeliegenden Sicherheitsprodukten.

Ferner ist der Einsatz von bestehenden Sicherheitsprodukten zur Implementierung von Sicherheitsfunktionalität eine wesentliche Grundvoraussetzung für die Entwicklung von komplexen Software-Systemen [DE+09]. Unter Sicherheitsprodukten werden hierbei bestehende Komponenten, Dienste oder Anwendungen verstanden, welche sicherheitsrelevante Funktionalität komplett oder in Teilen bereitstellen. Bei der Implementierung von Sicherheitsfunktionalität sollte daher ebenfalls auf bereits in dieser Form bestehende Artefakte zurückgegriffen werden. Diese müssen gegebenenfalls erst beschafft werden, haben aber den Vorteil, dass die geforderte Funktionalität durch den Einsatz in anderen Software-Systemen bereits lang andauernde Betriebs- und Entwicklungszeiten besitzen und somit ausreichend getestet sind. Die hierbei erreichte Reife und Stabilität kann selten ohne großen Aufwand durch eine Eigenimplementierung umgesetzt werden.

Des Weiteren ist zu beachten, dass Unternehmen bereits große Investitionen in Sicherheitstechnologien getätigt haben, welche es bei der Entwicklung neuer Software-Anwendungen zu berücksichtigen gilt. Vor allem in der Dienstorientierung ist die Verwendung von Sicherheitsdiensten eines internen oder externen Dienstleisters zur Absicherung von fachlichen Diensten wesentlicher Bestandteil des Paradigmas [KB+07]. Diese bestehenden Dienste und Technologien bildet eine IT-Sicherheitsinfrastruktur und hat einen Einfluss auf die Entwicklung von sicherer Software, da diese an die Technologien angepasst werden muss, um ohne größere Anpassungskosten in der Infrastruktur betrieben werden zu können. Wird diese nicht betrachtet, so besteht die Gefahr, dass Sicherheitsmaßnahmen modelliert werden, zu welchen entweder nicht die geeigneten Technologien zur Verfügung stehen oder die Maßnahmen redundant modelliert werden. In beiden Fällen besteht ein Effizienzverlust in der Entwicklung der Software.

In bisherigen Ansätzen wird dieser Punkt oft vernachlässigt, da von einer Neuentwicklung eines Software-Systems ausgegangen wird. Zudem wird die Umsetzung von modellierten Sicherheitsmaßnahmen nicht ausreichend berücksichtigt oder die Sicherheitsmaßnahmen werden mittels eigener Implementierungen umgesetzt. Ein flexibles und effizientes Vorgehen zur sicherheitsbasierten Software-Entwicklung sollte das bestehende Sicherheitswissen berücksichtigen und Entwicklern in

angemessener Form zur Verfügung stellen. Dabei ist es nötig, das Sicherheitswissen zu kategorisieren und in den Entwicklungsprozess einzuordnen.

Problemstellung PS3: Bereitstellung von Handlungsalternativen

Das entwickelte und dokumentierte Sicherheitswissen beeinflusst die Entwicklung von sicherer Software in jeder Phase eines Entwicklungsprozesses. Aufgrund der unstrukturierten Fülle an Sicherheitswissen haben Entwickler Probleme, geeignete Entwicklungsartefakte für die jeweiligen Phasen eines Entwicklungsprozesses zu spezifizieren, da sie in der Entwicklung von sicherer Software unerfahren sind. Hierbei handelt es sich zum Beispiel um die zweckmäßige Formulierung und Abstraktionsstufe von passenden Sicherheitsanforderungen in der Analysephase sowie der Modellierung von geeigneten Sicherheitsmaßnahmen in der Entwurfsphase.

Die separate Untersuchung von einzelnen Entwicklungsphasen ergibt, dass sich in jeder Phase unterschiedliche Möglichkeiten zur Spezifizierung dieser Artefakte ergeben. So existieren meist verschiedene alternative Sicherheitsanforderungen zu bestimmten Bedrohungen. Ebenfalls können bestimmte Sicherheitsanforderungen gegebenenfalls abhängig von weiteren Sicherheitsanforderungen sein oder diese ausschließen. Auch für modellierte Sicherheitsmaßnahmen existieren verschiedene Varianten, welche unterschiedliche Vor- und Nachteile haben. So existieren für die abstrakte Sicherheitsmaßnahme „Zugriffskontrolle“ verschiedene Ausprägungen wie zum Beispiel rollen- oder attributbasierte Zugriffskontrolle [SC+96] [YT+05], welche das Problem des unautorisierten Zugriffs behandeln, sich in der jeweiligen Umsetzung jedoch unterscheiden.

Um die Auswahl passender Artefakte und die Umsetzung von geeigneten Sicherheitsmaßnahmen zu ermöglichen, ist eine Dokumentation notwendig, welche die Zusammenhänge zwischen den alternativen Entwicklungsartefakten aufzeigt. Bei der Modellierung von alternativen Sicherheitsmaßnahmen existieren mit Sicherheitsmustern bereits erste Ansätze, um die Vielfalt an bewährten Methoden mit ihren Vor- und Nachteilen sowie Kombinationsmöglichkeiten zu dokumentieren und aufzuzeigen [SF+05]. Allerdings werden hierbei die Kombinationsmöglichkeiten nur informal angedeutet, wodurch konkrete Auswirkungen bei der Kombination nicht berücksichtigt werden. Ebenfalls sind solche Ansätze für die weiteren Phasen wie zum Beispeil die Anforderungsanalyse oder der Implementierung, kaum ausgeprägt, obwohl auch in diesen Phasen eine Unterstützung bei der Auswahl notwendig ist.

Problemstellung PS4: Durchgängigkeit der Entwicklungsphasen

Neben der Betrachtung der Variationsmöglichkeiten von sicherheitsrelevanten Entwicklungsartefakten innerhalb einzelner Entwicklungsphasen ist die nachvollziehbare Abbildung der Entwicklungsartefakte zwischen den Phasen problematisch. Sicherheitsfunktionalität sollte in einem methodischen Vorgehen iterativ abgeleitet und verfeinert werden, so dass auch die Anforderungen der fachlichen Funktionalität eines Software-Systems sowie gegebenenfalls durch Gesetze vorgeschriebene Anforderungen berücksichtigt werden [VG02]. Hierdurch wird ermöglicht, dass nur die benötigten Sicherheitsmaßnahmen in einem ausreichenden Maße umgesetzt werden.

Um dies zu erreichen, ist die durchgängige Abbildung von Sicherheitsanforderungen auf entpsrechende Maßnahmen und anschließend auf zugehörige Technologien notwendig. Dies bedarf jedoch einer Unterstützung, da diese Entwicklungsartefakte jeweils in einer n:m-Beziehung stehen. So existiert zu einer Sicherheitsanforderung beispielsweise mehr als eine passende Maßnahme, welche zur Umsetzung der Anforderung kombiniert werden müssen. Entsprechend kann eine Maßnahme bei der Umsetzung von mehreren Sicherheitsanforderungen eingesetzt werden. So kann zum Beispiel die Anforderung „Schutz vor unautorisiertem Zugriff“ einer Webseite, durch Zugriffskontrollmaßnahmen umgesetzt werden. Dies führt zu Abhängigkeiten zu weiteren Sicherheitsmaßnahmen, unter anderem zu Identifikations- und Authentifikationsmaßnahmen. Des Weiteren kann die Zugriffskontroll-

maßnahme zum einen bei der Umsetzung von weiteren Sicherheitsanforderungen eingesetzt werden und zum anderen auf unterschiedliche Art, zum Beispiel durch rollen- oder attributbasierte Zugriffskontrolle, umgesetzt werden.

Eine ähnliche Unterstützung ist in der Implementierungsphase bei der Ableitung von konkreten Sicherheitstechnologien aus den modellierten Maßnahmen notwendig, da es zu jeder Maßnahme mehr als eine Technologie gibt, welche diese unterstützt. So existiert ein breites Spektrum an Identifikations- und Authentifikationstechnologien, wie zum Beispiel Benutzername/Passwort, Smartcards und biometrische Merkmale, welche abhängig von den Anforderungen und modellierten Sicherheitsmaßnahmen des Software-Systems auszuwählen sind.

Jedoch sind diese Zusammenhänge, welche Entwickler bei der Auswahl und Abbildung von alternativen Entwurfsentscheidungen behilflich sind, selten explizit dokumentiert und werden in derzeitigen Ansätzen nicht ausreichend berücksichtigt. Hier steht zumeist die Betrachtung von einzelnen Entwicklungsphasen im Vordergrund, indem bestimmte Modellierungswerkzeuge erweitert werden, um zum Beispiel Sicherheitsanforderungen [SO05] [BF+08] oder -maßnahmen [SF+05] zu modellieren. Die Abbildung der Entwicklungsartefakte zwischen den Phasen wird jedoch zumeist vernachlässigt.

Problemstellung PS5: Mehrdeutige Interpretation von Sicherheitsbegriffen

Die Integration von Sicherheit in den Entwicklungsprozess wird häufig durch unterschiedliche Auffassungen über die Konzepten und Begriffen der Sicherheitsdomäne durch die am Prozess beteiligten Personen erschwert. Im Laufe der Jahre hat sich eine unüberschaubare Menge an Sicherheitswissen angesammelt, welches sowohl konkrete Technologien und Standards als auch konzeptionelle Modelle umfasst. Dieses Wissen zu beherrschen und anzuwenden, d.h. die abstrakten Begriffe sowie die konkreten Technologien einzuordnen und in Bezug zu setzen, ist ein schwieriges Unterfangen, welches nur selten durch entsprechende Werkzeuge unterstützt wird. Jedoch ist das Verständnis von Sicherheitskonzepten eine wesentliche Grundlage, zur zielorientierten Entwicklung von sicheren Software-Systemen.

Ein solches Wissen ist zudem meist nur Sicherheitsexperten geläufig und nur implizit vorhanden, wodurch es häufig aufgrund von Ambiguitäten zu Missverständnissen in der Kommunikation mit fachlichen Entwicklern kommt. So werden zum Beispiel oftmals Sicherheitsanforderungen in der Analysephase durch konkrete Maßnahmen beschrieben. Hierbei wird eine ungewünschte Abhängigkeit zwischen der Anforderungsspezifikation, welche den Einsatz einer Maßnahme erfordert, und der Entwurfs- bzw. Implementierungsentscheidung, welche die Umsetzung einer Maßnahme spezifiziert, erzeugt [HL+08].

Durch die unterschiedlichen Definitionen wird zudem eine Klassifikation der bestehenden Ansätze zur sicherheitsbasierten Software-Entwicklung verhindert. Dies beeinträchtigt deren Einordnung in spezifische Entwicklungsphasen und erschwert deren kombinierte Anwendung. Es existieren zwar bereits erste Ansätze, welche eine solche Einordnung durch die Einführung eines Domänenmodells ermöglichen [FH06] [MF+10], diese sind jedoch ebenfalls nur auf einzelne Phasen beschränkt und somit nicht für eine durchgehende Betrachtung im Entwicklungsprozess geeignet.

1.4 Zielsetzung und Beiträge dieser Arbeit

Die im vorherigen Abschnitt beschriebene Auswahl an Problemstellungen fokussiert zum einen die Integration von bestehendem Sicherheitswissen in ein sicherheitsbasiertes Software-Entwicklungsvorgehen. Es ist deutlich zu sehen, dass die Berücksichtigung und der Einbezug von existierendem Sicherheitswissen vorteilhaft für die strukturierte Entwicklung von Sicherheitsfunktionalität ist. Zum

anderen wird die Notwendigkeit der Spezifikation einer durchgängigen strukturierten Methode zur Entwicklung von sicheren Software-Systemen diskutiert, welche das Sicherheitswissen berücksichtigt. Diese Anforderungen sind Ausgangspunkt und Motivation für die vorliegende Arbeit.

Ziel dieser Arbeit ist es daher, eine Methode zur Entwicklung von sicherheitsrelevanter Funktionalität in Software-Systemen zu entwickeln, welche bestehendes Sicherheitswissen in den Entwicklungsprozess integriert und entsprechende Unterstützung zur Auswahl von Entwicklungsartefakten bereitstellt.

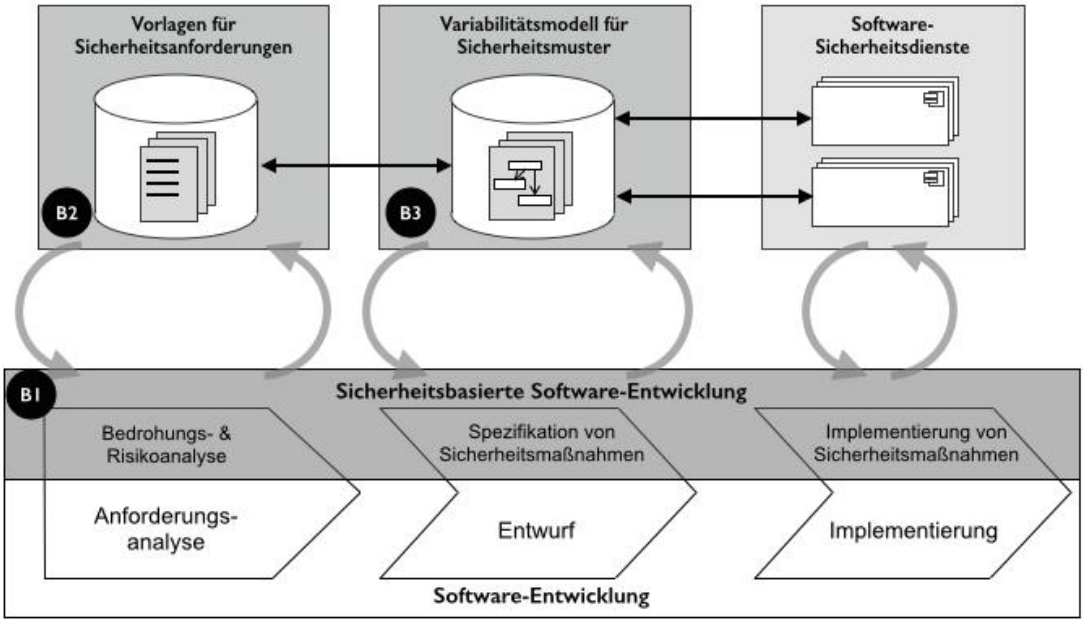

Abbildung 1: Überblick und Zusammenhang der Beiträge der Arbeit

Beitrag B1: Sicherheitsbasiertes Software-Entwicklungsvorgehen

Zentraler Beitrag dieser Arbeit ist die Spezifikation eines Vorgehens zur strukturierten Entwicklung von Sicherheitsfunktionalität in Software-Systemen. Das Vorgehen fokussiert dabei den strukturierten Einsatz von bestehendem Sicherheitswissen in einen Software-Entwicklungsprozess und behandelt somit die Problemstellungen PS1 und PS2. Ziel des Vorgehens ist es dabei, die wesentlichen Grundbausteine für Aktivitäten der sicherheitsbasierten Software-Entwicklung aufzuzeigen.

Für die Verwendung von bestehendem Sicherheitswissen werden zusätzliche Aktivitäten definiert, mittels welcher bestehendes Sicherheitswissen für den Einsatz in dem Entwicklungsvorgehen aufbereitet wird. Auf diese Weise stellt der erste Beitrag den Rahmen für die weiteren Beiträge der Arbeit dar. Zudem wird der Einsatz des aufbereiteten Sicherheitswissens zur Spezifikation von Sicherheitsanforderungen in einer Anforderungsanalyse sowie für die Modellierung von Sicherheitsmaßnahmen in einer Entwurfsphase beleuchtet. Zu diesem Zweck baut das vorgestellte Vorgehen auf bestehende Ansätze zur sicherheitsbasierten Software-Entwicklung auf und ordnet diese in das Vorgehen ein.

Bei der Spezifikation der Aktivitäten zur Entwicklung von Sicherheitsfunktionalität wird dabei die Unabhängigkeit von einem spezifischen Software-Entwicklungsprozess berücksichtigt. Hierdurch wird der Einsatz in verschiedenen Entwicklungsprozessmodellen unterstützt. Die spezifizierten Aktivitäten dienen dabei als generische und flexible Grundlage, wodurch eine Adaption an spezifische Prozessmodelle unterstützt wird. Hierzu werden wesentliche Assoziationen zu einem fachlichen Entwicklungsprozess aufgezeigt.

Beitrag B2: Vorlagen für Sicherheitsanforderungen

Zur Umsetzung von Sicherheitsmaßnahmen sind zunächst die Sicherheitsanforderungen eines Software-Systems zu spezifizieren. Im Kontext des sicherheitsbasierten Software-Entwicklungsvorgehens im Beitrag B1 beschäftigt sich der zweite Beitrag im Detail mit der Aufbereitung von Sicherheitswissen zur strukturierten Erhebung von Sicherheitsanforderungen. Die zur Spezifikation von Sicherheitsanforderungen notwendigen Konzepte werden zu diesem Zweck in Form von Vorlagen wiederverwendbar aufbereitet. Die Vorlagen beschreiben somit bekannte und bewährte Sicherheitsanforderungen unabhängig von einer spezifischen Fachdomäne, wodurch dieser Beitrag ebenfalls die Problemstellung PS2 behandelt.

Die zur Umsetzung der Vorlagen notwendigen Konzepte werden zunächst in einem Domänenmodell spezifiziert und die Beziehungen zwischen den Konzepten definiert. Somit befasst sich der zweite Beitrag zudem mit der Problemstellung PS5. Auf Grundlage des Domänenmodells wird ein Vorlagenkatalog beschrieben, welcher als Ablagestruktur für Sicherheitsanforderungsvorlagen dient. Dieser wird in einem spezifischen Software-Entwicklungsprozess eingesetzt, um aus den Vorlagen konkrete Sicherheitsanforderungen für zu schützende fachliche Ressourcen zu spezifizieren. Hierzu wird in dem Katalog eine Unterstützung bei der Auswahl von Vorlagen gemäß dem Kontext der untersuchten fachlichen Domäne sowie der zu schützenden Ressourcen bereitgestellt. Dies entspricht der Behandlung von Problemstellung PS3 in Bezug auf die Anforderungsanalyse.

Abschließend wird die Überführung von Sicherheitsanforderungen in abstrakte Sicherheitsmaßnahmen der Entwurfsphase aufgezeigt. Hierdurch wird die konzeptionelle Trennung zwischen der Analyse- und Entwurfsphase berücksichtigt, jedoch eine Verknüpfung der Artefakte durchgeführt, so dass eine effiziente Abbildung zwischen den Phasen ermöglicht wird. Somit wird in diesem Beitrag auch die Problemstellung PS4 behandelt. Die Verknüpfung stellt den Ausgangspunkt für den folgenden Beitrag B3 dar.

Beitrag B3: Variabilitätsmodell für Sicherheitsmuster

Ausgehend von den spezifizierten Sicherheitsanforderungen werden zu deren Umsetzung in der Entwurfsphase Modelle für Sicherheitsmaßnahmen entwickelt. Das im Beitrag B1 vorgesehene Entwicklungsvorgehen sieht hierfür ebenfalls den Einsatz von bestehendem Sicherheitswissen vor, um eine effiziente Entwicklung von stabilen Sicherheitslösungen zu ermöglichen. Im dritten Beitrag wird daher die strukturierte und iterative Ableitung von Sicherheitsmaßnahmen für Software-Systeme auf Basis von bestehenden Lösungsverfahren behandelt. Somit wird die Problemstellung PS2 im Kontext der Entwurfsphase des sicherheitsbasierten Entwicklungsvorgehens tiefergehend behandelt.

Hierzu wird der Ansatz von Sicherheitsmustern [SF+05] eingesetzt, um wiederverwendbare Sicherheitsmaßnahmen zu beschreiben. Der Ansatz wird um ein Variabilitätsmodell erweitert, mittels welchem es ermöglicht wird, bestehende Sicherheitsmaßnahmen anhand ihres Abstraktionsgrades zunächst zu kategorisieren. Dies ermöglicht den Einsatz von Sicherheitsmustern in den entsprechenden abstrakten fachlichen Entwurfsmodellen. Zudem lassen sich Abhängigkeiten sowie Verfeinerungsbeziehungen zwischen verschiedenen Sicherheitsmaßnahmen spezifizieren. Insbesondere lassen sich bei der Verfeinerung verschiedene alternative Umsetzungsmöglichkeiten zu einer Sicherheitsmaßnahme spezifizieren sowie zugehörige Handlungsempfehlungen zur Auswahl angeben, wodurch die Problemstellung PS3 adressiert wird.

Die Aufbereitung von Sicherheitsmaßnahmen mittels des Variabilitätsmodells wird dabei durch die Berücksichtigung von existierenden Sicherheitstechnologien eingeschränkt. Somit wird ein Middle-Out-Vorgehen zur Aufbereitung des Sicherheitswissens angewendet, welches abstrakte Sicherheitskonzepte mit konkreten Sicherheitstechnologien in Verbindung setzt. Durch die Einschränkung wird zum einen die Komplexität der betrachteten Sicherheitsfunktionalität reduziert und zum anderen die

Durchgängigkeit der Abbildung von Entwurfsmodellen auf bestehende Sicherheitstechnologien unterstützt. Der Beitrag befasst sich somit ebenfalls mit der Problemstellung PS4.

Demonstration der Tragfähigkeit

Die in dieser Arbeit vorgestellten konzeptionellen Beiträge wurden in zwei Software-Entwicklungsprojekten aus unterschiedlichen fachlichen Domänen angewendet. Die Praktikabilität sowie die Effektivität der Beiträge wird dabei anhand deren Einsatzes zur strukturierten Entwicklung der Sicherheitsfunktionalität der zu entwickelnden Software-Systeme demonstriert. In der anschließenden Bewertung der erzielten Ergebnisse wird die Realisierung der Zielsetzung der Beiträge im praktischen Einsatz evaluiert.

Im Rahmen des Projektes „KITCampusGuide“ (KCG) wurde in der Forschungsgruppe Cooperation & Management (C&M, Prof. Abeck) am Karlsruher Institut für Technologie (KIT) ein Web-basiertes Assistenzsystem entwickelt, welches die alltäglichen Tagesabläufe von Studierenden, Mitarbeitern und Gästen des KIT unterstützen soll. Insbesondere wurden hier Informationen über den Kontext und der Umwelt der Benutzer eingesetzt, um die assistierende Funktionalität bereitzustellen. Als Beispiel für die Tragfähigkeit der Beiträge dieser Arbeit wird die Kartenfunktionalität des KCG fokussiert. Diese Funktion ermöglicht den auf den verschiedenen Campus des KIT aktiven Personen unter anderem eine Suche nach interessanten Orten oder Personen (engl. points of interest, POI) und zeigt die Suchergebnisse in einer kartenbasierten Darstellung an. Das Ziel dieses Projektes ist es, in Anlehnung an ähnliche erfolgreiche Projekte, wie zum Beispiel Google Maps, Personen des KIT einen Basisdienst zur Navigation über den Campus zur Verfügung zu stellen, welcher in weiteren Diensten genutzt werden kann.

Zur Implementierung der Anwendung sind aus sicherheitstechnischer Sicht die Gegebenheiten am KIT zu beachten. Beispielsweise müssen, da das KIT eine öffentliche Einrichtung darstellt, die bestehenden rechtlichen Rahmenbedingungen zum Schutz von privaten Daten berücksichtigt werden. Zudem wird am KIT bereits eine Sicherheitsinfrastruktur eingesetzt, in welche die KCG-Anwendung integriert werden muss. Zur Berücksichtigung dieses Sachverhaltes wird das in dieser Arbeit beschriebene sicherheitsbasierte Entwicklungsvorgehen eingesetzt.

Im Rahmen des EU-Projektes „OpenIoT“, an welchem unter anderem das Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung (IOSB) und die Forschungsgruppe C&M beteiligt sind, wird eine Middleware zur Unterstützung von Anwendungen entwickelt, welche dem Paradigma des Internets-der-Dinge (engl. Internet of Things, IoT) folgen. Die Middleware stellt dabei betriebene Dienste zur Erfassung, Verarbeitung und Bereitstellung von heterogenen Sensordaten bereit, welche in Anwendungen integriert werden können. In der Forschungsgruppe C&M wurde eine prototypische Anwendung entwickelt, welche auf der OpenIoT-Middleware aufbaut. Diese verteilte „SmartMeetings”-Anwendung ermöglicht es Studierenden des KIT, freie Arbeitsplätze für Gruppen- oder Einzelarbeiten auf dem KIT-Campus zu suchen und diese für eine gewisse Zeitperiode zu reservieren.

Die über die Arbeitsplätze verfügbaren Informationen werden dabei nicht zentral auf einem Server verwaltet, sondern werden durch die jeweiligen Räume und den darin befindlichen Gegenständen autonom verwaltet. Hierzu wurde das IoT-Paradigma eingesetzt, bei welchem Gegenstände der realen Welt mit Sensorik sowie gegebenenfalls Aktorik ausgestattet werden und mittels einer Internetverbindung ihre Meta-Informationen in die OpenIoT-Middleware veröffentlichen können. Die Arbeitsplatzsuche wurde in die zuvor entwickelte KITCampusGuide-Anwendung integriert, so dass die Suche nach POIs die freien Arbeitsplätze und deren Reservierung mit einschließt. Als Ergänzung wurde das Software-System als mobile Anwendung für die Android-Plattform entwickelt.

Der Einsatz des Paradigmas des Internets der Dinge ist mit zahlreichen Sicherheitsbedenken verbunden. Aufgrund der Sammlung von zahlreichen, teilweise personenbezogenen Sensordaten, welche die Analyse von Nutzerverhalten zulassen und somit deren Privatsphäre verletzen, sind Sicherheitsvorkehrungen unabdingbar. In diesem Zusammenhang werden die Sicherheitsanforderungen sowie passende Sicherheitsfunktionalität bezüglich Authentifizierung und Autorisierung für das SmartMeetings-System unter Einsatz der Beiträge dieser Arbeit entwickelt.

1.5 Prämissen der Arbeit

Angesichts der Tatsache, dass der Themenkomplex der sicherheitsbasierten Software-Entwicklung umfangreich ist und noch zahlreiche Forschungsfragestellungen offen sind [Ec09:168], stellt die Entwicklung von sicheren Software-Systemen ein komplexes Vorhaben dar. Im Rahmen der vorliegenden Arbeit wurde die Integration von bestehendem Sicherheitswissen in ein durchgängiges Software-Entwicklungsvorgehen untersucht. Zur Erstellung der Beiträge wurden hierfür Prämissen gesetzt, welche im Folgenden erläutert werden.

Prämisse P1: Betrachtung von Software-Sicherheit

Der effektive Schutz von Software-Systemen sowie den verarbeitenden Ressourcen umfasst eine Vielzahl von mehrschichtigen Sicherheitsmaßnahmen. Im Rahmen dieser Arbeit wird die Entwicklung von Sicherheitsmaßnahmen betrachtet, welche den direkten Schutz von Software-Systemen umfassen. Hierdurch werden zahlreiche Faktoren zunächst nicht berücksichtigt, welche jedoch ebenso essentiell zur Umsetzung von Sicherheitsanforderungen sind. Dazu gehört die Entwicklung von physischen Schutzmaßnahmen, welche in realen Szenarien sicherlich berücksichtigt werden müssen. Grund für die Vernachlässigung ist dabei, dass diese Maßnahmen einerseits in kritischen Fällen als erstes eingeführt werden und dass andererseits aufgrund der Vernetzung von Software-Systemen die virtuellen Schutzmaßnahmen eine höhere Priorität einnehmen.

Der Hauptfokus dieser Arbeit liegt in der Absicherung von verteilten, Internet-basierten Software-Systemen, welche heutzutage den Großteil der Informationssysteme ausmachen. Diese Art von Software-Systemen werden in IT-Infrastrukturen betrieben, welche ebenfalls geschützt werden müssen, um einen vollständigen Schutz eines Software-Systems zu gewährleisten. Hierunter fallen Sicherheitsmaßnahmen, welche sich der Netzsicherheit widmen, wie zum Beispiel Firewalls, Authentifzierungssysteme für drahtlose Netze, etc., sowie die Absicherung der Ausführungsumgebung ermöglichen, wie zum Beispiel von durch den Einsatz von sicheren Betriebssystemen und Anwendungsservern. Solche Sicherheitsmaßnahmen, welche nicht die Ebene des Software-Systems betreffen, sondern auf den zugrundeliegenden IT-Infrastrukturen operieren, sind für die umfassende Betrachtung einer sicheren Anwendung ebenfalls notwendig. Die umfassende Betrachtung in dieser Arbeit wird jedoch ausgeschlossen, da sie den Rahmen der Arbeit überschreitet.

Zum anderen werden auch Schutzmaßnahmen außer Acht gelassen, welche den menschlichen Nutzer von Software-Systemen betreffen. Der menschliche Benutzer eines Software-Systems gilt nach Expertenaussagen nach wie vor als höchstes Sicherheitsrisiko. Methoden wie Social Engineering [Sc01:266ff], bei denen Angreifer darauf abzielen, in einer sozialen Interaktion mit autorisierten Benutzern eines Software-Systems Zugangsdaten in Erfahrung zu bringen, können nur schwer durch Software-basierte Sicherheitsmaßnahmen verhindert werden. Die in diesem Fall angebrachten Schulungsmaßnahmen für Benutzer werden in dieser Arbeit somit nicht berücksichtigt.

Prämisse P2: Fokussierung auf Analyse und Entwurf von Sicherheitsmaßnahmen

Eine weitere Einschränkung betrifft den Umfang der konkret betrachteten Software-Sicherheitsmaßnahmen. So ist eine wesentliche Prämisse des in dieser Arbeit beschriebenen Ansatzes

die Verwendung von existierenden Sicherheitskomponenten und -diensten. Somit werden die umfangreichen Arbeiten und Ansätze zur sicherheitsorientierten Programmierung [HL09] sowie dem Testen von Quelltext auf sicherheitskritische Fehler [Wy07] nicht weiter betrachtet.

Neben der Entwicklung sind der Betrieb und die Wartung in einer ebenfalls abgesicherten Laufzeitumgebung eine wesentliche Voraussetzung für die Absicherung von Software-Systemen. Die entwickelte Sicherheitsfunktionalität muss installiert und gewartet werden. Sollten bereits existierende Sicherheitsprodukte zum Einsatz kommen, so müssen diese ebenfalls gewartet werden und auch an die neuen Anforderungen der entwickelten Anwendung angepasst und entsprechend konfiguriert werden, um eine Absicherung zu gewährleisten. In dieser Arbeit wird jedoch ausschließlich auf den Entwicklungsaspekt eingegangen und der weitere Betrieb vernachlässigt.

Prämisse PS3: Einschränkung der betrachteten Sicherheitsprobleme und -maßnahmen

Die Breite der bereits entdeckten Angriffe und der entwickelten Sicherheitsmaßnahmen macht eine vollständige Betrachtung derer in einer einzelnen Arbeit unmöglich. Daher wird in dieser Arbeit eine Auswahl getroffen, welche die wesentlichen Sicherheitsmaßnahmen eines Software-Systems umfassen. Im Speziellen wird bei der konkreten Absicherung die Anwendung von Authentifizierungs- und Autorisierungs- sowie Zugriffskontrollmaßnahmen betrachtet. Ebenso wird, als Ausnahme zu Prämisse PS2, die in der Entwicklung und Betrieb von verteilten Software-Systemen relevante Nachrichtensicherheit mit einbezogen.

Prämisse P4: Vernachlässigung von weiteren Software-Qualitätsmerkmalen

Sicherheit stellt im Allgemeinen eine qualitative bzw. nicht-funktionale Eigenschaft von Software dar [So07] [IEEE-610.12-1990]. Da durch Sicherheitsmaßnahmen jedoch ausgedrückt wird, was einem Benutzer in der Interaktion mit der Software untersagt ist bzw. welche Interaktionen unerwünscht sind, haben Sicherheitsmaßnahmen einen wesentlichen Einfluss auf die Entwicklung und Ausführung eines Software-Systems und beeinflussen dessen funktionale sowie weitere nicht-funktionale Eigenschaften. Weitere Software-Qualitätseigenschaften, welche durch Sicherheit beeinflusst werden sind unter anderm Bedienbarkeit, Performanz und auch Entwicklungskosten eines Software-Systems [FH06]. Eine Betrachtung der Auswirkungen auf diese und weitere Eigenschaften von Software wird in dieser Arbeit nicht explizit betrachtet.

Prämisse PS5: Vernachlässigung von formalen Modellen

Mathematisch fundierte Modelle werden häufig im Forschungsbereich der Software-Sicherheit eingesetzt, um validierbare und verifizierbare sichere Software zu entwickeln. Allerdings sind solche Verfahren zurzeit noch sehr komplex und skalieren selten, wodurch sie in realen Software-Entwicklungsprojekten oft nicht eingesetzt werden. Auch in dieser Arbeit finden sie keine explizite Berücksichtigung, da das Ziel dieser Arbeit darin besteht, auch im Sicherheitsbereich unerfahrenen Software-Entwicklern ein Werkzeug zur Hand zu reichen, mit welchem eine effiziente Entwicklung von sicherer Software ermöglicht wird. Für viele der hier eingesetzten Modelle und Verfahren wurden jedoch die mathematischen Grundlagen in anderen Arbeiten bereits gezeigt, so dass eine Überführung des hier vorgestellten Ansatzes auf formale Modelle durchaus möglich ist.

Ebenso wird die formale Überprüfung, ob durch die Schutzmaßnahmen der gewünschte Schutz eines Software-Systems umgesetzt wird, nicht im Rahmen dieser Arbeit untersucht. Die betrachteten Schutzmaßnahmen werden als bewährte Methoden zum Schutz von bekannten Bedrohungen eingesetzt. Auf diese praktische Gewährleistung des Schutzes wird sich in dieser Arbeit verlassen.

1.6 Aufbau der Arbeit

In diesem Kapitel wurde der Themenkontext der Arbeit eingeführt und die erarbeiteten Beiträge motiviert. Wie in Abbildung 2 dargestellt, werden in diesem Kapitel sowie in den folgenden Kapiteln 2 und 3 die grundlegenden Kenntnisse zum Verständnis der Beiträge dieser Arbeit beschrieben. Anschließend werden in Kapiteln 4 bis 6 die Beiträge im Detail erläutert. Die Tragfähigkeit der Beiträge wird in Kapitel 7 demonstriert, während Kapitel 8 die abschließende Zusammenfassung der Arbeit enthält.

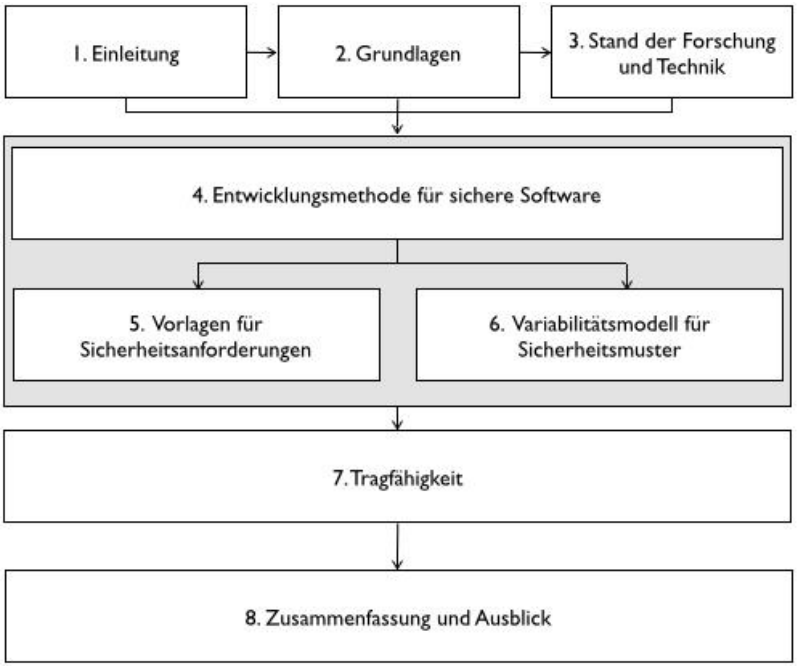

Abbildung 2: Aufbau der Arbeit

In Kapitel 2 werden die für das weitere Verständnis der Beiträge der Arbeit notwendigen Grundlagen vorgestellt. Hierzu wird insbesondere auf die Motivation und die Ziele der sicherheitsbasierten Software-Entwicklung eingegangen. Der Beitrag A1 zu einem sicherheitsbasierten Entwicklungsvorgehen basiert im Grundsatz auf dem Paradigma der Software-Produktlinien, weshalb die zugrundeliegenden Konzepte hierzu eingeführt werden. Sicherheitsmuster sind wesentlicher Bestandteil des Beitrags A3, weswegen die Grundlagen von Muster und Mustersprachen präsentiert werden.

Anschließend werden in Kapitel 3 relevante Forschungsansätze präsentiert, welche ebenfalls die Entwicklung von sicheren Software-Systemen untersuchen. Zur Abgrenzung der in dieser Arbeit vorgestellten Beiträge wird zunächst ein Anforderungskatalog für ein sicherheitsbasiertes Software-Entwicklungsvorgehen erstellt, anhand dessen die untersuchten Ansätze kritisch bewertet werden. Von dieser Untersuchung ausgehend, wird der Handlungsbedarf identifiziert, welcher die Motivation für die Beiträge der vorliegenden Arbeit darstellt.

Der in Kapitel 4 vorgestellte Beitrag B1 dieser Arbeit umfasst ein Vorgehensmodell für sicherheitsbasierte Software-Entwicklung. Das Vorgehensmodell fokussiert die Wiederverwendung von existierendem Sicherheitswissen zur effizienten Entwicklung von Sicherheitsmaßnahmen für Software-Systeme. Dabei wird insbesondere die Analyse von Sicherheitsanforderungen sowie der Entwurf von Sicherheitsfunktionalität fokussiert. Im Vorgehensmodell wird zwischen einer Entwicklung von wiederverwendbaren sicherheitsrelevanten Entwicklungsartefakten und deren Einsatz in einem konkreten

Entwicklungsprozess für ein Software-System unterschieden. Zu den wiederverwendbaren Entwicklungsartefakten gehören hierbei die in den folgenden Kapiteln vorgestellten Vorlagenkataloge für Sicherheitsanforderung sowie das Variabilitätsmodell für Sicherheitsmuster.

Vorlagen für Sicherheitsanforderungen stellen den Beitrag B2 dar und sind Thema des darauffolgenden Kapitels 5. Die wiederverwendbaren Vorlagen für Sicherheitsanforderungen beschreiben existierendes Sicherheitswissen bezüglich der Analyse von Schutzzielen eines Software-Systems und ermöglichen der entsprechenden Spezifikation sowie Modellierung von Sicherheitsanforderungen. Hierfür wird zunächst ein Domänenmodell für die in den Vorlagen verwendeten Konzepte und deren Relation untereinander definiert, wodurch die Struktur eines Vorlagenkatalogs spezifiziert wird. Anhand von ausgewählten Beispielen werden Vorlagen für bekannte Sicherheitsanforderungen vorgestellt. Abschließend wird im Kontext des übergeordneten Entwicklungsvorgehens die Abbildung von Sicherheitsanforderungen auf abstrakte Sicherheitsmaßnahmen dargestellt.

Darauf aufbauend wird in Kapitel 6 der Beitrag A3 vorgestellt, welcher ein Variabilitätsmodell für Sicherheitsmuster umfasst. Dieses kann als Werkzeug zur iterativen Verfeinerung von Sicherheitsmaßnahmen in Form von Sicherheitsmustern in der Entwurfsphase eingesetzt werden. Hierzu wird die Struktur des Variabilitätsmodells spezifiziert, wodurch die wesentlichen Konzepte zur Spezifikation von Beziehungen zwischen Sicherheitsmustern definiert werden. Im Gegensatz zu existierenden Mustersprachen, werden dabei die Verfeinerungsbeziehungen zwischen den Sicherheitsmaßnahmen hinsichtlich ihrer Abstraktion fokussiert. Hinzu kommt die Beachtung von optionalen und verpflichtenden Verwendungsbeziehungen von Sicherheitsmustern. Nach der Vorstellung der Grundstruktur werden existierenden Sicherheitsmuster für Authentifizierung, Autorisierung und Zugriffskontrolle in das Variabilitätsmodell eingeordnet.

Nach Vorstellung des sicherheitsbasierten Software-Entwicklungsvorgehens sowie der fokussierten Präsentation der wesentlichen Bestandteilen, wird in Kapitel 7 die Anwendung der Beiträge anhand konkreter Entwicklungsprozesse demonstriert. Hierzu werden für zwei verteilte Software-Systeme im Kontext des Karlsruher Institutes für Technologie (KIT) mittels des vorgestellten Entwicklungsvorgehens notwendige Sicherheitsmaßnahmen abgeleitet. Das KITCampusGuide-System ist dabei eine Web-basierte Anwendung, welche den Studierenden, Angestellten und Gästen des KIT eine kartenbasierte Benutzerschnittstelle zur Verfügung stellt, mit welcher relevante Orte und Personen des KIT-Campus gesucht und auf der Karte dargestellt werden können. Anhand des KITCampusGuides werden die Analyse von Sicherheitsanforderungen sowie die Entwicklung von Sicherheitsmaßnahmen für klassische Web-Anwendung demonstriert.

In Kooperation mit dem Fraunhofer Institut für Optronik, Systemtechnik und Bildauswertung (IOSB) wurde in der Forschungsgruppe Cooperation & Management (C&M) im Rahmen des EU-Projektes OpenIoT eine mobile Anwendung für die Suche und Reservierung von Arbeitsplätzen auf dem KIT-Campus entwickelt. Hierbei werden Konzepte des Internets-der-Dinge unter dem Einsatz von semantischen Technologien und Sensornetzwerken praktisch umgesetzt. In diesem zukunftsorientierten Kontext existieren zahlreiche Sicherheitsanforderungen, welche mit Hilfe der Beiträge dieser Arbeit analysiert und entsprechende Sicherheitsmaßnahmen entworfen werden.

Abschließend wird in Kapitel 8 eine Bewertung der erzielten Ergebnisse anhand der Anforderungen durchgeführt, welche zur Evaluierung von bestehenden Ansätzen in Kapitel 3 herangezogen wurden. Die Erweiterung des in dieser Arbeit vorgestellten Ansatzes zur sicherheitsbasierten Entwicklung wird in einem Ausblick auf zukünftige Arbeiten beleuchtet.

2 Grundlagen

Die in der vorliegenden Arbeit vorgestellten Beiträge zur Entwicklung von sicheren Software-Systemen setzen grundlegende Kenntnisse über die wesentlichen Konzepte der behandelten Themenbereiche voraus. In diesem Kapitel werden daher die wichtigsten Themen eingeführt, welche als Hintergrundwissen zum Verständnis der Arbeit vorausgesetzt werden. An die Beiträge der Arbeit angelehnt, lassen sich die Themengebiete dabei in zwei Teilbereiche unterscheiden.

Zunächst werden die wesentlichen Grundlagen zum Thema Sicherheit in Software-Systemen eingeführt. Hierzu wird mit Software-Sicherheit eine Einordnung der betrachteten Sicherheitsmaßnahmen sowie eine Abgrenzung zu weiteren Sicherheitsbereichen vorgenommen. Des Weiteren wird das Ziel eines sicherheitsbasierten Software-Entwicklungsvorgehens erläutert. Ebenso wird der abstrakte Ablauf eines solchen Vorgehens sowie verwendbare Werkzeuge vorgestellt. Hierbei wird der Fokus auf die Analyse und Entwurfsphase gelegt, welche für die vorliegende Arbeit relevant sind.

Im zweiten Teil werden Software-Entwicklungsparadigmen vorgestellt, welche einen wesentlichen Einfluss auf die Beiträge der Arbeit haben. Das im Beitrag B1 behandelte sicherheitsbasierte Entwicklungsvorgehen fokussiert die Wiederverwendung von Sicherheitsartefakten in den Entwicklungsphasen. Die zugrundeliegende Ideen basiert dabei auf dem Prinzip der Software-Produktlinien, welche die Wiederverwendung von Software-Artefakten in den Fokus stellen. Der Beitrag B3 basiert zudem auf dem Werkzeug der Funktionsmodelle, welche zur Dokumentation von Abhängigkeiten zwischen Software-Artefakten einer Software-Produktlinie eingesetzt werden. Ebenso stellen Sicherheitsmuster eine Grundlage für den Beitrag B3 dar, weshalb der Einsatz von Mustern in einen Software-Entwicklungsprozess erläutert wird.

2.1 Software-Sicherheit

Die vorliegende Arbeit befasst sich mit Fragestellungen zum Thema Software-Sicherheit, welche einen Teilbereich der gesamten Sicherheitsdomäne darstellt. Software-Sicherheit befasst sich mit der Entwicklung von sicheren Software-Systemen, indem Sicherheitsprinzipen durchgängig im Entwicklungsprozess betrachtet werden. Im folgenden Kapitel werden daher die grundlegenden Sicherheitskonzepte eingeführt, welche für den Bereich der Software-Sicherheit relevant sind.

2.1.1 Begriffsklärung

Eine umfassende Definition des Begriffes „Sicherheit” gestaltet sich als schwierig, da Sicherheit eine relative Eigenschaft ist und abhängig vom Kontext unterschiedliche Semantik annimmt [VG02:18]. So lassen sich im Kontext von IT-Systemen verschiedene Ebenen identifizieren, in welchen Sicherheitsmaßnahmen berücksichtigt werden müssen. Datensicherheit (engl. data security) betrachtet beispielsweise die sichere Speicherung von persistenten Daten. Dagegen wird unter Netzsicherheit (engl. network security) die Absicherung von Daten verstanden, welche zwischen IT-Systemen ausgetauscht wird. Systemsicherheit betrachtet die Absicherung von einzelnen IT-Systemen [WM05:8]. Hierdurch wird der Bedarf eines mehrschichtigen Ansatzes zur Sicherheit deutlich.

Im Rahmen dieser Arbeit wird der Bereich der Software-Sicherheit fokussiert, welche den Schutz von Funktionen und Daten eines Software-Systems betrachtet und hierzu die Entwicklung von sicheren Software-Komponenten in den Mittelpunkt stellt [MC04]. Ein Software-System setzt sich dabei aus Software-Komponenten zusammen, welche Daten verarbeiten, speichern und an andere Entitäten übermitteln. Dies wird durch die von Eckert getroffenen Definitionen präzisiert [Ec09:4], durch

welche zudem eine Unterscheidung zwischen dem Schutz von verarbeiteten Informationen sowie dem Schutz der Datenrepräsentation getroffen wird. Demnach wird Software-Sicherheit als Kombination von Funktions-, Informations- und Datensicherheit definiert.

Funktionssicherheit wird dabei als die Systemeigenschaft festgelegt, durch welche die implementierte Ist-Funktionalität mit der zuvor spezifizierten Soll-Funktionalität übereinstimmt. Hierdurch werden unzulässige Systemzustände verhindert. Informationssicherheit wird als die Systemeigenschaft definiert, welche die Vermeidung von unberechtigter Informationsgewinnung durch die Vorbeugung von unzulässigen Systemzuständen behandelt. Entsprechend wird die Systemeigenschaft, nur Systemzustände zuzulassen, durch welche kein unautorisierter Zugriff und Veränderung von Systemdaten erfolgt, als Datensicherheit definiert [Ec09:4f].

Die Unterteilung der Informationszustände Verarbeitung, Speicherung und Übermittlung wird auch im Sicherheitsmodell des National Training Standard for Information Systems Security Professionals (NSTISSC) des Committee on National Security Systems (CNSS) berücksichtigt [WM05:13] [NTSISSI-4011]. Das Modell beschreibt ein dreidimensionales Modell, in welchem die Informationszustände mit elementaren Schutzielen und Organisationsbereichen, wie zum Beispiel Sicherheitsrichtlinien und -technologien, in Verbindung gesetzt werden (vgl. Abbildung 3).

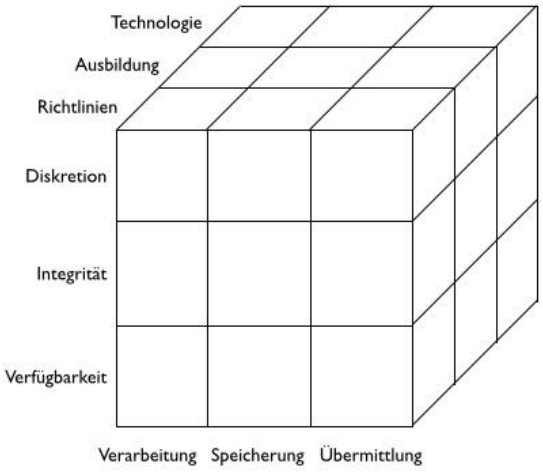

Abbildung 3: NSTISSC-Sicherheitsmodell [WM05:13]

Ebenso wird Sicherheit als dynamischer Prozess statt einer statischen Eigenschaft beschrieben [Ec09:5]. Die Sicherstellung der Sicherheit eines Systems bedarf somit der kontinuierlichen Inspektion und Revision. Dieser dynamische Aspekt ist ebenfalls die Grundlage für eine sicherheitsbasierte Software-Entwicklung.

2.1.2 Schutzziele

Im Gegensatz zu den abstrakten Definitionen des Sicherheitsbegriffs stellen Schutzziele (engl. security goals) konkretere Eigenschaften und Anforderungen von schützenswerten Ressourcen eines Software-Systems dar. Diese Ressourcen umfassen Daten sowie die sie verarbeitenden Funktionen des Software-Systems. Schützenswerte Ressourcen stellen dabei einen Wert für die beteiligten Personen dar. Eine Änderung der sicherheitsrelevanten Eigenschaften durch ungewünschte Ereignisse führt zu einer Reduzierung des Wertes [WM05:9] [Ec09:6]. Das Ziel von Sicherheitsmaßnahmen ist es somit, eine negative Veränderung der Eigenschaften zu verhindern. Im Zuge eines mehrschichtigen Ansatzes zur Umsetzung von Sicherheitsmaßnahmen lassen sich diese und weitere Schutzziele auf

unterschiedliche Art und Weise in ein Software-System und dessen Ausführungsumgebung integrieren. [FP01].

Zu den bekanntesten Schutzzielen werden die Diskretion (engl. confidentiality), Integrität (engl. integrity), sowie Authentizität (engl. authenticity) und Verfügkarbeit (engl. availability) gezählt. Unter Diskretion wird dabei die Eigenschaft oder Anforderung verstanden, dass eine Einsicht in Daten für unberechtigte Entitäten unmöglich ist [Ec09:8] [WM05:10] [Ra02:53]. Integrität wiederum bezeichnet die Eigenschaft, dass Daten nicht unbemerkt von unberechtigten Entitäten manipuliert werden können [Ec09:7] [WM05:12] [Ra02:52]. Im Gegensatz zur unberechtigten Veränderung und Einsicht fokussiert die Eigenschaft der Authentizität die Echtheit von Daten, wodurch der Nachbildung von Daten vorgebeugt werden soll [WM05:10] [Ec09:6]. Die Anforderung der Verfügbarkeit behandelt den ungehinderten Zugang zu Daten für berechtigte Entitäten [WM05:10] [Ec09:10] [Ra02:52].

Schutzziele stehen dabei in direkten Bezug zu einander und beeinflussen sich gegenseitig [WM05:11]. Beispielsweise steht durch die immer stärkere Personalisierung von Software-Systemen der Schutz von personengebundenen Daten immer mehr im Vordergrund. Schutzziele wie Schutz der Privatsphäre (engl. Privacy, [VG02:20] [Ec09:12]) sowie Anonymität bzw. Pseudonymität geben dies zum Ausdruck [VG02:21] [Ec09:12]. Anonymitäterfordert die Modifikation von Daten dahingehend, dass die ursprünglichen personenbezogenen Angaben einem Subjekt zugeordnet werden können [Ec09:12]. Dies steht im Widerspruch zum Schutzziel der Nachweisbarkeit (engl. non-repudiation [WM05:370]), bei welchem die Unleugbarkeit von versendeten oder geänderten Daten durch die Prüfung der Absenderidentität im Vordergrund steht. Somit sind bei der Entwicklung von sicheren Software-Systemen die Schutzziele entsprechend abzuwägen [VG02:22].

2.1.3 Sicherheitsprinzipien

Schutzziele werden in Software-Systemen durch Sicherheitsmaßnahmen umgesetzt. Bei der Implementierung von Sicherheitsmaßnahmen haben sich Sicherheitsprinzipien bewährt, welche während der Konstruktion von Software-Systemen berücksichtigt werden sollten [HS75] in [Ec09:168]. Zu diesen Sicherheitsprinzipien zählt unter anderem das Erlaubnisprinzip (engl. fail-safe defaults), durch welches jeder Zugriff zunächst untersagt werden sollte und nur bei explizit gewährter Erlaubnis zugelassen wird. Das Vollständigkeitsprinzip (engl. complete mediation) besagt, dass die Zulässigkeit von jeglichen Zugriffen überprüft werden sollte. Das bekannte Prinzip der minimalen Rechte (engl. need-to-know bzw. least privilege, [VG02:100ff]) schreibt weiterhin vor, das jedem Subjekt nur die zur Erfüllung seiner Aufgaben nötigen Zugriffsrechte zugewiesen wird.

Für den Software-Entwurf elementar ist das Prinzip des offenen Entwurfs (engl. open design bzw. no security through obscurity), welches die Offenlegung der Verfahren und Mechanismen zur Umsetzung von Sicherheitsmaßnahmen fordert. Kernaussage dieses Prinzips ist, dass die Sicherheitsfunktionalität von Software-Systemen nicht von dessen Geheimhaltung abhängig ist, da eine ungewünschte Veröffentlichung der Funktionalität die Schutzmaßnahmen untergraben würde [VG02:109].

Neben allgemeinen Sicherheitsprinzipien lassen sich auch spezifische Prinzipen identifizieren, welche häufig in einer bestimmten Fachdomäne angewendet werden. So besagt zum Beispiel das Mehr-Augen-Prinzip, das kritische Tätigkeiten nicht durch einzelne Subjekte durchgeführt werden sollen, sondern die Aktivitäten von unterschiedlichen Subjekten akzeptiert werden müssen. Die Spezialisierung dieses Prinzips für zwei Subjekte ist das Vier-Augen-Prinzip, welches häufig im Bankenbereich angewendet wird [WM05:480]. Das Prinzip der Trennung von Zuständigkeiten (engl. seperation of duties [Pa72]) lässt sich ebenfalls als Sicherheitsprinzip (engl. compartmentalization [VG02:102]) anwenden. Hierbei wird Subjekten lediglich eine Teilaufgabe zugewiesen, so dass die gesamte Aufgabe keiner zentralen Kontrolle unterliegt [WM05].

Die Berücksichtigung von Sicherheitsprinzipien ist ausschlaggebend für die Konstruktion von zuverlässigen Sicherheitsmaßnahmen in Software-Systemen. Insbesondere wird durch die Integration von Sicherheitsprinzipien in die Entwicklung von Software-Systemen eine Absicherung gegen aktuelle sowie zukünftige Bedrohungen ermöglicht [VG02:92]. Die Anwendung der Sicherheitsprinzipien ist ebenso wie die Bestimmung von Schutzzielen kontextabhängig, da die Prinzipien sich gegenseitig beeinflussen. Ebenso bietet deren Berücksichtigung keinen vollständigen Schutz, sondern ermöglicht die Aufdeckung von möglichen Sicherheitsrisiken bei der Entwicklung von Sicherheitsmaßnahmen [VG02:92].

2.1.4 Authentifizierung, Autorisierung und Zugriffskontrolle

Zu den bekanntestes Sicherheitsmaßnahmen, welche Schutzziele von Software-Systemen umsetzen können, zählen Verfahren zur Authentifizierung, Autorisierung und Zugriffskontrolle. Authentifizierung ermöglicht die Überprüfung der Authentizität der Identität von Subjekten und Objekten in einem Software-System und ist gleichzeitig Voraussetzung für die Umsetzung von weiteren Sicherheitsmaßnahmen [Ec09:439]. Hierzu muss die Identität anhand von charakteristischen Eigenschaften zweifelsfrei nachgewiesen werden. Demnach ist die Vorlage von Identitätsnachweisen (engl. credentials) notwendig, sodass die Abbildung von einem Subjekt bzw. Objekt auf die zu beanspruchende Identität im Software-System überprüft werden kann [Ec09:339].

Authentifizierung ist die wesentliche Voraussetzung für die Sicherheitsmaßnahme der Zugriffskontrolle. Die wesentliche Aufgabe der Zugriffskontrolle besteht dabei in der Bereitstellung von Mechanismen, welche die Überprüfung von Zugriffsrechten auf Objekte im Software-System spezifizieren [Ec09:621]. Zugriffskontrolle ermöglicht die Umsetzung von verschiedenen Schutzzielen, unter anderem der Integrität und Diskretion von Objekten, welche durch Zugriffskontrollmechanismen geschützt sind. Ebenso steht Zugriffskontrolle im Zentrum der Umsetzung von verschiedenen Sicherheitsprinzipen, wie zum Beispiel minimale Rechte und Trennung von Zuständigkeiten.

Eine weitere Voraussetzung von Zugriffskontrolle besteht in der Verwaltung von Zugriffsrechten. Diese Autorisation von Subjekten wird, vor allem in praktischem Einsatz, häufig unter dem Begriff Zugriffskontrolle zusammengefasst. In dieser Arbeit wird diese konzeptionelle Trennung explizit berücksichtigt. Grund hierfür sind die unterschiedlichen Umsetzungen dieser Sicherheitsmaßnahmen. Unter Autorisierung wird die Vergabe von Rechten nach bestimmten Modellen, wie zum Beispiel rollenbasierte Zugriffskontrolle, verstanden. Daher steht in dieser Sicherheitsmaßnahme die Formulierung und Vergabe von Zugriffsrechten im Vordergrund. Dagegen stehen bei der Zugriffskontrolle die architekturellen Mechanismen im Vordergrund, mittels welchen die durch die Autorisierung vorgegeben Zugriffsrechte bei Eintritt eines Zugriffes ausgewertet und umgesetzt werden.

2.2 Sicherheitsbasierte Analyse und Entwurf

Gemäß Anderson fokussiert die sicherheitsbasierte Software-Entwicklung die Konstruktion von Software-Systemen, welche trotz des Eintretens von ungewollten Ereignissen zuverlässig funktionieren. Hierzu werden Werkzeuge, Prozesse und Methoden bereitgestellt, welche bei dem Entwurf, der Implementierung und dem Testen von Software-Systemen hinsichtlich Sicherheit unterstützen [An08:3]. Die Disziplin der sicherheitsbasierten Software-Entwicklung ist ein relativ junges Forschungsgebiet, welches zunehmende Berücksichtigung erfährt [Ec09:168].

Im Folgenden werden die wesentlichen Werkzeuge, Prozesse und Methoden vorgestellt, welche derzeit für die Entwicklung von sicheren Software-Systemen zur Verfügung stehen. Hierbei wird

insbesondere auf die für diese Arbeit relevanten Phasen der Anforderungsanalyse und der Entwurfsphase eingegangen. Weitere Phasen stehen nicht im Fokus der Arbeit und werden nur der Vollständigkeit halber kurz beschrieben. Ansätze, welche sich um die Einordnung der hier vorgestellten Ansätze in einen durchgängigen Entwicklungsprozess bemühen, sind die Motivation für diese Arbeit und werden daher im Kapitel 3 kritisch diskutiert.

Sicherheitsbasierte Software-Entwicklung ist eine Ergänzung zu einem fachlichen Software-Entwicklungsprozessmodell wie etwa dem Scrum-Rahmenwerk [Gl13], dem Rational Unified Process (RUP [Ba06]) oder dem Spiralmodell von Boehm [So07]. Daher wird zur Vorstellung der Ansätze der in Abbildung 4 dargestellte generische Entwicklungsprozess verwendet [MC04]. Hierbei wird der Entwicklungsprozess in drei grobe Phasen eingeteilt. Die Anforderungsanalysephase ermittelt die Sicherheitsanforderungen für das Software-System und modelliert abstrakte Domänenmodelle für die Fachdomäne. In der Entwurfsphase werden die Anforderungen in eine Sicherheitsarchitektur umgesetzt, deren Komponenten anschließend in einem Feinentwurf modelliert werden. Die Implementierungs- und Testphase fokussieren die Umsetzung der Entwurfsmodelle in einen ausführbaren Programmcode sowie die Validierung und Verifikation der Entwurfsmodelle bzw. Anforderungen.

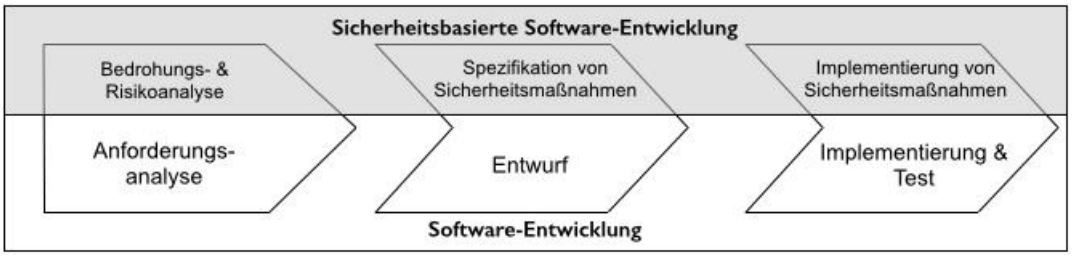

Abbildung 4: Generischer Entwicklungsprozess

2.2.1 Analyse von Sicherheitsanforderungen

Die Bestimmung von Sicherheitsanforderungen ist eine der grundlegendsten Aktivitäten in einem sicherheitsbasierten Entwicklungsprozess. Mittels einer Anforderungsspezifikation können individuell auf ein zu entwickelndes Software-System abgestimmte Sicherheitsmaßnahmen umgesetzt werden. Auf diese Weise werden Sicherheitslücken durch das unstrukturierte nachträgliche Hinzufügen von Sicherheitsmaßnahmen vermieden [TJ+08] [Fi03a] [HM+06].

Generisches Vorgehen

Ausgangspunkt der Analyse ist dabei zunächst die Identifikation und Bewertung von schützenswerten Ressourcen und die Bestimmung eines zugehörigen Schutzbedarfes [Ra02:26ff] [VG02:34] [Ec09:169ff]. Anschließend werden wesentliche Schutzziele für die Ressourcen definiert und Sicherheitsanforderungen spezifiziert, durch welche diese Schutzziele gewährleisten werden sollen. Dieser Sachverhalt ist in Abbildung 5 dargestellt.

Ein zu entwickelndes Software-Systems verwaltet Ressourcen, welche einen geschäftlichen Wert darstellen. Hierbei werden in dieser Arbeit unter Ressourcen sowohl die zugrundeliegenden Informationsressourcen als auch deren Verarbeitung und die Funktionen zum Zugriff auf die Ressourcen durch das Software-System verstanden. Der Wert der Ressource ist zum einen von dem fachlichen Kontext des Software-System abhängig und zum anderen von den am Software-System beteiligten Personen (engl. stakeholder). Eine Reduktion des Wertes einer Ressource ist dabei nicht erwünscht bzw. sollte minimiert werden, da hierdurch ein Wettbewerbsvorteil verloren geht. Der

Abbildung 5: Generisches Vorgehen zur Sicherheitsanforderungsanalyse

sicherheitsrelevante Wert der Ressource ist den beteiligten Personen nicht immer bewusst. Daher werden Ressourcen durch den fachlichen Kontext des Software-Systems festgelegt, während die Bestimmung des Ressourcenwertes Teil einer Sicherheitsanalyse ist.

Ausgehend von dem Wert der Ressource wird der Schutzbedarf der Ressource definiert. Die Ausprägung des Schutzbedarfes wird dabei durch eine Bedrohungs- bzw. Risikoanalyse bestimmt, in welcher die Gefährdungslage der Ressourcen sowie die möglichen Schäden und Auswirkungen analysiert werden [Ec09:176]. In einer Bedrohungsanalyse (engl. threat analysis bzw. risk analysis, [VM04]) werden die möglichen Bedrohungen für ein Software-System identifiziert und diese anhand ihrer Auftrittswahrscheinlichkeit priorisiert. Bedrohungen sind hierbei als durch Entitäten bewusst oder unbewusst ausgelöste, potentielle Ereignisse anzusehen, welche den Schutzzielen entgegenwirken [Ec09:15] [WM05:31]. Durch die Priorisierung erschließt sich das Ausmaß des Schutzbedarfes für eine Ressource.

Die Bestimmung des Schutzbedarfes ist die Grundlage für die Spezifikation von verschiedenen Schutzzielen für eine schützenswerte Ressource. Schutzziele stellen dabei sicherheitsrelevante Eigenschaften der Ressource dar, welche eingehalten werden müssen, um den Wert der Ressource zu erhalten. Diese Schutzziele spezifizieren somit Bedingungen, welche entgegengesetzt zu den analysierten Bedrohungen stehen. Zu den bekanntesten Schutzzielen gehören hierbei Vertraulichkeit, Integrität und Authentizität [Ec09:6f] [WM05:12] [An08:13], mittels welchen entsprechend eine ungewünschte Einsicht und Veränderung der Ressource verhindert werden soll sowie die Echtheit der Ressource garantiert werden kann.

Die Durchsetzung dieser abstrakten Schutzziele mithilfe von konkreten Sicherheitstechnologien ermöglicht eine Absicherung von Ressourcen. Hierzu werden zunächst Sicherheitsanforderungen formuliert [Fi03a], welche den Bedrohungen entgegenwirken, deren Eintrittswahrscheinlichkeit minimieren bzw. deren Auswirkungen reduzieren und somit die Schutzziele erhalten sollen. Analog zu fachlichen Anforderungen abstrahieren Sicherheitsanforderungen von konkreten Sicherheitstechnologien. Die Sicherheitsanforderungen drücken somit zunächst aus, dass eine Sicherheitsmaßnahme für eine Ressource ergriffen werden soll, aber legen nicht fest, wie diese Maßnahme umgesetzt werden soll [Fi03a] [HM+06]. Andererseits spezifizieren sie häufig Bedingungen, welche die fachlichen Anforderungen einschränken. Hierdurch ist eine weitere Abhängigkeit zur fachlichen Entwicklung gegeben.

Werkzeuge

Die Spezifikation von Sicherheitsanforderungen steht somit in direktem Zusammenhang mit der fachlichen Domäne des zu entwickelnden Software-Systems. Um diese Abhängigkeit auch in Analysemodellen zu verdeutlichen, lassen sich Werkzeuge wie Missbrauchsanwendungsfälle (engl. misuse cases [SO05] bzw. abuse cases [MF99]) bzw. Missbrauchsaktivitäten (engl. misuse activities [BF+08]) sowie Sicherheitsanwendungsfälle (engl. security use cases [Fi03b]) einsetzen.

Missbrauchsanwendungsfälle ergänzen UML-Anwendungsfalldiagramme um Modellierungselemente, mittels welchen Angreifer (engl. mis-actor [SO05]) sowie Angriffe und Bedrohungen für ein oder

mehrere Anwendungsfälle spezifiziert werden können. Somit ermöglichen sie es, die Sicherheitsproblematik einer fachlichen Domäne besser zu verstehen [SO05]. Entsprechend wirken Sicherheitsanforderungsfälle den Angriffen entgegen und beschreiben somit Sicherheitsanforderungen, welche für ein oder mehrere Anwendungfälle berücksichtigt werden müssen. Abbildung 6 zeigt beispielhaft diese Erweiterungen der Modellierungslemente von UML-Anwendungsfalldiagramme.

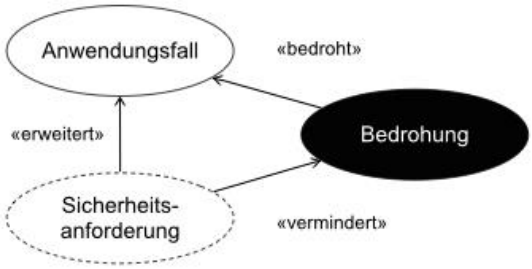

Abbildung 6: Missbrauchsanwendungsfälle und Sicherheitsanwendungsfälle

Als Ergänzung der Missbrauchsanwendungsfälle lassen sich die Interaktionen eines Anwendungsfalles mittels Missbrauchsaktivitäten (engl. misuse actions [FV+06] bzw. misuse activities [BF+08]) hinsichtlich möglicher Bedrohungen untersuchen. Hierbei wird jede Aktivität des Anwendungsfalles auf Verletzung von Schutzzielen durch Angreifer analysiert.

Die Erstellung von Analysemodellen zur Beschreibung von wesentlichen Begriffen der fachlichen Domäne ist eine weitere wesentliche Aktivität in der Software-Entwicklung [BD10]. Zur Bestimmung von sicherheitsrelevanten Begriffen für eine fachliche Domäne können Sicherheitsanalysemuster [FY07] eingesetzt werden. Diese beschreiben Sicherheitskonzepte, welche sich im Einsatz zum Schutz einer fachliche Domäne bewährt haben. Konkrete Werkzeuge hierzu liefern zum Beispeil UMLSec [Jü05] und SecureUML [LB+02], welche basierend auf UML, Objektmodelle zur Beschreibung von Sicherheitsmechanismen, wie etwa rollenbasierte Zugriffskontrolle, zur Verfügung stellen. Ebenso ist auch die feingranulare Modellierung von Software-Systemen möglich.

2.2.2 Entwurf von Sicherheitsarchitekturen und -maßnahmen

Ausgehend von den Sicherheitsanforderungen wird der Umfang der Sicherheitsfunktionalität bestimmt. Hierzu wird eine Sicherheitsarchitektur entworfen, welche die notwendige Sicherheitsfunktionalität in Form von Diensten zentralisiert bereitstellt. Bei der Umsetzung der Sicherheitsarchitektur ist dabei die fachliche Software-Architektur sowie deren Subsysteme und Komponenten zu berücksichtigen. Die aus den Sicherheitsanforderungen stammenden Sicherheitseinschränkungen der fachlichen Komponenten bestimmten den Umfang der Sicherheitsarchitektur [Ra02:49]. Die Aktivitäten für den Enwurf sind in Abbildung 7 dargestellt.

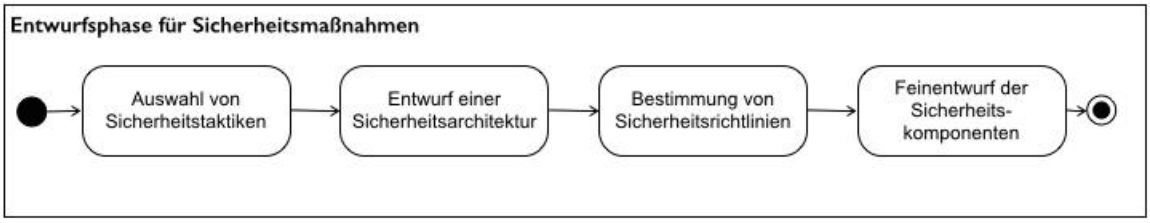

Abbildung 7: Generisches Vorgehen für den Entwurf von Sicherheitsmaßnahmen

Taktiken

Für die Umsetzung von Sicherheitsanforderungen werden zunächst angemessene Sicherheitsmaßnahmen ausgewählt. In einer Kosten- und Nutzenanalyse werden unter Berücksichtigung des Schutzbedarfes und des Wertes einer Ressource ein oder mehrere Maßnahmen ausgewählt. Die Sicherheitsmaßnahmen lassen sich dabei entsprechend ihrer Kosten und der Umsetzung der Sicherheitsanforderung grob in die Kategorien bzw. Taktiken vorbeugende, detektierende und wiederherstellende Maßnahmen einteilen [SF+05:150] [FH06:283] [WM05:138ff].

Präventive Maßnahmen sind darauf ausgelegt, Bedrohungen zu unterbinden bzw. deren Eintrittswahrscheinlichkeit zu reduzieren [SF+05:149]. Im Gegensatz dazu sind detektierende Maßnahmen darauf ausgelegt, eintretende Bedrohungen lediglich zu erkennen [SF+05:149] [FH06:284]. Wiederherstellende Maßnahmen haben wiederum zum Ziel, die Schadensauswirkungen von Bedrohungen zu reduzieren [SF+05:149] [FH06:284]. Zur Umsetzung von Sicherheitsanforderungen wird zumeist eine Kombination von mehreren Sicherheitsmaßnahmen aus unterschiedlichen Kategorien benötigt, so dass auch eine mehrschichtige Absicherung erzielt wird.

Referenzarchitekturen

Sind die Taktiken für eine Sicherheitsanforderung ausgewählt, können anschließend entsprechende Maßnahmen zur Umsetzung der Sicherheitsanforderungen in einer Sicherheitsarchitektur modelliert werden. Analog zur fachlichen Software-Architektur lassen sich verschiedene Abstraktionsebenen unterscheiden, welche iterativ entwickelt werden [Ra02:16]. Zunächst beschreibt eine abstrakte Sicherheitsarchitektur den wesentlichen Architekturstil sowie die wesentlichen Sicherheitskomponenten. Das Ziel hierbei ist es, das Entwurfsprinzip der Datenkapselung (engl. information hiding) umzusetzen, indem die Schnittstellen zwischen den Sicherheitskomponenten spezifiziert werden [Ra02:16].

Für die Beschreibung von Sicherheitsarchitekturen auf höchster Abstraktionsebene lassen sich Referenzarchitekturen als Vorlage verwenden, welche die wesentlichen Architekturstile und Sicherheitskomponenten unabhängig von einer fachlichen Domäne spezifizieren [FH06] [EB+07] [BM07]. Der Entwurf einer konkreten Sicherheitsarchitektur für ein Software-System orientiert sich dabei an einer generischen Referenzarchitektur und kann aufgrund von konkreten Einschränkungen der fachlichen Domäne davon abweichen.

Sicherheitsmuster

Die abstrakte Sicherheitsarchitektur ist Ausgangspunkt für den Feinentwurf, in welcher die Sicherheitsfunktionalität im Detail entworfen wird. Ziel ist es dabei, präzise jedoch weitestgehend technologieunabhängige Modelle der Sicherheitsfunktionalität zu spezifizieren, welche in der folgenden Implementierungsphase auf konkrete Sicherheitstechnologien abgebildet werden können. Die interne Struktur der Sicherheitskomponenten lässt sich mittels Sicherheitsmuster beschreiben. Diese beschreiben bewährte Strukturen und stellen anerkannte Lösungen zu bekannten Sicherheitsproblemen bereit.

Analog zum Ansatz von Entwurfsmustern [GH+94] beschreiben Sicherheitsmuster bewährte Methoden und Praktiken zur Lösung von bereits bekannten Sicherheitsproblemen auf einem technologieunabhängigen Abstraktionsniveau [SF+05]. Es existieren dabei zahlreiche Kataloge, in denen auf Sicherheitsanforderungen zutreffende Sicherheitsmuster dokumentiert sind [SF+05] [CN+05]. Aufgrund der allgemeinen Verfügbarkeit und Dokumentation ermöglicht der Einsatz von Mustern zudem eine einheitliche Kommunikationsgrundlage, anhand derer sich die am Entwicklungsprozess beteiligten Personen austauschen können [SF+05]. Da Sicherheitsmuster eine elementare Grundlage der vorliegenden Arbeit sind, wird auf diesen Bereich im Kapitel 2.4 genauer eingegangen.

Werkzeuge

Zur Modellierung von Sicherheitslösungen auf Objektebene können verschiedene Werkzeuge verwendet werden [NB+10]. UMLSec [Jü05] und SecureUML [LB+02] sind Beispiele für Erweiterungen der UML, welche zur Modellierung von sicherheitsrelevanten Informationen, insbesondere von Zugriffskontrollmaßnahmen, in UML-Modellen eingesetzt werden können. Wie in [MC04] dargestellt, sollte eine Risikoanalyse auch während der Architektur- und Feinentwurfsphase durchgeführt werden, um den Einsatz von Sicherheitslösungen zur Minderung von Angriffen zu verifizieren. Angriffs- und Bedrohungsbäume [Sc01] [IK+08] können genutzt werden, um Angriffe auf einen tieferen Detaillierungsgrad als in der Analysephase zu spezifizieren.

Sicherheitsstandards und -produkte

Auf einer tieferen Abstraktionsstufe wird die Abbildung auf eine Implementierungsplattform spezifiziert. Hierzu wird eine Middleware bestimmt, welche die Infrastrukturkomponenten der Sicherheitsfunktionalität bereitstellt. Ebenso werden die sicherheitsrelevanten Daten bestimmt, welche von dem Software-System verarbeitet und gespeichert werden.

Zur Spezifikation der Implementierungsplattform stehen zahlreiche Sicherheitsstandards zur Verfügung, welche für spezifische Sicherheitsfunktionalitäten definiert wurden. Der Einsatz von Standards ermöglicht dabei die Integration von fachlichen Komponenten und Sicherheitskomponenten. Ebenso wird die einheitliche Handhabung der gleichen Sicherheitsfunktionalität in verschiedenen Software-Systemen ermöglicht. Sicherheitsstandards werden zudem durch Sicherheitsprodukte umgesetzt, welche bevorzugt zur Umsetzung der Implementierungsplattform eingesetzt werden sollten. Grund hierfür ist vor allem die Reife der Produkte, wodurch zusätzlichen Sicherheitslücken durch Eigenentwicklungen von Sicherheitsfunktionalität vorgebeugt wird [Ra02].

2.3 Software-Produktlinienentwicklung

Zahlreiche Paradigmen zur Entwicklung und Strukturierung von Software-Systemen befassen sich mit der Wiederverwendung von Entwicklungsartefakten zur Beherrschung der zunehmenden Komplexität der Software-Systeme. Insbesondere das Paradigma der Software-Produktlinien stellt eine Methode bereit, wiederverwendbare Entwicklungsartefakte aus einer Vielzahl von ähnlichen Software-Systemen aufzubereiten und in die Entwicklung neuer Systeme einfließen zu lassen.

Das Paradigma ist wesentliche Grundlage für die in Kapitel 4 vorgestellte sicherheitsbasierte Software-Entwicklungsmethode, welche ebenfalls die Aufbereitung und Integration von Sicherheitswissen in einen fachlichen Software-Entwicklungsprozess zum Ziel hat. Ebenso sind Funktionsmodell (engl. feature models) ein in der Software-Produktlinienentwicklung angewendetes Werkzeug, welches Ausgangspunkt für das in Kapitel 6 vorgestellte Variabilitätsmodell ist.

2.3.1 Software-Produktlinien

Die Entwicklung von Software-Produktlinien ist angelehnt an industrielle Produktlinienansätze, welche in verschiedenen Ingenieursbranchen etabliert sind. Beispiele für industrielle Produktlinien finden sich in der Automobilbranche, in welcher ein Kunde für ein Auto eines bestimmten Modells ebenfalls aus verschiedenen vom Hersteller vorgegebenen Optionen sein individuelles Modell zusammenstellen kann. Ziel des Produktlinien-Paradigmas ist es, eine Individualisierung von Produkten trotz einer Massenproduktion zu ermöglichen. Mit dem Ansatz der Software-Produktlinienentwicklung wird insbesondere die Reduktion von Entwicklungskosten, eine verbesserte Qualität der erzeugten Produkte sowie eine reduzierte Entwicklungszeit verfolgt [PB+05:9f].

Der Begriff der Software-Produktlinie (engl. software product line, SPL) wurde vom Software Engineering Institute (SEI) der Carnegie Mellon University geprägt:

"A software product line is a set of software-intensive systems sharing a common, managed set of features that satisfy the specific needs of a particular market segment or mission and that are developed from a common set of core assets in a prescribed way." [CN02:5]

Ausgehend von der Definition lässt sich die Ausrichtung des von einer Software-Produktlinie erstellten Software-Systems auf eine spezifische Domäne ableiten. Eine Domäne bezeichnet hierbei einen Spezialisierungs- bzw. Wissensbereich oder eine Menge von verwandten Funktionalitäten und kann selbst in Unterdomänen aufgeteilt werden [CN02:14]. Ein für diese Arbeit relevanter Teilbereich stellt dabei die Software-Sicherheit dar, welcher in Software-Systemen unterschiedlicher fachlicher Domänen eine Rolle spielt.

Die wesentlichen Bestandteile (engl. core assets) einer Software-Produktline beschreiben die Software-Plattform, welche zur Erstellung der Software-Systeme eingesetzt wird. Die Software-Plattform umfasst dabei unter anderem eine domänenspezifische Referenzarchitektur für die Software-Systeme, wiederverwendbare Komponenten sowie Domänenmodelle, Anforderungen und Dokumentationen. Einzelne Software-Systeme werden aus der Plattform anhand vorgegebener Vorschriften konstruiert. Dadurch ergeben organisatorische Aspekte, da die Pflege und Evolution der Plattform gegenüber der Entwicklung von Produkten in den Vordergrund gestellt wird.

Die Plattform einer SPL wird durch zwei wesentliche Eigenschaften geprägt. Zum einen stellen die gemeinsamen Artefakte der Plattform das Grundgerüst von allen Software-Systemen dar, welche durch die SPL gefertigt werden. Da die Produkte einer Produktlinie für eine spezielle Domäne bestimmt sind, sind große Gemeinsamkeiten in der Funktionalität zwischen den Software-Systemen abzusehen, wodurch eine Massenproduktion ermöglicht wird. Diese Gemeinsamkeiten werden somit in jedem Software-System eingesetzt und stellen hochgradig wiederverwendbare Entwicklungsartefakte dar.

In Anlehnung an die Software-Sicherheit stellen beispielsweise Sicherheitsmaßnahmen, welche in mehreren Software-Systemen eingesetzt werden, gemeinsame Funktionen der SPL dar. Zum anderen wird explizit die Variabilität von Bestandteilen der Plattform betrachtet, wodurch die Individualisierung von Software-Systemen ermöglicht wird. Hierbei werden für Teile der Software-Plattform alternative Varianten bereitgestellt, wodurch Funktionen zwischen Software-Systemen abweichen können. So kann zum Beispiel die Funktion „Zugriffskontrolle" in einem Software-System der SPL durch die Variante rollenbasierte Zugriffskontrolle umgesetzt werden, während in einem anderen Software-System eine attributbasierte Zugriffskontrolle gewählt wird.

Die Variabilität wird durch sogenannte Variationspunkte für bestimmte Entwicklungsartefakte der Software-Plattform spezifiziert. Variationspunkte bezeichnen dabei konkrete Stellen der Entwicklungsartefakte, an denen verschiedene Ausprägungen, sogenannte Varianten, vorgesehen sind [PB+05:62] [CN02:155]. In der Software-Entwicklung können Variationspunkte und Varianten durch Variationsmechanismen wie etwa Vererbung in objektorientierten Modellen, Parametrisierung von Komponenten, generativen Methoden, etc. [CN02:155] ausgedrückt werden. Zu beachten ist auch, dass in der Entwicklung einer Produktlinie nicht alle möglichen Variationspunkte und nicht alle möglichen Varianten von vornherein unterstützt werden müssen. Die Variationspunkte ergeben sich aus der Evolution und Pflege der Plattform, in dem neue Kundenanforderungen berücksichtigt werden und in die Plattform mit einfließen.

Neben dem Bereich der Software-Entwicklung spielen im Paradigma der Software-Produktlinien auch technische und organisatorische Verwaltungsaufgaben eine entscheidende Rolle [CN02:29]. Aufgrund

der Ausrichtung dieser Arbeit werden im Folgenden lediglich die Aspekte des SPL-Paradigmas betrachtet, welche eine Rolle für die Software-Entwicklung spielen.

2.3.2 Generisches Entwicklungsvorgehen

Die Entwicklung von Software-Systemen im Paradigma der Software-Produktlinien ist durch einen zweigeteilten Entwicklungsprozess gekennzeichnet. Da der Software-Plattform eine entscheidende Rolle in der Entwicklung von Software-Systemen zukommt, wird deren Pflege und Evolution in einer separaten Domänenentwicklung (engl. domain engineering) vollzogen. Dagegen wird in der Anwendungsentwicklung (engl. application engineering) die Plattform zur Entwicklung von konkreten Software-Systemen eingesetzt. Häufig findet man die Kurzbeschreibungen „Entwicklung für Wiederverwendung“ und „Entwicklung mit Wiederverwendung“ für Domänen- bzw. Anwendungsentwicklung. Der in Abbildung 8 abgebildete generische Enwicklungsprozess orientiert sich an dem in [PB+05:22] vorgestellten SPL-Entwicklungsvorgehen.

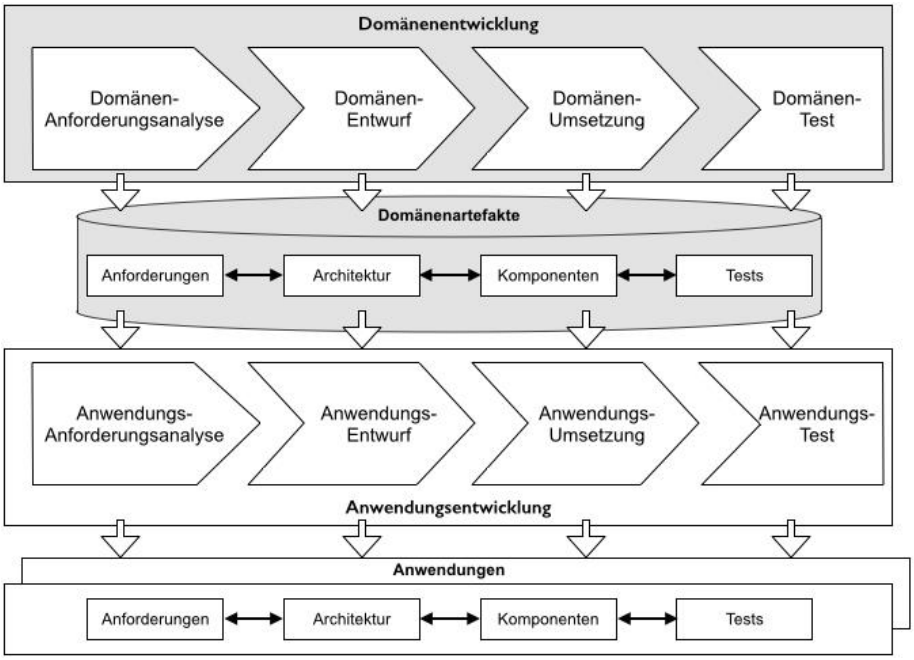

Abbildung 8: Entwicklungsvorgehen für Software-Produktlinien

Domänenentwicklung

Das Ziel der Domänenentwicklung ist die Festlegung und Abgrenzung des Anwendungsbereiches der Software-Systeme, welche mittels eines Produktlinienansatzes entwickelt werden sollen. Zusätzlich werden die Bestandteile der Software-Plattform spezifiziert, um Gemeinsamkeiten und Variabilität der Produktlinie zu definieren. Außerdem wird ein Produktionsplan erstellt, anhand dessen die Fertigung der Software-Systeme aus den Bestandteilen der Plattform vorgeschrieben wird. In der domänenspezifischen Anforderungsentwicklung werden gemeinsame und variable Anforderungen aller durch die SPL unterstützten Software-Systeme identifiziert, analysiert und als textuelle oder modellbasierte Anforderungen dokumentiert. Im domänenspezifischen Entwurf wird eine Referenz- bzw. Domänenarchitektur für die konkreten Software-Systeme der SPL spezifiziert. Diese umfasst zudem Vorschriften zur Entwicklung von Software-Systemen auf Basis dieser Architektur. Hierbei wird vor allem auf die Konfigurierbarkeit der Architekturkomponenten geachtet, um die Variabilität im Entwurf zu berücksichtigen.

Dabei wird zwischen wiederverwendbaren Komponenten, welche in verschiedenen Software-Systemen Anwendung finden, und applikationsspezifischen Komponenten unterschieden. Auch die Verwendung von existierenden Rahmenwerken, Komponenten, etc. und deren spezifische Anwendung wird dokumentiert. In der domänenspezifischen Implementierung werden wiederverwendbare Komponenten entwickelt oder konfiguriert. Die Komponenten sind dabei unabhängig voneinander und bilden keine eigene Anwendung. Erst in der Anwendungsentwicklung werden die Komponenten zu ausführbaren Software-Systemen zusammengesetzt. Das Ziel der domänenspezifischen Testphase ist die Validierung und Verifikation der wiederverwendbaren Komponenten sowie die Bereitstellung von wiederverwendbaren Testfällen und Testszenarien. Die Ergebnisse der einzelnen Phasen bilden die Bestandteile der domänenspezifischen SPL-Plattform.

Anwendungsentwicklung

Die Anwendungsentwicklung befasst sich mit der Entwicklung von konkreten Software-Systemen auf Basis der domänenspezifischen Software-Plattform. Hierbei soll ein möglichst hoher Grad an Wiederverwendung der bestehenden Bestandteile der domänenspezifischen Plattform durch die Ausnutzung von Gemeinsamkeiten und Festlegung von Varianten erreicht werden. So werden in der Anforderungserhebung einer Anwendung hauptsächlich die Unterschiede zu den in der domänenspezifischen Anforderungserhebung identifizierten Anforderungen ermittelt, um einen hohen Überdeckungsgrad zu ermöglichen. Anwendungsspezifische Anforderungen werden nach wie vor erhoben und, falls sie bei mehreren Anwendungen ebenfalls ermittelt werden, in die Plattform eingearbeitet.

Der Anwendungsentwurf besteht aus der Instanziierung der Referenzarchitektur für die Architektur eines konkreten zu entwickelnden Software-Systems. Dabei werden die benötigten Bestandteile der Referenzarchitektur ausgewählt und konfiguriert und eventuelle Varianten ausgewählt. Während der Implementierung wird ein Software-System aus den benötigten wiederverwendbaren Komponenten, welche ausgewählt und konfiguriert werden, sowie aus weiteren anwendungsspezifischen Komponenten zusammengesetzt. Die wiederverwendbaren Komponenten der Plattform werden dabei so konfiguriert, so dass ihre Varianten festgelegt werden. Die Vorgaben, welche innerhalb der Plattform an die Komponenten gemacht werden, müssen dabei eingehalten werden. Anwendungsspezifische Komponenten werden selbst implementiert oder erworben und fließen in die Plattform mit ein, sofern sie in mehreren Software-Systemen eingesetzt werden.

In der Testphase werden beide Arten von Komponenten und das Software-System gegen ihre Spezifikationen verifiziert. Hierbei wird ein Großteil der Testfälle aus den domänenspezifischen Testfällen abgeleitet. Zusätzlich werden die vorgenommenen Konfigurationen sowie die fest ausgeprägten variablen Bestandteile des Software-Systems bei der Testüberdeckung berücksichtigt.

2.3.3 Funktionsmodelle

Die wesentliche Aktivität der Spezifikation der Software-Plattform einer Software-Produktlinie erfordert die Beschreibung von wiederverwendbaren Entwicklungsartefakten. Mittels Funktionsmodellen (engl. feature models) lassen sich die gemeinsamen und variablen Artefakte einer Plattform beschreiben [CE02:86ff], wobei Funktionsdiagramme (engl. feature diagram [CE02:87]) als Teil eines Funktionsmodells hierzu eine grafische Modellierungsmöglichkeit anbieten. Im Kontext der Domänenentwicklung beschreiben Funktionen wiederverwendbare und konfigurierbare Artefakte, welche auf beliebigen Abstraktionsebenen, zum Beispiel Anforderungen, Entwurfsmodelle, etc., auftreten können [CE02:83]. Im Rahmen dieser Arbeit werden Funktionsmodelle als Ausgangspunkt für die Spezifikation von alternativen Sicherheitsmustern im Beitrag B3 genutzt (siehe Kapitel 6).

Die Notation von Funktionsmodellen erfolgt anhand von Funktionsdiagrammen. Mittels der Diagramme lässt sich die Verfeinerung einer Funktion durch das Aufzeigen von Relationen zu Unterfunktionen spezifizieren. Die Diagramme setzen sich aus Knoten und Kanten zusammen, welche einen Baum bilden. Die Knoten repräsentieren dabei die einzelnen Funktionen, während die Kanten die Relationsart präzisieren. Durch die Anordnung als Baum werden Kindknoten einer Funktion als direkte bzw. indirekte Funktionen bezeichnet [CE02:87f].

Direkte Funktionen können in einer verpflichtenden Beziehung zu dem Elternknoten stehen. Hierdurch müssen bei Auswahl des Elternknotens alle direkten Funktionen ebenfalls umgesetzt werden. Dagegen können direkte Funktionen, welche in einer optionalen Beziehung zum Elternknoten stehen, auch ausgelassen werden. Alternative Funktionen stellen eine Menge von direkten Funktionen dar, aus welchen genau eine Funktion ausgewählt werden muss. Als Erweiterung der alternativen Funktionen können aus disjunkten Funktionen eine beliebige nichtleere Untermenge an Funktionen ausgewählt werden. Abbildung 9 zeigt die verschiedenen Relationen zwischen Funktionen [CE02:88ff].

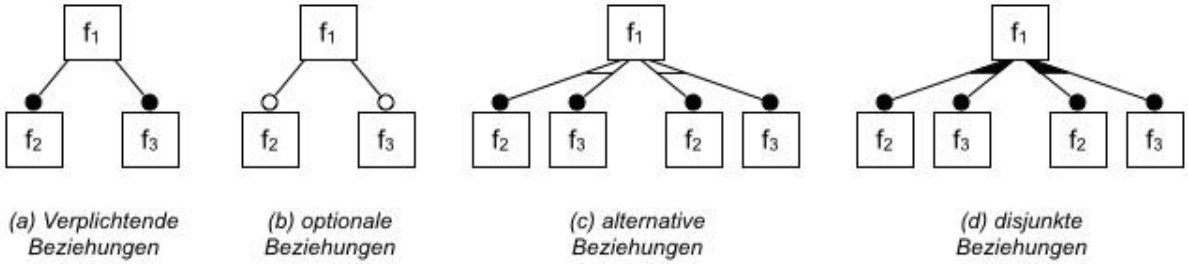

Abbildung 9: Beziehungen zwischen Funktionen

Mittels der Beziehungsarten in den Diagrammen lassen sich Gemeinsamkeiten und Variationen von Funktionen in Funktionsmodellen spezifizieren [CE02:95f]. Die Funktionen, für welche ein Pfad von verpflichtenden Beziehungen bis zur Wurzel des Diagramms existiert, bilden die gemeinsamen Funktionen. Die Unterfunktionen einer Funktion, für welche eine optionale Beziehung auf dem Weg zur Wurzel existiert, bilden die Menge der gemeinsamen Unterfunktionen in dem Fall, dass die Elternfunktion mit einbezogen wird. Variable Funktionen lassen sich mittels alternativen, optionalen sowie disjunkten Beziehungen spezifizieren. Hierdurch werden Variationspunkte im Funktionsdiagramm erzeugt, welche je nach Art der Beziehung unterschiedlich kategorisiert werden können.

2.4 Musterbasierte Software-Entwicklung

Ein wesentlicher Schwerpunkt der vorliegenden Arbeit ist die Aufbereitung von bestehendem Sicherheitswissen, so dass dieses anschließend in einem Software-Entwicklungsprozess zur effizienten Entwicklung von Sicherheitsfunktionalität eingesetzt werden kann. Das im vorangegangenen Abschnitt vorgestellte Paradigma der Software-Produktlinen ordnet die Wiederverwendung von Entwicklungartefakten in den Kontext eines Software-Entwicklungsprozesses ein. Dagegen bietet der Ansatz von Software-Mustern eine Möglichkeit zur Beschreibung von wiederverwendbaren Entwicklungsartefakten, welche im Kontext dieser Arbeit genutzt werden, um bestehendes Sicherheitswissen zu beschreiben. Daher werden im Folgenden die Grundlagen von Mustern sowie der Einsatz von Mustern in der Entwicklung von Software-Systemen betrachtet.

2.4.1 Musterbegriff

Der ursprüngliche Musterbegriff ist zurückzuführen auf die Arbeit des Architekten Christoph Alexander [Al79], welcher Muster als wiederkehrende Stilelement bei der Konstruktion von Gebäude- und Städtearchitekturen beschrieb. Im Bereich der Softwaretechnik wurde durch die Arbeit von Erich Gamma et al. [GH+94] der Musterbegriff auf die Modellierung von objektorientierten Systemen übertragen. In den vergangenen Jahren wurden zahlreiche Muster dokumentiert [BR+99] [SS+00] [KJ04] [BK+07] [BH+07], wodurch die Aktivität und Bedeutung dieses Ansatzes deutlich wird. In diesen Arbeiten werden Muster folgendermaßen definiert:

„A pattern for software architecture describes a particular recurring design problem that arises in specific contexts, and presents a well-proven generic scheme for its solution. The solution scheme is specified by describing its constituent components, their responsibilities, and relationships, and the ways in which they collaborate.“ [BR+99]

Der Einsatz von Mustern in einem strukturierten Software-Entwicklungsprozess dient somit der effizienten Lösung von Entwurfsproblemen, da mittels Mustern bewährte Lösungen zu wiederkehrenden Problemen beschrieben und kommuniziert werden können. Die Effektivität von Mustern ergibt sich somit aus der Beobachtung, dass in der Entwicklung von Software-Systemen heutzutage die Behandlung von bekannten und sich ähnelnden Problemstellungen der Lösung von neuartigen Problemen überwiegt. Muster sind im Allgemeinen eine Lösung zu einem Problem, welches in einem spezifischen Kontext auftritt. Diese im alltäglichen Gebrauch genutzte Aussage ist zutreffend, umfasst jedoch nicht das gesamte Konzept von Mustern. Ohne Hintergrundwissen über die Natur und den Einsatz von Mustern im Software-Entwurf kann es zu negativen Auswirkungen für ein Software-System kommen [BH+07].

Mustereigenschaften

Ausgehend von der zitierten Definition lassen sich neben der Dokumentation von bewährten Problemlösungen weitere Eigenschaften von Mustern ableiten. Zunächst sollte das Problem und die dazu vorgeschlagene Lösung wiederkehrend sein (recurring in der obigen Definition), d.h. es ist für ein Muster allein nicht ausreichend, wenn zu einem Problem eine Lösung existiert. Der zu diesem Problem verwendete Lösungsentwurf muss zudem in verschiedenen Projekten erfolgreich angewendet werden. In dieser Hinsicht können Muster nicht „erfunden“ werden, sondern beschreiben implizites Entwurfswissen von erfahrenen Entwicklern, welches von Entwicklern in anderen Projekten mit ähnlichen Problemstellungen eingesetzt werden kann. Dieses explizite Wissen, welches üblicherweise mittels eines bezeichnenden Namens kurz umschrieben wird, dient dem Aufbau eines gemeinsamen Vokabulars, mittels welchem sich Entwickler austauschen und kommunizieren können sowie sich auf ein gemeinsames Verständnis für Entwurfskonzepte einigen können.

Gleichzeitig wird der Musterbegriff und die Anwendung eines Musters implizit mit einem bestimmten Mehrwert zur Erreichung eines Ziels gleichgesetzt. Ebenso ist allerdings denkbar, dass wiederkehrende Muster einen unausgewogenen Entwurf nach sich ziehen. Diese „schlechten Muster“ (engl. bad patterns, [Al79]) haben unter Umständen keinen, häufig aber einen negativen Einfluss auf einen Entwurf und werden trotzdem aufgrund ihrer Popularität statt eines bewährten Musters (engl. best practice) üblicherweise (engl. common practice) eingesetzt. Aber auch die unsachgemäße Anwendung von Mustern in unpassenden Problemstellungen führt häufig zu einem unausgewogenen Entwurf [BH+07].

Neben der Wiederverwendung in verschiedenen Szenarien lässt sich die Qualität eines Musters mittels verschiedener Faktoren anhand der Dokumentation eines Musters heuristisch untersuchen. Ein wesentlicher Bestandteil des Lösungsteils eines Musters besteht in der Beschreibung eines Vorgehens zur Erstellung eines Artefakts [Al79], wobei im Sinne eines Software-Entwicklungsprozesses das

Artefakt einen Software-Entwurf darstellt und das Vorgehen auf die Lösung des durch das Muster zu lösenden Problems eingeht [BH+07]. Dabei sollte der durch die Lösung beschriebene Entwurf robust sein und das Problem auf bestmögliche Art lösen. Erst die Robustheit einer Lösung ermöglicht die Wiederverwendung der Lösung als Muster [BH+07] [GH+94].

Für die Lösungsartefakte von Mustern ist zu bemerken, dass diese statt konkreter Klassen oder Komponenten, Rollen für diese Entwurfsartefakte aufzeigen sowie die Beziehungen und die Zusammenarbeit zwischen diesen spezifiziert. Dadurch bleibt die Lösung unabhängig von einer Implementierung. Obwohl die bekanntesten Muster objektorientierte Lösungsvorschläge beinhalten [GH+94], sind sie häufig auch auf andere Technologien übertragbar. Generell sollte die Lösung eines Musters so unabhängig oder abhängig wie nötig von einer Implementierungstechnologie spezifiziert sein, um die Essenz eines Musters auszudrücken.

Struktur und Einsatz

Um die Lösung eines Musters zu beschreiben, werden häufig Diagramme eingesetzt, welche die an der Lösung beteiligten Rollen und deren Beziehungen untereinander illustrieren. Es ist wichtig, zwischen dem Muster und dem zur Kommunikation sowie zur Illustration verwendeten Diagramm zu unterscheiden. Ein Diagramm ist in dem Sinne kein essentieller Bestandteil eines Musters. Auch bei der Darstellung des Diagramms ist zwischen einem formalen Ansatz, zum Beispiel mittels der Unified Modeling Language (UML), oder einem informaleren Ansatz zu unterscheiden. Ersterer Ansatz hat den Nachteil, dass ein Diagramm als einzig erlaubte Umsetzungsmöglichkeit für das Muster interpretiert wird. Da Muster jedoch lediglich Rollen für Klassen oder Komponenten spezifizieren, existieren vielfältige Umsetzungsmöglichkeiten für ein Muster. Allerdings muss bei Einsatz einer weniger formalen Spezifikation darauf geachtet werden, dass das Muster unmissverständlich beschrieben wird.

Die Problemstellung, welche von einem Mustern behandelt wird, können meist nicht in Isolation betrachtet werden. Es existieren meist zum Teil widersprüchliche Randbedingungen und Anforderungen an die Lösung. Diese Faktoren nehmen Einfluss auf die Lösungsstruktur eines Musters und werden als Entwurfskräfte (engl. design forces) bezeichnet. Die Entwurfskräfte machen das Problem eines Musters explizit und verdeutlichen den Grund für die vorgeschlagene Struktur der Lösung für ein Problem. Beim Einsatz eines Musters zur Problemlösung in einem Kontext müssen somit die Entwurfskräfte abgewogen werden, um einen unausgeglichenen Entwurf zu vermeiden.

Abgeleitet aus der obigen Definition, ist ein weiterer zentraler Bestandteil eines Musters der Kontext in dem es auftritt. Idealerweise präzisiert der Kontext die Situation, in welcher das Muster angewendet werden sollte [BH+07]. Dies verhindert das Anwenden des Musters in nicht dafür vorgesehene Situationen. Ein zu allgemein formulierter oder zu weit gefasster Kontext birgt die Gefahr, dass verschiedene Entwickler ein Muster aus unterschiedlichen Perspektiven betrachten können und in unangebrachten Situationen einsetzen, wodurch sich gegebenenfalls ein unausgewogener Entwurf ergibt. Gleichzeitig muss allerdings sichergestellt werden, dass der Kontext alle Situationen mit einschließt, in denen ein Muster angewendet werden kann. Eine zu detaillierte Kontextbeschreibung verhindert die Anwendung des Musters.

2.4.2 Beziehungen zwischen Mustern

Während Muster zur Katalogisierung unabhängig voneinander dokumentiert werden, ist zur Entwicklung von Software-Systemen mit Hilfe von Mustern deren kombinierter Einsatz von wesentlicher Bedeutung. Stabile Software-Architekturen tendieren dazu, eine hohe Dichte an Mustern aufzuweisen [BH+07], welches nur durch die Erkennung und Verwendung von Beziehungen zwischen

Mustern möglich ist. Aufdeckung und Beschreibung dieser Beziehungen ist zentraler Aspekt in der Dokumentation und Anwendung von Mustern [GH+94] [BR+99] [Fo04].

Kombination von Mustern

Für die musterbasierte Entwicklung ist es notwendig, Muster als Knoten in einem Netzwerk von Beziehungen zu verstehen. Beziehungen werden durch die Eigenschaft von Mustern ermöglicht, Komponenten im Sinne von Rollen zu spezifizieren. Bei der Komposition können dabei Rollen von verschiedenen Mustern kombiniert und integriert werden, ohne die beteiligten Muster in ihrer Essenz zu beeinträchtigen. Darin liegt auch eine Schwierigkeit in der Entwicklung mittels Mustern, da ohne Leitfaden unerfahrene Nutzer Musterrollen falsch kombinieren [BH+07]. Beispielsweise lässt sich das bekannte Entwurfsmuster Model-View-Controller [GH+94] als Komposition von Rollen aus 12 weiteren Mustern, unter anderem Domain Object, Template View, Transform View, Page Controller, Front Controller, Command, Command Processor, Application Controller, Chain of Responsibility, Wrapper Facade, Data Transfer Object und Observer, darstellen [BH+07].

Bei der Kombination lassen sich komplementäre Muster identifizieren, welche andere Muster auf struktureller Ebene und auf Ebene von Entwurfsentscheidungen ergänzen. Im ersten Fall lässt sich durch die Kombination von Mustern ein verbesserter Entwurf erreichen. Im letzteren Fall stellen verschiedene Muster Alternativen zur Kombination dar, wodurch Entwurfsentscheidungen explizit verdeutlicht werden. Eine solche Entscheidung wird dabei durch den Kontext, die Entwurfskräfte sowie die Konsequenzen eines Musters bestimmt.

Komplementäre Muster, welche in bestimmten Entwurfsfragen wiederholt in einer bestimmten Kombination verwendet werden, können selbst als Muster dokumentiert werden. Solche Musterverbünde (engl. pattern compound) sind benannte, wiederkehrende und kohäsive Kombinationen von Mustern, welche in einem gegenseitigen unterstützenden Zusammenspiel einen Mehrwert liefern. Da die Musterverbindungen selbst Muster darstellen, lösen sie ein bestimmtes Problem in einem festen Kontext. Hierdurch sind sie weniger generisch als die einzelnen in dem Verbund verwendeten Muster, da das Arrangement und die Verwendung der Rollen der einzelnen Muster fest vorgegeben ist.

Mustersequenzen

In einem Entwicklungsprozess werden Muster nicht gleichzeitig sondern sukzessive eingesetzt. In einem Software-Entwurf werden iterativ einzeln auftretende Probleme gelöst. Werden Muster zur Lösung verwendet, ergibt sich eine Kombination und Aneinanderreihung von Mustern, welche einem Erfahrungsbericht (engl. pattern story) zur Erstellung des Entwurfs entsprechen [BH+07]. Diese Berichte beschreiben den sich ergebenen Entwurfsprozess, d.h. sie beschreiben ein Problem, zu welchem Entwurfsfragen gestellt und durch den adäquaten Einsatz von Mustern beantwortet werden. Aus diesen Berichten lassen sich sogenannten Mustersequenzen (engl. pattern sequence) ableiten. Hierbei handelt es sich um eine zeitlich geordnete Folge von Mustern, welche im Laufe der Entwicklung eines Entwurfes eingesetzt wurden. Durch Sequenzen können signifikante Entwurfsentscheidungen und -details dokumentiert werden.

2.4.3 Sicherheitsmuster

Mit der zunehmenden Popularität des Musteransatzes für den fachlichen Software-Entwurf, werden Muster zur Beschreibung von weiteren Bereichen des Software-Entwurfes eingesetzt. Der für diese Arbeit relevante Ansatz der Sicherheitsmuster setzt auf diesen Ansatz auf, um bewährte Lösungsverfahren für Sicherheitsprobleme zu beschreiben. Insbesondere sollen Sicherheitsmuster eingesetzt werden, um von technischen Implementierungen zu abstrahieren und architekturelle Problemstellun-

gen und Lösungen zu spezifizieren. Zudem ermöglicht der Einsatz des Musteransatzes eine Integration in ein Entwicklungsvorgehen, welches bereits fachliche Architektur- bzw. Entwurfsmuster einsetzt, da hierdurch eine gemeinsame Kommunikationsplattform geschaffen wird [SF+05:34].

Begriffsklärung

In [SF+05:31] wird die allgemeine Definition von Muster um den Sicherheitskontext ergänzt:

„A security pattern describes a particular recurring security problem that arises in specific contexts, and presents a well-proven generic solution for it. The solution consists of a set of interacting roles that can be arranged into multiple concrete design structures, as well as process to create one particular such structure"

Im Kontext von Sicherheitsmustern wird somit die Semantik der Beschreibung eines Musters angepasst. Zunächst beschreibt die Problemstellung ein oder mehrere Verletzungen von Sicherheitsrichtlinien auf organisationeller, architektureller und operationeller Ebene. Sicherheitsmuster sind somit nicht nur auf einen Architekturentwurf bzw. auf einen Software-Entwurf eingeschränkt. Hierdurch ändert sich auch die Art der Lösungsbeschreibung. Wie in Kapitel 1 angemerkt, wird der Fokus in dieser Arbeit jedoch auf Sicherheitsmuster gelegt, welche Software-Sicherheit betrachten.

Die Beschreibung der Problemstellung spielt insbesondere bei der Software-Sicherheit eine entscheidende Rolle. Technische Sicherheitslösungen bieten eine Vielzahl an Sicherheitsfunktionalitäten an, welche mehrere Sicherheitsbereiche umfassen. Ohne die Spezifikation des zu lösenden Problems werden diese Sicherheitslösungen unstrukturiert eingesetzt. Hierdurch kann nicht bewertet werden, ob die gewählte Lösung in dem Kontext des Software-Systems passend gewählt wurde.

Die Lösungsbeschreibung eines Sicherheitsmusters stellt anstelle von technischen Lösungen Beschreibungen von Gegenmaßnahmen vor, welche Bedrohungen verhindern bzw. deren Auswirkungen reduzieren sollen. Analog zur Beschreibung des Sicherheitsproblems lassen sich dabei die Lösungen prinzipiell auf verschiedenen Ebenen anwenden. Die Konsequenz der Anwendung einer Sicherheitslösung wird durch die Diskussion der Vor- und Nachteile ergänzt. Da sich der Nachweis von Sicherheit als schwierig gestaltet, wird hierdurch das Verständnis der Lösung vermittelt. Hierzu gehören auch Beispiele für die fehlerhafte Anwendung eines Sicherheitsmusters.

Insbesondere wird durch den Einsatz von Sicherheitsmustern die Verknüpfung von verschiedenen Sicherheitsbereichen verdeutlicht. Sicherheitsprobleme sowie deren Lösungen ziehen weitere Sicherheitsprobleme nach sich. Durch die Verwendung von Sicherheitsmustern lassen sich detaillierte Umsetzungen eines abstrakteren Sicherheitsmusters referenzieren, wodurch die Abhängigkeiten verschiedener Sicherheitsbereiche aufgezeigt werden. Eine solche Verknüpfung ist auf technischer Ebene nur schwer nachzuvollziehen. Somit wird die strukturierte Entwicklung von Sicherheitslösungen unterstützt, da assoziierte Sicherheitsprobleme sowie der Einsatz von geeigneten Lösungen im Vornherein analysiert werden.

Entwicklung mit Sicherheitsmustern

Der Einsatz von Sicherheitsmustern in der Entwicklung von Software-Systemen wird exemplarisch in [Fe04] [FL06] [Sc03] behandelt. Dabei wird zur Veranschaulichung des Mustereinsatzes ein generischer Entwicklungsprozess zugrunde gelegt, welcher in Abbildung 10 dargestellt ist [FL06]. Jedoch wird in den Arbeiten auf den Einfluss von spezifischen Entwicklungsmethoden bei der Entwicklung von sicheren Software-Systemen hingewiesen. Sicherheitsmuster werden eingesetzt, um die Integration und Berücksichtigung von Sicherheitsprinzipien im gesamten Entwicklungsprozess zu ermöglichen [Fe04].

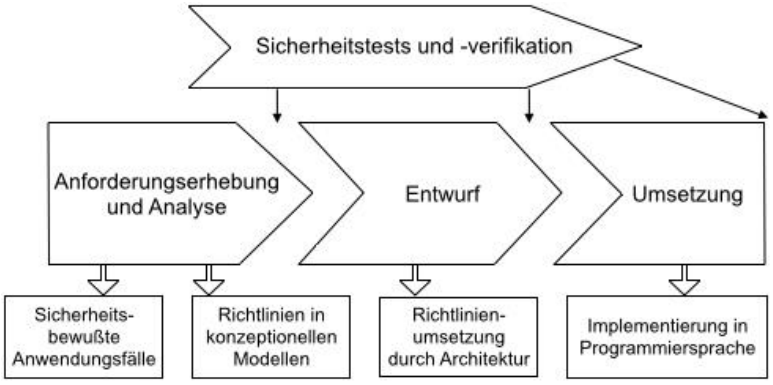

Abbildung 10: Sicherheitsbasierte Entwicklung mittels Sicherheitsmuster

In der Anforderungsanalysephase werden in Anwendungsfällen die Interaktionen von Nutzern mit dem Software-System definiert. Sicherheitsmuster, wie etwa rollenbasierte Zugriffskontrolle [SF+05:249ff], können dabei eingesetzt werden, um die Rechte von Akteuren in einem Anwendungsfall festzulegen. Anwendungsfälle dienen ebenfalls als Grundlage für die Untersuchung von Bedrohungen und Angriffen durch Werkzeuge wie zum Beispiel Missbrauchsanwendungsfälle [SO05], Missbrauchsaktivitäten [BF+08] sowie Sicherheitsanwendungsfälle [Fi03b].

Die ermittelten Rechte können anschließend in der Analysephase durch das Autorisierungsmuster [SF+05:245] umgesetzt werden. Hierbei werden die erstellten konzeptionellen Domänenmodelle erweitert. Der Einsatz von Analysemustern [Fo04] ermöglicht hierbei robuste Domänenmodelle. Die Erweiterung von Analysemustern um Autorisierungsregeln erlaubt somit die effiziente Modellierung von sicheren Domänenmodellen [Fe04] [FY07].

In der Entwurfsphase wird die Sicherheitsarchitektur des zu entwickelnden Software-Systems spezifiziert. Hierzu werden die Komponenten und Dienste, welche durch die Sicherheitsarchitektur bereitgestellt werden, modelliert sowie die Integration mit fachlichen Entwurfsmodellen angegeben. Zahlreiche strukturelle Sicherheitsmuster, wie etwa „Kontrollpunkt“ [SF+05:287] oder „Referenzmonitor“ [SF+05:256], lassen sich zur Umsetzung der Sicherheitsarchitektur einsetzen. In Verteilungsdiagrammen lassen sich Sicherheitsmuster, wie „Sicherer Kanal“ [SF+05:434] und „Firewall“ [SF+05:403ff], einsetzen, um die Ausführungsumgebung von Software-Systemen zu schützen.

In der Implementierung werden plattformspezifische Sicherheitsmuster oder Idiome eingesetzt, wodurch die Umsetzung von Sicherheitslösungen in spezifischen technischen Plattformen vereinfacht wird [CN+05]. Auch lassen sich Sicherheitsmuster für Sicherheitsstandards wie Extensive Access Control Markup Language (XACML [OASIS-XACML-v3.0]) und WS-Security (WSS [OASIS-WSS-v1.1]) einsetzen, wie in [DF05] und [IT03] gezeigt wird. Hierdurch wird der Einsatz dieser komplexen Standards handhabbar gemacht. Bei der Programmierung lassen sich sicherheitskritische Fehlerquellen mittels Sicherheitsmuster bzw. Idiome unterbinden, indem diese in statische Analysewerkzeuge integriert werden. Diese Werkzeuge untersuchen Quelltexte auf entsprechende Fehlerquellen und geben Hinweise auf bessere Strukturen.

Der Einsatz von Sicherheitsmustern in der Entwicklung von sicheren Software-Systemen ist somit praktikabel. Für die strukturierte Nutzung ist das beschriebene Vorgehen jedoch zu generisch und unpräzise. Vor allem der Zusammenhang zwischen den Sicherheitsmustern innerhalb einer Phase sowie phasenübergreifend bleibt ungeklärt. Andere Ansätze wie etwa [FP+08] [PF+04] [HH+07] greifen diese Problematik auf und schaffen mit Mustersystemen für verschiedene Sicherheitsbereiche wie Zugriffskontrolle und Authentifizierung Strukturen, welche sich an Mustersprachen anlehnen.

3 Stand der Forschung

Trotz der zahlreichen entwickelten Sicherheitsmodelle und -technologien ist die strukturierte und methodische Integration von Verfahren zur Entwicklung sicherer Software-Systeme in einen Software-Entwicklungsprozess eine noch relativ junge Disziplin [An08] [Ec09:168]. Insbesondere wird der Einsatz von modernen Software-Entwicklungsparadigmen, wie etwa Dienstorientierung und Software-Produktlinien, vernachlässigt, obwohl diese die effiziente Entwicklung von komplexen Systemen durch intensive Wiederverwendung von existierender Funktionalität in den Fokus stellen.

Im Rahmen dieser Arbeit wird der Einsatz dieser Paradigmen zur Integration von existierenden Sicherheitsmodellen und -technologien in Software-Entwicklungsprozesse untersucht. Die zu diesem Thema relevanten Ansätze werden in diesem Kapitel vorgestellt und im Rahmen der an diese Arbeit gestellten Anforderungen diskutiert. Hierzu werden die Anforderungen selbst zunächst vorgestellt und erläutert. Diese stellen zudem die Grundlage dar, anhand welcher die in dieser Arbeit vorgestellten Beiträge bewertet werden. Anschließend werden die bestehenden Arbeiten und Ansätze beschrieben und kritisch analysiert. Die abschließende Bewertung anhand der gestellten Anforderungen bildet schließlich den Ausgangspunkt für den Handlungsbedarf und somit die Motivation für die in den folgenden Kapiteln ausgeführten Beiträge der vorliegenden Arbeit.

3.1 Anforderungskatalog

Die im Rahmen dieser Arbeit untersuchte Integration von sicherheitsbasierter und fachlicher Software-Entwicklung wird unter dem Gesichtspunkt der im Folgenden vorgestellten Anforderungen betrachtet. Diese werden zum einen aus bestehenden Arbeiten abgeleitet, welche sich ebenfalls mit dieser Fragestellung auseinandergesetzt haben. Zum anderen werden auch allgemeine Anforderungen gestellt, welche sich aus der Praxis der Software-Entwicklung ergeben.

Wie in Tabelle 1 dargestellt, werden die Anforderungen dabei in vier Kategorien unterteilt. Die Kategorie „Integrationsfähigkeit“ enthält Anforderungen, welche sich mit der integrativen Entwicklung von Sicherheitsfunktionalität und Fachfunktionalität beschäftigen. Die Kategorie „Nachvollziehbarkeit“ betrachtet Anforderungen bezüglich der Entwicklung von Sicherheitsartefakten. Die Kategorie „Wiederverwendbarkeit“ umfasst Anforderungen, welche die Berücksichtigung von existierendem und neuem Sicherheitswissen bei der Software-Entwicklung betreffen. Letztlich betrachtet die Anforderungskategorie „Praktikabilität“ die Anwendbarkeit von Prozessen und Werkzeugen der sicherheitsbasierten Software-Entwicklung. Die einzelnen Anforderungen werden im Folgenden im Detail erläutert.

Anforderungskategorie 1 (AK 1): Integrationsfähigkeit

Ein sicherheitsbasiertes Entwicklungsvorgehen stellt eine Erweiterung eines fachlichen Software-Entwicklungsprozesses dar [An08:3] und ist dabei weitestgehend unabhängig von einem konkret eingesetzten Entwicklungsprozess. Die zusätzlich bereitgestellten Prozesse, Methoden, Aktivitäten und Werkzeuge sollten sich daher in beliebige fachliche Entwicklungsprozesse integrieren lassen. Die Erweiterung eines fachlichen Entwicklungsprozesses erhöht den Aufwand der durchzuführenden Aktivitäten. Eine effiziente Integration hat somit zum Ziel, die Durchführung der sicherheitsbezogenen Aktivitäten zu optimieren und den damit verbundenen Aufwand zu minimieren. Dies führt zu einer erhöhten Akzeptanz der sicherheitsbezogenen Entwicklungsaktivitäten, wodurch diese in der Praxis eingesetzt werden.

Tabelle 1: Anforderungskatalog

Anforderungskategorie	Kriterien
AK1: Integrationsfähigkeit	• Unabhängig von einem fachlichen Entwicklungsprozess • Anpassungsfähigkeit
AK 2: Nachvollziehbarkeit	• Trennung der Entwicklungsphasen • Abbildung zwischen Entwicklungsphasen • Verfeinerung von Entwurfsartefakten
AK 3: Wiederverwendbarkeit	• Definition von Sicherheitskonzepten • Kategorisierung von Sicherheitslösungen • Entscheidungsunterstützung bei Auswahl von Alternativen • Berücksichtigung von existierender Sicherheitsinfrastruktur
AK 4: Praktikabilität	• Nutzung von standardisierten Sprachen • Werkzeugunterstützung • (Semi-) Automatisierte Entwicklungsunterstützung

Um eine effiziente Integration zu erreichen, ist es zum einen notwendig, dass die sicherheitsbezogenen Entwicklungsaktivitäten und -artefakte unabhängig von einem spezifischen Entwicklungsprozessmodell sind. Aufgrund der unterschiedlichen Eigenschaften von Prozessmodellen erschwert eine Kopplung die Migration der sicherheitsbezogenen Entwicklungsaktivitäten in andere Prozessmodelle. Eine Unabhängigkeit manifestiert sich dabei durch die flexible Unterstützung von beliebigen Entwicklungsphasen, -aktivitäten und -prozessen, welche durch ein Prozessmodell vorgegebenen werden.

Um die geforderte Integrationsfähigkeit zu ermöglichen, bedingt diese Unabhängigkeit zum anderen die Anpassungsfähigkeit des sicherheitsbezogenen Entwicklungsvorgehens an spezifische Entwicklungsprozessmodelle. Durch die Anpassung wird eine Einbettung der Entwicklungsaktivitäten und -artefakte an ein spezifisches Prozessmodell vorgenommen, so dass eine effiziente Software-Entwicklung nach wie vor ermöglicht wird. Die Anpassungsfähigkeit wird durch eine detaillierte Beschreibung der bereitgestellten Entwicklungsaktivitäten und -artefakte ermöglicht, so dass eine Einordnung in einen fachlichen Entwicklungsprozess vorgenommen werden kann.

Anforderungskategorie 2 (AK 2): Nachvollziehbarkeit

Wie bei der fachlichen Funktionalität eines Software-Systems, ist eine nachvollziehbare Entwicklung der Sicherheitsfunktionalität erforderlich. Die Anforderung der Nachvollziehbarkeit bezieht sich in diesem Kontext auf die Plausibilität von Entscheidungen im Entwicklungsprozess, welche zur Spezifikation und zum Einsatz von sicherheitsbezogenen Entwicklungsartefakten führen.

Hierzu ist zunächst aufzuzeigen, zu welchen generellen Entwicklungsphasen die bereitgestellten Entwicklungsaktivitäten und -artefakte zuzuordnen sind. Dabei sollten alle relevanten Entwicklungsphasen unterstützt werden, um eine durchgängige Entwicklung der sicherheitsbezogenen Artefakte zu ermöglichen. Ebenso ist es notwendig, eine strikte Trennung bei der Phasenzugehörigkeit der Aktivitäten und der Artefakte einzuhalten. Dies erlaubt die Modularisierung der sicherheitsbezogenen Entwicklungsaktivitäten und -artefakte gemäß ihrer Phasenzugehörigkeit.

Des Weiteren sollte es möglich sein, den Kontext der Entscheidung für den Einsatz von Entwicklungsartefakten zu spezifizieren. Hierzu ist die zu lösende Sicherheitsproblematik während der Entwicklung festzuhalten und die Begründung für die Auswahl eines oder mehrerer sicherheitsbezogener Entwicklungsartefakte zu dokumentieren. Dies ermöglicht es, bei der Evolution der Sicherheitsfunktionalität die Aktualität der Entscheidung zu revidieren und die Entscheidung gegebenenfalls zu ändern. Ebenso ist die Verwendung und der Einsatz der Entwicklungsartefakte eindeutig zu spezifizieren. Hierdurch können die Auswirkungen der sicherheitsbezogenen Entwicklungsartefakte verdeutlicht werden. Ein sicherheitsbasiertes Entwicklungsvorgehen sollte die Entscheidungsfindung dabei unterstützen.

Aufgrund der Trennung zwischen den Entwicklungsphasen sollte die Überführung der Entwicklungsartefakte zwischen den Phasen präzise beschrieben werden. Dies ermöglicht sowohl eine strukturierte und systematische Erstellung von weiteren Entwicklungsartefakten als auch die Rechtfertigung von bereits erstellten Artefakten. Eine ähnliche Anforderung lässt sich auf die iterative Verfeinerung von Entwicklungsartefakten innerhalb einer Phase übertragen, zum Beispiel die Spezialisierung von Entwurfsmodellen der Sicherheitslösungen. Die bei der Verfeinerung getroffene Auswahl von sicherheitsbezogenen Entwicklungsartefakten sollte wiederum nicht willkürlich, sondern anhand begründeter Entscheidungen durchgeführt werden, welche dem Schutzbedarf entsprechen bzw. den Sicherheitsanforderungen des Software-Systems genügen.

Anforderungskategorie 3 (AK 3): Wiederverwendbarkeit

Zur effizienten Entwicklung von Sicherheitslösungen für ein Software-System sollte bestehendes Sicherheitswissen berücksichtigt werden. Wie bereits dargelegt wurde, existiert eine Vielzahl an Sicherheitsmodellen, -standards und -technologien, welche anerkannte Lösungen für bereits bekannte Sicherheitsprobleme bereitstellen. Die Verwendung dieses Sicherheitswissens ist eine wesentliche Unterstützung bei der effizienten Entwicklung von Sicherheitslösungen. Hierzu ist es jedoch erforderlich, dass das Wissen in einer angemessenen Form aufbereitet wird, so dass es in einem Software-Entwicklungsprozess eingesetzt werden kann.

Dies setzt zum einen die Definition der verwendeten Sicherheitskonzepte voraus, um das bestehende Sicherheitswissen einzuordnen und Missverständnisse über die verwendeten Begriffe zu vermeiden. Zum anderen wird die Kategorisierung von verschiedenen Sicherheitsbereichen benötigt, so dass Sicherheitslösungen eindeutig zu Sicherheitsproblemen zugeordnet werden können.

Hinzu kommt, dass eine Vielzahl von möglichen Sicherheitslösungen zu einem bestimmten Sicherheitsproblem existieren. Somit stehen einem Entwickler mehrere Alternativen zur Verfügung, um ein gegebenes Sicherheitsproblem zu lösen. Diese Alternativen sind somit zunächst aufzuzeigen und eine Unterstützung bei der Auswahl anzubieten. Letzteres ist insbesondere angesichts der enorm großen Menge an bestehendem Sicherheitswissen notwendig.

Eine Einschränkung des Umfangs des aufzubereitenden Sicherheitswissen wird durch die Berücksichtigung von bereits existierenden Sicherheitsinfrastrukturen oder Sicherheitsprodukten ermöglicht. Vor allem im Unternehmenskontext existieren gewachsene Infrastrukturen zur Lösung von Sicherheitsproblemen. Eine Berücksichtigung dieser Sicherheitslösungen im Entwicklungsprozess führt zum einen zur Entwicklung von Software-Systemen, welche an die Sicherheitsinfrastruktur eines

Unternehmens angepasst sind. Zum anderen lässt sich durch die Vorgabe von Sicherheitslösungen die Entwicklung der Sicherheitsfunktionalität eines neuen Software-Systems effizient gestalten.

Anforderungskategorie 4 (AK 4): Praktikabilität

Die Praktikabilität ist eine weitere wichtige Anforderung für ein sicherheitsbasiertes Entwicklungsvorgehen. Die eingesetzten Prozesse, Methoden und Werkzeuge sollten die Entwicklung von Sicherheitslösungen für Software-Systeme in einer strukturierten und methodischen Art unterstützen. Hierfür ist zum einen die Nutzung von verbreiteten und standardisierten Sprachen zur Spezifikation von Entwicklungsartefakten notwendig. Dies ermöglicht den Einsatz von gängigen Entwicklungswerkzeugen, welche bereits zur Entwicklung von fachlicher Funktionalität eingesetzt werden. Hierdurch wird wiederum die Akzeptanz der sicherheitsbasierten Entwicklung gefördert und die Integration in einen Entwicklungsprozess erleichtert. Des Weiteren sollte die Spezifikation der Entwicklungsartefakte den auszudrückenden Sachverhalt der Sicherheit präzise wiedergeben, um eventuelle Missverständnisse zu vermeiden, da dies im Kontext von Sicherheitsfunktionalität besonders kritisch ist.

Im Entwicklungsprozess beziehen sich Sicherheitsaspekte auf schützenswerte fachliche Ressourcen, wodurch die sicherheitsbasierte Entwicklung in einer Abhängigkeitsbeziehung zu einer fachlichen Entwicklung steht. Somit müssen auch die zur Entwicklung von sicherheitsrelevanten Artefakten genutzten Prozesse und Werkzeuge nachvollziehbar integriert werden. Dabei sollten diese zusätzlichen Prozesse die fachliche Entwicklung nicht beeinträchtigen, da diese in den meisten Fällen im Vordergrund steht. Somit ist es zum Beispiel wünschenswert, während des Entwicklungsprozesses eine teilweise bis vollständig automatische Generierung von Sicherheitsartefakten zu ermöglichen.

3.2 Übersicht und Bewertung bestehender Arbeiten

Die in dieser Arbeit behandelten Fragestellungen zur Integration und Wiederverwendung von Sicherheitswissen in einem sicherheitsbasierten Entwicklungsvorgehen wurden teilweise in verschiedenen wissenschaftlichen Ansätzen behandelt. Entsprechend der Ziele und der Beiträge der vorliegenden Arbeit wurde daher eine Auswahl von Ansätzen untersucht, welche vergleichbare Absichten verfolgen. Diese werden im Folgenden vorgestellt und anhand der im vorherigen Abschnitt dargestellten Anforderungen untersucht. In Anlehnung an die Beiträge dieser Arbeit wurden Ansätze ausgewählt, welche moderne Software-Entwicklungsparadigmen einsetzen, um eine strukturierte Entwicklung von Sicherheitsartefakten in einem Software-Entwicklungsprozess unter der Wiederverwendung von bestehendem Sicherheitswissen zu ermöglichen. In der Vorstellung werden die Ansätze dabei nicht in ihrer Vollständigkeit beschrieben, sondern nur die Teile betrachtet, welche den Zielen dieser Arbeit entsprechen.

Der Beitrag B1 dieser Arbeit umfasst ein sicherheitsbezogenes Entwicklungsverfahren für die Analyse und den Entwurf von Sicherheitsmaßnahmen, in welchem die Wiederverwendung von Sicherheitswissen in Form von Entwicklungsartefakten eine zentrale Rolle spielt. Ein vergleichbarer Einsatz von wiederverwendbaren Entwicklungsartefakten in allen Entwicklungsphasen wird insbesondere im Paradigma der Software-Produktlinien genutzt, um eine effiziente Software-Entwicklung zu ermöglichen. Im Folgenden werden daher Ansätze vorgestellt, in welchen die Entwicklung von Sicherheitsmaßnahmen für Software-Produktlinien behandelt wird. Dabei wird jedoch vorrangig die Wiederverwendung der Entwicklungsartefakte untersucht. Der Bezug zum Software-Produktlinienparadigma wird in der Untersuchung aufgrund der gestellten Anforderungen vernachlässigt.

Mit Sicherheitsanforderungsvorlagen als Beitrag B2 und einem Variabiliätsmodell für Sicherheitslösungen im Beitrag B3 dieser Arbeit werden Werkzeuge zur Aufbereitung und zur Verwendung von sicherheitsbezogenen Entwicklungsartefakten betrachteten. Mit Missbrauchs- und Sicherheitsanwendungsfällen von Sindre und Opdahl bzw. Firesmith werden daher Ansätze untersucht, welche ebenfalls die Strukturierung von wiederverwendbaren Sicherheitsanforderungen betrachten. Sicherheitsmuster stellen eine weit verbreitete Form der Beschreibung von anerkannten Entwurfslösungen zu Sicherheitsproblemen dar. Der strukturierte Einsatz von Sicherheitsmustern ist Kernthema des Beitrags B3, weshalb mit den Musterdiagrammen von Fernandez et al. sowie der Referenzarchitektur von Fægri und Hallenstein relevante Ansätze untersucht werden.

3.2.1 Mellado et al.: Entwicklungsprozess für Sicherheitsanforderungen in Software-Produktlinien

Mit ihren Arbeiten tragen Mellado et al. zu einem holistischen Ansatz zur Analyse von Sicherheitsanforderungen in Kontext von Software-Produktlinienentwicklung bei [MF+10] [MF+08] [MF+08a] [MF+09]. Dabei erweitern sie ihre generischen Beiträge zu Sicherheitsanforderungsanalyse [MF+06]. Der Ansatz beinhaltet ein Rahmenwerk, welches einen zweigeteilten abstrakten Prozess zur Analyse von Sicherheitsanforderungen sowie ein zugrunde liegendes Sicherheitsreferenzmodell für die im Prozess entwickelten Sicherheitsartefakte umfasst. Wesentlicher Fokus des Ansatzes ist zudem die Berücksichtigung von Sicherheitsstandards, welche Vorgaben für Sicherheitsanforderungen in Software-Systemen bereitstellen. Dies hat eine Zertifizierung der zu erhebenden Sicherheitsanforderungen gegenüber diesen Standards zum Ziel.

Ergebnisse des Ansatzes

Ein Resultat der Unterstützung von Software-Produktlinien ist die Zweiteilung des Anforderungsanalyseprozesses. Der als Domänenanalyse von Sicherheitsanforderungen benannte Teil des Entwicklungsprozesses (engl. Product Line Security Domain Requirements Engineering Process [MF+09]) fokussiert die Entwicklung von wiederverwendbaren Sicherheitsanforderungen. Diese repräsentieren Sicherheitsanforderungen, welche in einer Menge von gleichartigen Software-Produkten einer fachlichen Domäne vorhandenen sind. Dabei wird zwischen gemeinsamen und variablen Sicherheitsanforderungen unterschieden. Gemeinsamkeiten repräsentieren Sicherheitsanforderungen, welche in allen Software-Produkten zum Einsatz kommen. Variable Anforderungen dagegen werden nur in einigen Software-Produkten berücksichtigt.

In der Domänenanalyse wird zunächst der Umfang der zu betrachtenden Sicherheitsanforderungen spezifiziert. Anschließend werden die zu schützenden Ressourcen analysiert und eine erste Unterscheidung zwischen gemeinsamen und variablen Ressourcen vorgenommen. Darauf folgend werden die Schutzziele festgelegt, welche durch die wiederverwendbaren Sicherheitsanforderungen umgesetzt werden sollen. Als nächstes wird eine typische Bedrohungs- und Risikoanalyse durchgeführt, durch welche mögliche Bedrohungen und Angriffe, sowie deren Risiko spezifiziert werden. Im Anschluss wird der Umfang der Sicherheitsanforderungen festgelegt und die Anforderungen priorisiert. Die abschließenden Aktivitäten umfassen die Spezifikation der wiederverwendbaren Sicherheitsanforderungen sowie deren Verifizierung.

Die wiederverwendbaren Sicherheitsanforderungen werden im zweiten Teil des Prozesses zur Analyse von Sicherheitsanforderungen eines konkret zu entwickelnden Software-Produktes einer Software-Produktlinie eingesetzt (engl. Product Line Security Application Requirements Engineering Process [MF+09]). Hierzu werden die gemeinsamen Sicherheitsanforderungen instanziiert und gegebenenfalls geringe Anpassungen vorgenommen. Ebenso wird der Einsatz von variablen Sicherheitsanforderungen

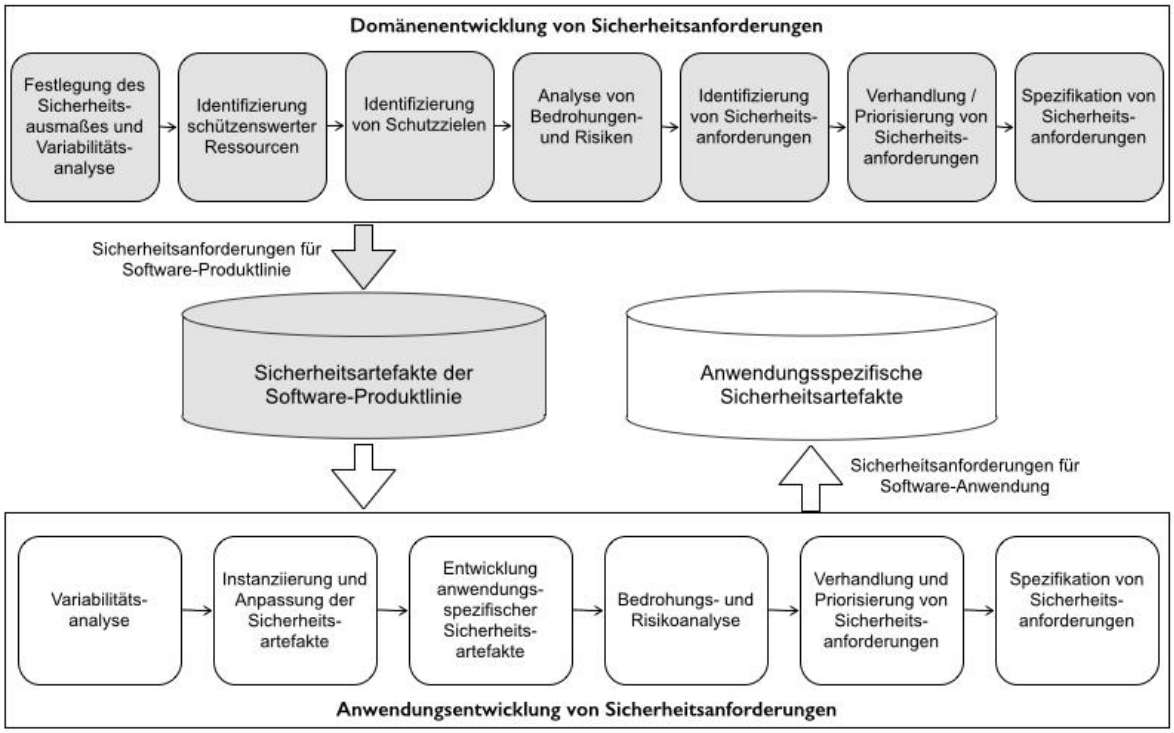

Abbildung 11: Analyseprozesse für Sicherheitsanforderungen in Software-Produktlinien [MF+09]

überprüft. Anschließend werden die Anforderungen festgelegt und priorisiert sowie spezifiziert. Der Ablauf der beiden Prozesse ist in Abbildung 11 dargestellt.

Das in Abbildung 12 dargestellte Sicherheitsreferenzmodell ist die Grundlage für die Spezifikation von wiederverwendbaren Sicherheitsanforderungen in der Domänenanalyse sowie deren Instanziierung in der Anforderungsanalyse eines konkreten Software-Systems. Im Referenzmodell werden zunächst die konzeptionellen Elemente der Sicherheitsanforderungsanalyse und deren Beziehungen untereinander spezifiziert. Zudem wird eine Unterscheidung von gemeinsamen und variablen Sicherheitsanalyseartefakten unterstützt. Für das Variabilitätsmetamodell wird das von Pohl et al. definierte Variabilitätsmodell für Software-Produktlinien erweitert [PB+05]. Das Metamodell für Sicherheitsanforderungsentscheidungen erweitert das Variabilitätsmodell um zusätzliche Sicherheitsartefakte. Hierbei werden die wesentlichen Konzepte zur Bedrohungsanalyse, wie zum Beispiel Schutzziele, Ressourcen, Bedrohungen und Sicherheitsanforderungen, festgelegt. Ebenso wird ein Abgleich mit bestehenden Standards zur Sicherheitsanforderungsanalyse wie zum Beispiel dem Common-Criteria-Katalog [CC09] und dem ISO 27001-Standard [ISO-27001] vorgenommen.

Bewertung des Ansatzes

Der von Mellado et al. gewählte Ansatz fokussiert die Wiederverwendung von bereits eingesetzten Sicherheitsanforderungen. Durch den Einsatz der Domänenentwicklung werden gemeinsame und variable Sicherheitsanforderungen definiert und für zukünftige Software-Entwicklungsprozesse bereitgestellt. Hierdurch wird das hierin enthaltene Wissen bezüglich Sicherheitsanforderungen für eine Wiederverwendung aufbereitet. Ebenso ermöglicht der Bezug zu geeigneten Standards die Zertifizierung der Sicherheitsanforderungen. Leider wird in diesem Kontext ein Top-Down-Ansatz zur Aufbereitung der Sicherheitsanforderungen eingesetzt, so dass die bereits in den Software-Systemen der Produktlinie vorhandenen Sicherheitsanforderungen nicht berücksichtigt werden [MF+10].

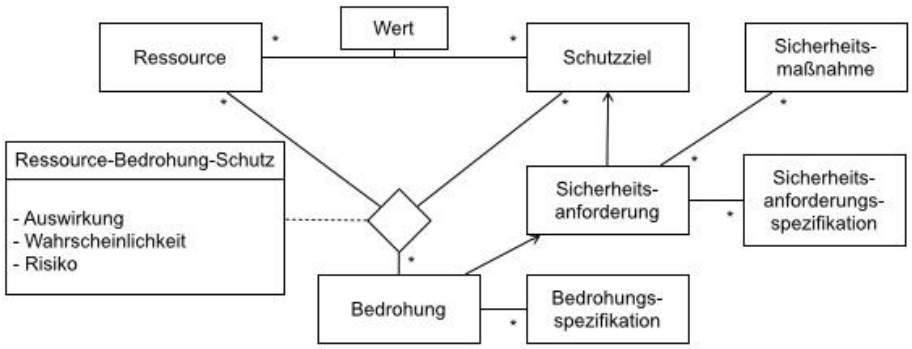

Abbildung 12: Metamodell für die Spezifikation von Sicherheitsanforderungen [MF+09]

Hierdurch bleiben einerseits die Erfahrungen mit den Sicherheitsanforderungen unberücksichtigt, während andererseits eine Diskrepanz zwischen den Top-Down definierten Anforderungen und der vorhandenen Sicherheitsfunktionalität einer Referenzarchitektur entsteht.

Die Sicherheitsanforderungen können in dem vorgeschlagenen Ansatz in beliebigen Formaten, zum Beispiel mittels Sicherheitsanwendungsfällen, spezifiziert werden. Jedoch beziehen sie sich auf in diesem Bereich existierende Standards. Es lässt sich argumentieren, dass in diesen Standards eine explizite Trennung zwischen Anforderungen und Umsetzungen nicht immer eindeutig gegeben ist [HL+08] [TJ+08]. Somit ist eine Trennung zwischen der Analysephase und der Entwurfsphase nicht gegeben, wodurch die Nachvollziehbarkeit der Sicherheitsanforderungen eingeschränkt wird.

Des Weiteren beschreiben Mellado et al. ihr Prozessmodell als flexibel, in dem Sinne, dass es auf konkrete Entwicklungsprozesse angepasst werden kann. Hierdurch soll auch ein leichtgewichtiger Entwicklungsprozess unterstützt werden. Jedoch wird der Ansatz spezifisch für die Software-Produktlinienentwicklung beschrieben, wodurch eine Unterstützung und Integration für Entwicklungsprozesse, welche nicht diesem Paradigma folgen, nicht vorausgesetzt werden kann.

3.2.2 Sindre und Opdahl: Wiederverwendbare Missbrauchs- und Sicherheitsanwendungsfälle

In Ergänzung zu ihrem Ansatz der Missbrauchsanwendungsfälle in [SO05] stellen Sindre et al. eine Methode zur Spezifikation von wiederverwendbaren Bedrohungen und Sicherheitsanforderungen vor [SF+03]. Die Motivation für diesen Ansatz liegt in der Beobachtung, dass Software-Systeme trotz unterschiedlicher funktionaler Anforderungen ähnlichen Bedrohungen ausgesetzt sind und somit ähnliche Kategorien von Sicherheitsmaßnahmen benötigen. Das Ziel des Ansatzes liegt im Aufbau und der Verwaltung von gemeinsamen wiederverwendbaren Sicherheitsanforderungen, durch welche die Qualität der Sicherheitsmaßnahmen in einem Software-System verbessert wird. Durch die Wiederverwendung der sicherheitsbezogenen Entwicklungsartefakte werden dabei eine fehlerhafte Spezifikation sowie verfrühte Entwurfsentscheidungen verhindert [SF+03].

Ergebnisse des Ansatzes

In dem vorgestellten Anforderungserhebungsvorgehen wird zwischen der Entwicklung von wiederverwendbaren Missbrauchs- und Sicherheitsanwendungsfällen (engl. development for reuse) und deren Einsatz in der Anwendungsentwicklung (engl. development with reuse) unterschieden. Für die Entwicklung von wiederverwendbaren Bedrohungen und Sicherheitsanforderungen werden Vorlagen

zur Beschreibung der Entwicklungsartefakte vorgestellt. Ebenso wird eine Struktur für einen Katalog beschrieben, in welchem die Artefakte zur deren Verwaltung abgelegt werden können. Die Vorlagen werden dabei in generische sowie anwendungsspezifische Bedrohungen und Sicherheitsanforderungen eingeteilt. Die generischen Entwicklungsartefakte sind dabei unabhängig von spezifischen fachlichen Domänen und können somit in verschiedenen Software-Systemen aus unterschiedlichen Domänen eingesetzt werden. Dagegen werden anwendungsspezifische Entwicklungsartefakte um spezifische Gegebenheiten einer fachlichen Domäne ergänzt, wodurch sie auf Software-Systeme dieser Domäne eingeschränkt werden.

Tabelle 2: Generischer Missbrauchsanwendungsfall „Zugriff vortäuschen“ [SF+03]

<table>
<tr><td colspan="2">Generischer Missbrauchsanwendungsfall: Zugriff vortäuschen</td></tr>
<tr><td colspan="2">Zusammenfassung: Der Angreifer gibt sich erfolgreich dem (physischen / menschlichen / rechnergestützten) System als ein legitimer Nutzer aus, wodurch er Zugriff auf ein zugangsbeschränktes System / Dienst / Ressource / Gebäude erhält.</td></tr>
<tr><td colspan="2">Vorbedingungen:
1) Der Angreifer ist in Besitz von Mitteln zur Identifikation und Authentifizierung eines legitimen Nutzers ODER
2) Der Angreifer ist in Besitz von ungültigen Mitteln zur Identifikation und Authentifizierung, welche jedoch so ähnlich zu legitimen Mitteln sind, dass das System diese nicht unterscheiden kann ODER
3) Das System wurde so beschädigt, dass es Mittel zur Identifikation und Authentifizierung akzeptiert, welche es im Normalfall zurückweisen würde (siehe Missbrauchsanwendungsfall „Systemmanipulation“)</td></tr>
<tr><td>Angreifer</td><td>System</td></tr>
<tr><td>Anfrage an System / Dienst</td><td></td></tr>
<tr><td></td><td>Abfrage von Identifikation und Authentifizierung</td></tr>
<tr><td>Vorweisen ungültiger Identifikation</td><td></td></tr>
<tr><td></td><td>Zugriff gewähren / Dienst bereitstellen</td></tr>
<tr><td colspan="2">Nachbedingungen:
1) Der Angreifer ist in der Lage Aktionen durchzuführen, welche auch ein legitimer Nutzer in einer Zugriffssitzung durchführen kann UND
2) Im Log des Systems wird der Zugriff als ein Systemzugriff durch einen legitimen Nutzer</td></tr>
</table>

Der Aufbau der generischen Missbrauchs- und Sicherheitsanwendungsfälle ist an Anwendungsfallbeschreibungen angelehnt. Jedoch werden die Intentionen von Benutzern anstelle von konkreten Interaktionen fokussiert. Neben einer Beschreibung des Artefakts werden Vor- und Nachbedingungen angegeben, welche vor der Ausführung der Interaktionen erfüllt sein müssen bzw. nach den Interaktionen gelten. Die Interaktionen selbst beschreiben die Aktionen von menschlichen Nutzern und dem untersuchten Software-System. Der Aufbau der generischen Entwicklungsartefakte ist in Tabelle 2 am Beispiel des Missbrauchsanwendungsfalles „Zugriff vortäuschen" sowie in Tabelle 3 für den zugehörigen Sicherheitsanwendungsfalle „Zugriffskontrolle" dargestellt.

Tabelle 3: Generischer Sicherheitsanwendungsfall „Zugriffskontrolle" [SF+03]

Generischer Sicherheitsanwendungsfall: Zugriffskontrolle	
Zusammenfassung: Ungültige Authentifizierung zurückweisen	
Vorbedingungen: 1) Der Angreifer ist in Besitz von Mitteln zur Identifikation jedoch nicht zur Authentifizierung	
Angreifer	**System**
	Abfrage von Identifikation und Authentifizierung
Gültige Identifikation mit ungültiger Authentifizierung vorweisen	
	Identifikation, Authentifizierung und Autorisierung durchführen
	Zugriff für Angreifer zurückweisen
Nachbedingungen: 1 Der Angreifer ist in Besitz von Mitteln zur Identifikation jedoch nicht zur Authentifizierung **UND** 2) Angreifer wurde nicht authentifiziert und der Zugriff zurückgewiesen **UND** 3) Ein Zugriffskontrollfehler wurde gemeldet	

Für die Katalogisierung der wiederverwendbaren Entwicklungsartefakte wird anhand eines Metamodells eine Ablagestruktur vorgestellt. In diesem wird die Beziehung zwischen den Elementen „Bedrohung" (engl. threat) und „Sicherheitsanforderungen" definiert. Hierbei wird eine indirekte Beziehung über das Element „Sicherheitsanforderungspaket" definiert, da nach Aussage der Autoren eine Bedrohung selten durch eine einzelne Sicherheitsanforderung abgedeckt wird. Ebenfalls wird die Einordnung der Entwicklungsartefakte durchgeführt, welche im Metamodell eine Form der Spezifikation von Bedrohungen bzw. Sicherheitsanforderungen darstellen. Durch die Zuordnung zu den Elementen „Generische Bedrohung" bzw. „Anwendungsspezifische Bedrohung" wird das Abstraktionsniveau eines Missbrauchsanwendungsfalles festlegt. Diese Zuordnung wird analog für Sicherheitsan-

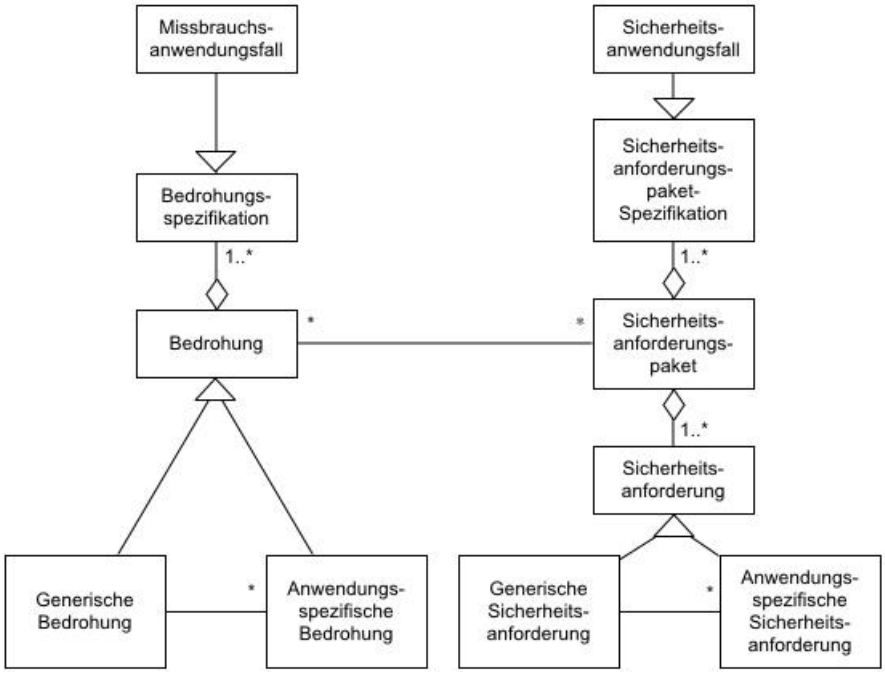

Abbildung 13: Metamodell für wiederverwendbare Missbrauchs- und Sicherheitsanwendungsfälle [SF+03]

forderungen durch entsprechende Metamodellelemente durchgeführt. Das Metamodell ist in Abbildung 13 dargestellt.

Für den Einsatz und die Einordnung der wiederverwendbaren Entwicklungsartefakte in der Erhebung von Sicherheitsanforderungen wird ein generischer Prozess vorgestellt, welcher dem in Kapitel 2.3 vorgestellten Prozess ähnelt. Die generischen und anwendungsspezifischen Missbrauchsanwendungsfälle werden zunächst zur Unterstützung der Identifikation und Spezifikation von möglichen Bedrohungen für ein Software-System bzw. für die darin verwalteten Ressourcen eingesetzt. Hierbei wird zum einen ein Top-Down-Vorgehen unterstützt, in welchem ausgehend von den für die Ressourcen definierten Schutzziele der Katalog auf mögliche Bedrohungen durchsucht wird. Zum anderen wird ein Bottom-Up-Vorgehen beschrieben, in welchem der Katalog unabhängig von spezifischen Ressourcen auf relevante Bedrohungen für das Software-System untersucht wird.

Die mit einer Bedrohung in Verbindung stehenden wiederverwendbaren Vorlagen für Sicherheitsanwendungsfälle werden zur Unterstützung bei der Spezifikation von Sicherheitsanforderungen verwendet. Hierzu werden die zugehörigen Pakete ausgewertet und generische Sicherheitsanwendungsfälle auf das bestehende Software-System adaptiert. Dabei wird auch die Untersuchung von verwandten Bedrohungen und Sicherheitsanforderungen berücksichtigt.

Bewertung des Ansatzes

Der Fokus des Ansatzes von Sindre et al. liegt in der Wiederverwendung von Sicherheitswissen über Bedrohungen und Sicherheitsanforderungen und liefert hierzu fundierte Ausgangspunkte. Hierzu wird zunächst eine Trennung zwischen der Entwicklung sowie der Verwaltung der sicherheitsbezogenen Entwicklungsartefakte von deren Einsatz in einem konkreten Software-Entwicklungsprozess vorgenommen. Im Gegensatz zum Ansatz von Mellado et al. ist diese Trennung unabhängig von einem Vorgehen für Software-Produktlinien.

Gemäß des Anforderungskataloges kann festgestellt werden, dass eine Definition der eingesetzten Sicherheitskonzepte durch das vorgestellte Metamodell für die Ablagestruktur des Kataloges durchgeführt wird. Jedoch ist diese Definition des Katalogs aufgrund des Fokusses des Ansatzes stark

eingeschränkt und eine Erweiterung nicht vorgesehen. Eine Kategorisierung zwischen verschiedenen Sicherheitsbereichen wird nicht durchgeführt, sondern lediglich eine Unterscheidung zwischen generischen und anwendungsspezifischen Artefakten. Ebenso werden Alternativen bei der Erhebung und Spezifikation von Sicherheitsanforderungen nicht explizit berücksichtigt. Dagegen werden zu Bedrohungen zugehörige Sicherheitsanforderungen in Pakete zusammengefasst. Hierdurch wird nur eine geringfügige Unterstützung bei der Auswahl von verschiedenen Bedrohungen bzw. Sicherheitsanforderungen angeboten. Ebenso wird keine Angabe darüber gemacht, ob die Missbrauchs- und Sicherheitsanwendungsfälle aus bestehenden Sicherheitsinfrastrukturen abgeleitet werden.

Der Aufbau der wiederverwendbaren Missbrauchs- und Sicherheitsanwendungsfälle wird zunächst nur textbasiert dokumentiert und orientiert sich an dem Aufbau von Anwendungsfällen. Da diese ein weit verbreitetes Werkzeug in der Anforderungserhebung darstellen [So07], wird eine Integration mit den vorgestellten Entwicklungsartefakten ermöglicht. Eine Modellierung der wiederverwendbaren Artefakte ist zunächst auf Missbrauchs- und Sicherheitsanwendungsfälle beschränkt, wobei jedoch auch weitere Modellierungsmöglichkeiten nicht ausgeschlossen werden. Eine konkrete Modellierung wird in dem Ansatz jedoch nicht aufgezeigt. Der Einsatz der Entwicklungsartefakte in einem Software-Entwicklungsprozess wird unabhängig von konkreten Prozessmodellen definiert. Eine Definition von notwendigen Schnittstellen zur Integration in ein Prozessmodell werden jedoch nur implizit angedeutet.

Die Autoren ordnen ihren Ansatz explizit der Anforderungsanalysephase zu und versuchen die verfrühte Festlegung von sicherheitsbezogenen Entwurfsentscheidungen durch entsprechende Formulierung der Sicherheitsanforderungen zu vermeiden. In den aufgezeigten Beispielen wird dies jedoch nicht immer deutlich, vor allem da keine explizite Unterscheidung zwischen Sicherheitsanforderung und Sicherheitsmaßnahme durchgeführt wird. Die Entscheidung, welche zur Verbindung von Bedrohungen und zugehörigen Sicherheitsanforderungen führt, wird in dem Ansatz nicht explizit dokumentiert. Grund hierfür ist die Annahme, dass diese Zuordnung durch Sicherheitsexperten vorgenommen werden soll. Ebenfalls wird die Übertragung der Sicherheitsanforderungen auf Sicherheitsmaßnahmen durch den Ansatz nicht berücksichtigt.

3.2.3 Firesmith: Vorlagen für wiederverwendbare Sicherheitsanforderungen

Ausgehend von der Hypothese, dass Sicherheitsanforderungen im Gegensatz zu funktionalen Anforderungen hochgradig wiederverwendbar sind, stellt Donald Firesmith in [Fi04] Vorlagen für wiederverwendbare Sicherheitsanforderungen vor. Hierbei wird besonderen Wert darauf gelegt, die Sicherheitsanforderungen unabhängig von konkreten Sicherheitsmechanismen zu spezifizieren, welche die Anforderungen umsetzen. Als Ergänzung werden die Vorlagen in einem Analyseprozess eingebettet, in welchem anhand von Ressourcen und den Bedrohungsrisiken passende Sicherheitsanforderungen unter Einsatz der Vorlagen ermittelt werden. Das Ziel der wiederverwenbaren Vorlagen ist die Unterstützung von unerfahrenen Entwicklern bei der Identifikation, Analyse und Spezifikation von Sicherheitsanforderungen.

Ergebnisse des Ansatzes

Zur Unterstützung des Verständnisses über Sicherheitsanforderungen werden in der Arbeit sicherheitsbezogene Konzepte definiert, welche für die Spezifikation von Sicherheitsanforderungen relevant sind. Ziel der Darstellung der Relationen zwischen den Konzepten ist es zum einen, die Abhängigkeiten zwischen Konzepten zu klären und somit die Einflüsse bei der Spezifikation von Sicherheitsanforderungen herauszustellen. Zum anderen wird eine Definition des Sicherheitsbegriffes im Kontext der Anforderungserhebung vorgenommen. Darüber hinaus wird mittels der vorgestellten

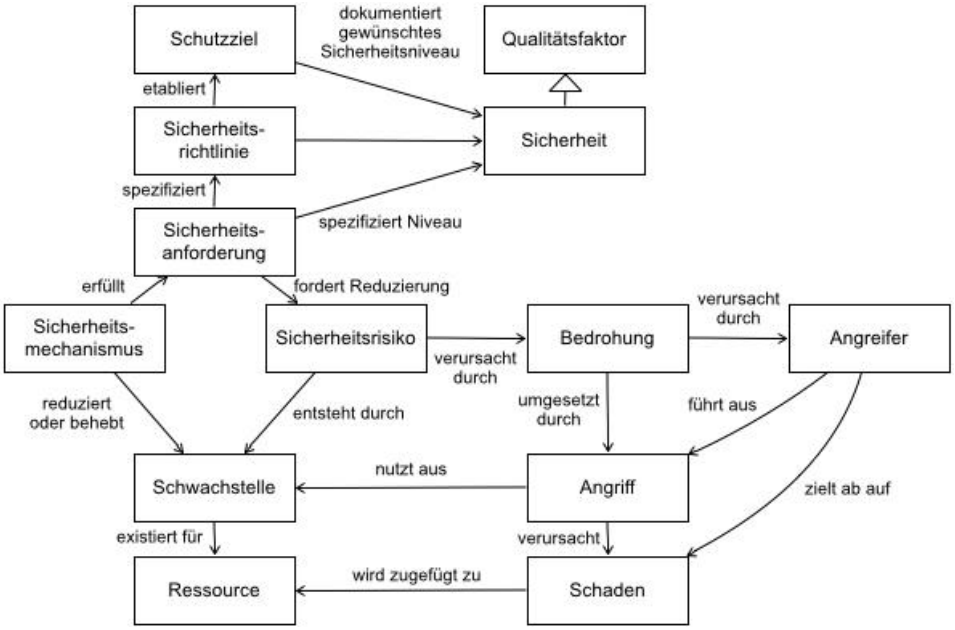

Abbildung 14: Definition von Konzepten für Sicherheitsanforderungen [Fi04]

Konzepte eine Einordnung von Sicherheitsanforderungen in einen Entwicklungsprozess vorgenommen.

Im Zentrum der Untersuchung von Firesmith stehen die Eigenschaften und der Aufbau von Sicherheitsanforderungen. Grundaussage der Arbeit ist, dass Sicherheitsanforderungen auf der abstrakten Ebene einer Anforderungserhebung selten variieren. Ursache hierfür ist, dass Software-Systeme aus unterschiedlichen Fachdomänen ähnlichen Angriffen und Bedrohungen ausgesetzt sind und somit ähnliche Schutzziele erfordern. Den konkreten Eigenschaften der Angriffe wird dabei im Verlauf der Entwurfsphase durch unterschiedliche Sicherheitsmechanismen entgegengewirkt. Dagegen wird durch Sicherheitsanforderungen lediglich der Bedarf für Sicherheitsmechanismen definiert sowie sicherheitsbezogene Qualitätseigenschaften festgelegt. Dies ermöglicht eine Uniformität zwischen Sicherheitsanforderungen verschiedener Software-Systeme, welches wiederverwendbare Vorlagen motiviert.

Die Vorlagen werden dabei informell in textueller Form spezifiziert, wobei mittels Platzhalter eine Parametrisierung der Anforderung ermöglicht wird. Als Beispiel für eine wiederverwendbare Integritätsanforderung wird folgende Vorlage angegeben:

*Die **[Ressource]** soll die **[Identifikator | Typ]**-Daten bei einer Übermittlung vor Korrumpierung durch **[Angriffsart]**-Angriffe während des **[Menge an Interaktionen / Anwendungsfälle]**-Anwendungsfalls schützen.*

Die in eckigen Klammern angegebenen Begriffe stellen die untypisierten Parameter dar, mittels welcher die Anforderungsvorlage individuell angepasst werden kann. Über den Ressource-Parameter können schützenswerte Ressourcen unterschiedlichen Abstraktionsniveaus, wie zum Beispiel Komponente oder ganze Software-Systeme, referenziert werden. Die Festlegung des Abstraktionsniveaus ist abhängig vom Kontext der Anforderungsanalyse sowie der fachlichen Domäne des Software-Systems. Zudem können über den Identifikator/Typ-Parameter, spezifische Datensätze referenziert werden, welche durch die Ressource bereitgestellt werden. Typen von Bedrohungen und Angriffen, welche durch die Sicherheitsanforderung behandelt werden sollen, können durch den Angriffsart-Parameter spezifiziert werden. Zudem kann mittels des

Anwendungsfall-Parameters eine Referenz auf fachliche Anwendungsfälle angegeben werden, in deren Rahmen die Bedrohung auftritt und die Sicherheitsanforderung gelten soll.

Die Parameter der Vorlagen werden in einem von Firesmith vorgestellten Verfahren zur Identifikation von Sicherheitsanforderungen festgelegt. In diesem werden zunächst schützenswerte Ressourcen ermittelt sowie relevante Kategorien von Angreifern bestimmt. Anschließend werden Bedrohungen für die ermittelten Ressourcen spezifiziert und deren Auswirkungen ermittelt. Ausgehend hiervon werden die Sicherheitsrisiken für die Ressourcen abgeschätzt und eine Priorisierung der Ressourcen bzw. Bedrohungen vorgenommen, wodurch die notwendigen Schutzziele in Form von Qualitätsfaktoren festgelegt werden. Die ermittelten Werte und Artefakte lassen sich im Weiteren für die Auswahl von relevanten Vorlagen für Sicherheitsanforderungen einsetzen.

Bewertung des Ansatzes

Der wesentliche Mehrwert der Untersuchung durch Firesmith liegt in der Beschreibung von wesentlichen Eigenschaften von Sicherheitsanforderungen, welche die Grundlage für deren Identifikation und Spezifikation darstellen. Als Resultat dieser Untersuchung wird die Spezifikation von Sicherheitsanforderungen als wiederverwendbare Entwicklungsartefakte ermöglicht, in dem parametrisierte Vorlagen beschrieben werden, welche auf den Kontext eines konkreten Software-Systems angepasst werden können.

Die Vorlagen für Sicherheitsanforderungen sowie der dazugehörige Analyseprozess werden unabhängig von einem konkreten Software-Entwicklungsprozess definiert, wodurch eine explizite Kopplung vermieden wird. Zwar werden in der Beschreibung der Vorlagen und des Analyseprozesses Verbindungen zu fachlichen Entwicklungsartefakten hergestellt und somit Vorbedingungen für den Einsatz in einem fachlichen Entwicklungsprozess definiert, jedoch wird eine solche Integration am Beispiel eines konkreten Prozesses nicht aufgezeigt.

Die von Firesmith vorgestellten Vorlagen für Sicherheitsanforderungen beziehen sich ausschließlich auf die Anforderungsanalysephase, wobei auf eine strikte Trennung zur Entwurfsphase geachtet wird. Ebenso wird in dem dargestellten Analyseverfahren lediglich die Identifikation und Spezifikation von Sicherheitsanforderungen berücksichtigt. Dementsprechend wird die Abbildung auf Sicherheitsmechanismen nicht weiter betrachtet. Hierdurch wird die durchgängige Betrachtung von Sicherheitsmaßnahmen in einem Entwicklungsprozess eingeschränkt. Des Weiteren wird das Vorgehen zur Spezifikation von Sicherheitsanforderungsvorlagen, insbesondere die damit verbundene Entscheidungsfindung, nicht weiter ausgeführt. Hierdurch ist der Einsatz der Vorlagen bei der Analyse von Sicherheitsanforderungen nur bedingt nachvollziehbar.

Zentraler Beitrag der Arbeit von Firesmith besteht in den wiederverwendbaren Sicherheitsanforderungsvorlagen. Hierzu werden die zur Spezifikation der Vorlagen notwendigen Konzepte informell definiert und in Relation gesetzt. Jedoch fließen diese nicht weiter in die Strukturierung der Vorlagen mit ein. Des Weiteren werden verschiedene Sicherheitsbereiche in Form von Qualitätsfaktoren definiert, mittels welcher die Vorlagen kategorisiert werden können. Eine Berücksichtigung von alternativen Vorlagen wird nicht explizit vorgenommen, da nach Aussage des Autors die Abstraktionsebene der Sicherheitsanforderungen dies nicht erfordert. Eine Unterstützung bei der Auswahl der Vorlagen wird durch die Bestimmung weiterer sicherheitsrelevanter Werte, wie zum Beispiel Bedrohungsrisiko und Schutzziele, in der Analyse ermöglicht, wodurch jedoch die Schwierigkeit auf die Ermittlung dieser Werte verschoben wird. Durch die Konzentration auf die Analysephase wird existierendes Sicherheitswissen in Form von Sicherheitsmodellen bzw. -technologien nicht weiter berücksichtigt.

3.2.4 Fernandez et al.: Sicherheitsmuster und Musterdiagramme für Sicherheitsmodelle

Die Arbeiten von Fernandez et al. umfassen eine vielfältige Untersuchung des Einsatzes von Sicherheitsmustern in der Software-Entwicklung. Basierend auf der Formulierung von Sicherheitsmustern unter Berücksichtigung von bekannten Sicherheitsmodellen, zum Beispiel in [SF+05], ist vor allem die Assoziation von verschiedenen Mustern in Musterdiagrammen relevant für die vorliegende Arbeit. Die Konsolidierung der zahlreichen Variationen von grundlegenden Zugriffskontrollmodellen steht im Zentrum des Ansatzes von Fernandez et al. [FP01] [PF+04] [DF+07] [FP+08] [FW+08]. Ausgangspunkt für den Ansatz ist die Feststellung, dass Software-Entwickler zumeist nur einfache Zugriffskontrollmodelle für Software-Systeme einsetzen, da kein Überblick über den spezifischen Zweck und die Verwendung von verfeinerten Variationen existiert. In der Arbeit werden daher zum einen Muster eingesetzt, um die Vor- und Nachteile verschiedener Zugriffskontrollmodelle vorzustellen. Zum anderen werden sogenannte Musterdiagramme verwendet, um die Beziehungen zwischen verschiedenen Mustern aufzuzeigen.

Ergebnisse des Ansatzes

Der Musteransatz zur Beschreibung von Zugriffskontrollmodellen wird von Fernandez et al. gewählt, da zum einen die vorhandenen Beschreibungselemente von Mustern, wie zum Beispiel Problemstellung und Lösung sowie UML-basierte Diagramme, Entscheidungskräfte und Konsequenzen, für die Autoren eine geeignete Grundlage bilden, um die Vor- und Nachteile der Zugriffskontrollmodelle sowie die Einsatzszenarien für spezifische Variationen zu beschreiben. Zum anderen beschreiben die in der Arbeit ausgewählten Zugriffskontrollmodelle bewährte Lösungen zu wiederkehrenden Sicherheitsproblemen, welches eine typische Charakteristik von Musterbeschreibungen darstellt.

Die Musterdiagramme liefern einen Zusammenhang zwischen den verschiedenen als Sicherheitsmuster beschriebenen Zugriffskontrollmodellen. Eine formale Beschreibung der Diagramme findet dabei nicht statt. Abbildung 15 gibt das in der Arbeit vorgestellte Beispiel für ein Musterdiagramm für Zugriffskontrollmodelle wieder. Sicherheitsmuster werden darin informell als abgerundete Rechtecke dargestellt, während gerichtete Pfeile die Beziehungen zwischen den Mustern ausdrücken. Die Art der Beziehung wird dabei durch Annotationen an den Beziehungspfeilen ausgedrückt.

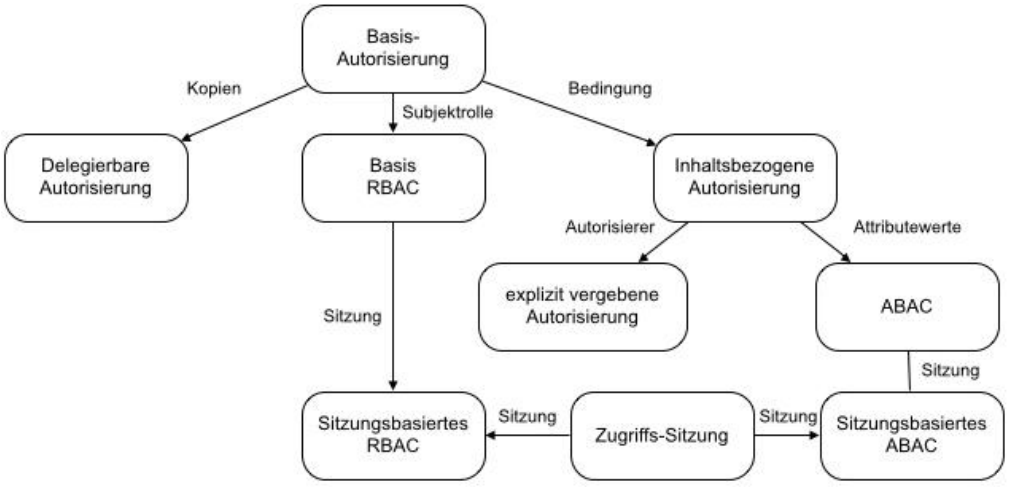

Abbildung 15: Musterdiagramm für Zugriffskontrollmodelle [FP+08]

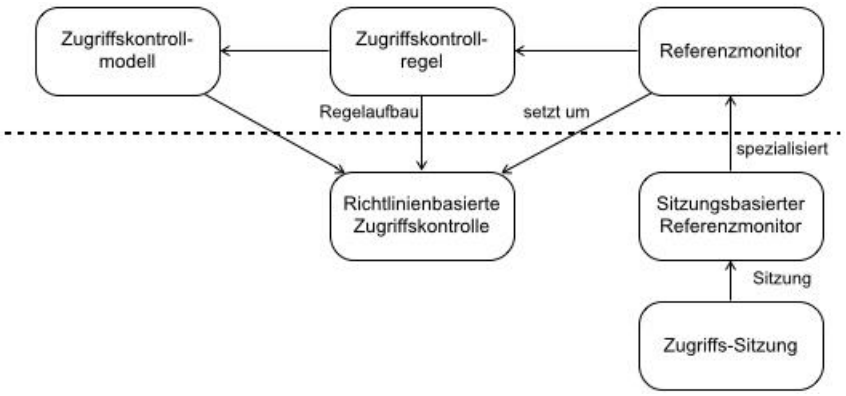

Abbildung 16: Abstraktionsebenen in Musterdiagrammen [FP+08]

Das abgebildete Musterdiagramm stellt die Zugriffskontrollmodelle bzw. Autorisierungsmodelle im Kontext ihrer zunehmenden Spezialisierung dar. Ein weiteres in Abbildung 16 dargestelltes Musterdiagramm zeigt eine abstrakte Übersicht über die wesentlichen Elemente einer Zugriffskontrolllösung und wird zur Auswahl eines Zugriffskontrollmodells eingesetzt. Hierdurch wird eine mehrstufige Abstraktion zwischen den Zugriffskontrollmustern hergestellt, welche optional im Musterdiagramm dargestellt werden kann. Somit wird nach Aussage der Autoren zunächst auf einem abstrakten Niveau eine grundlegende Entscheidung über das anzuwendende Zugriffskontrollmodell im Kontext des fachlichen Software-Systems getroffen und anschließend eine Spezialisierung des Modells durchgeführt.

Bewertung des Ansatzes

Die Beiträge der Arbeiten von Fernandez et al. sind richtungsweisend für den Einsatz von Sicherheitsmustern in der Software-Entwicklung. Die untersuchten Arbeiten verfolgen Ziele, welche mit den Absichten des Beitrags B3 dieser Arbeit vergleichbar sind. Hierbei sollen Sicherheitsmaßnahmen aus unterschiedlichen Abstraktionsebenen in Verbindung gesetzt werden, um eine Unterstützung bei der Auswahl von Maßnahmen in der iterativen Verfeinerung des Entwurfsprozesses zu ermöglichen. Hierzu werden grundlegende Überlegungen und Ergebnisse in den untersuchten Arbeiten vorgestellt.

Die Beschreibung der Zugriffskontrollmodelle unter Einsatz des Musteransatzes sowie die Darstellung der Zusammenhänge zwischen den Mustern wird unabhängig von einem spezifischen Entwicklungsprozess betrachtet. Durch den Musteransatz ist eine Integration in eine fachliche Entwicklung denkbar, da Muster bekannte Werkzeuge beim Entwurf von Software-Systemen darstellen. Dies wird in dem Ansatz jedoch nicht explizit betrachtet. Die Musterbeschreibungen sowie die Musterdiagramme für Zugriffskontrollmodelle stellen somit ein Werkzeug dar, welches in der Entwurfsphase für die Umsetzung von Zugriffskontrollmaßnahmen eingesetzt werden kann. Hierzu werden anhand der gegebenen Sicherheitsprobleme eines Software-Systems die Diagramme navigiert und entsprechende Muster ausgewählt. Eine Adaption dieses Vorgehens auf beliebige Entwicklungsverfahren ist durchaus denkbar, jedoch wird hierzu kein Nachweis geliefert.

Der Kern der Arbeiten von Fernandez et al. bezieht sich auf die Nachvollziehbarkeit der Entwurfsentscheidungen für Zugriffskontrollmaßnahmen. Hierzu liefern die Musterdiagramme einen wesentlichen Beitrag, um die Zusammenhänge zwischen existierenden Zugriffskontrollmaßnahmen aufzuzeigen. Insbesondere wird durch die Berücksichtigung von verschiedenen Abstraktionsebenen der Zusammenhang zwischen abstrakten und implementierungsnahen Zugriffskontrollmodellen stufenweise hergestellt. Durch die fehlende Formalisierung der Musterdiagramme findet jedoch keine explizite

Beschreibung von Varianten statt. Hierdurch wird keine Unterstützung bei der Auswahl von passenden Zugriffskontrollmustern geliefert, sondern lediglich eine Übersicht gegeben.

Die Verwendung des Musteransatzes wurde laut den Autoren gewählt, da die untersuchten Zugriffskontrollmodelle bewährte Praktiken zur Lösung von entsprechenden Sicherheitsproblemen darstellen. Somit ist die Wiederverwendung dieser Modelle ein wesentliches Ziel der Arbeit. Bei der Formulierung der Sicherheitsmuster ist festzustellen, dass in der Arbeit von Fernandez et al. Zugriffskontrolle auch die Autorisierung beinhaltet, wodurch eine Mischung von Mustern aus konzeptionell unterschiedlichen Sicherheitsbereichen stattfindet. Des Weiteren wird zwar eine Unterscheidung zwischen abstrakten und technischen Zugriffskontrollmaßnahmen, wie zum Beispiel für Dateisystemzugriff, durchgeführt, jedoch werden keine bestehenden Sicherheitsprodukte hinsichtlich der Umsetzung der Muster untersucht. Lediglich in [DF+07] findet sich die Berücksichtigung des SAML-Standards zur Formulierung von Musterdiagrammen für föderiertes Identitäts- und Zugriffsmanagement. Dennoch wird dies von den Autoren als möglicher Anwendungszweck der Musterdiagramme angesehen.

3.2.5 Fægri und Hallenstein: Sicherheitsreferenzarchitektur für Software-Produktlinien

In ihrer Arbeit stellen Fægri und Hallenstein eine Referenzarchitekturfür Sicherheit in Software-Produktlinien (SPL) vor [FH06]. Die Referenzarchitektur hat zum Ziel, Software-Architekten bei der Integration von Sicherheitsanforderungen und zugehöriger Maßnahmen in die Architektur einer SPL zu unterstützen. Hierzu werden verschiedene Variationspunkte und Varianten zur Umsetzung von Sicherheit aufgezeigt sowie eine Entscheidungsunterstützung bei der Auswahl geeigneter Varianten angeboten.

Ergebnisse des Ansatzes

Die Arbeit umfasst ein konzeptionelles Modell für die Referenzarchitektur, welches in drei Untermodelle aufgeteilt wird (vgl. Abbildung 17). Mittels eines Sicherheits-Submodells lässt sich eine einfache Bedrohungsanalyse und Risikobeurteilung bei der Entwicklung von Software-Systemen durchführen. Diese soll nach Aussage der Autoren durch externe Methoden zur Risikobeurteilung unterstützt werden. Die wesentlichen Konzepte dieses Submodells bestehen aus Bedrohungen, welche Ausgangspunkt für die Betrachtung von Sicherheitsmaßnahmen darstellen. Diese wiederum dienen der Vermeidung oder Verringerung von Schwachstellen von Software-Entitäten bzw. Ressourcen. Unerwünschte Ereignisse sind konkrete Ausprägungen von Bedrohung. Sie treten mit einer gewissen Wahrscheinlichkeit ein und haben bestimmte Auswirkungen auf die Software-Entität. Das Risiko des Auftretens eines ungewollten Ereignisses wird als Produkt der Wahrscheinlichkeit und der Auswirkung des Ereignisses modelliert.

In einem Architektur-Submodell werden die Maßnahmen zur Umsetzung von Sicherheitsanforderungen zusammengefasst. Hierbei wird eine Kategorisierung in Maßnahmen zur Vermeidung und Entdeckung von ungewollten Ereignissen sowie in Maßnahmen zur Wiederherstellung nach dem Eintritt von ungewollten Ereignissen vorgenommen. Mit jeder dieser sogenannten Architekturtaktiken sind konzeptionelle Vorgehensweisen zur Umsetzung von Sicherheitsanforderungen verbunden, welche schrittweise verfeinert und zu bestimmten Sicherheitsmustern konkretisiert werden. Jede Taktik ist dabei als Lösung im Sinne einer Gegenmaßnahme gegen eine Bedrohung anzusehen und stellt somit verschiedene Varianten einer Sicherheitsmaßnahme dar.

Die verschiedenen Sicherheitsmaßnahmen der Taktiken werden mittels Sicherheitsmuster ausgedrückt, wodurch sie eine Architektursprache bilden, mittels welcher die Muster kategorisiert werden. Ebenso werden Spezialisierungsbeziehungen zwischen den Mustern hergestellt, durch welche eine

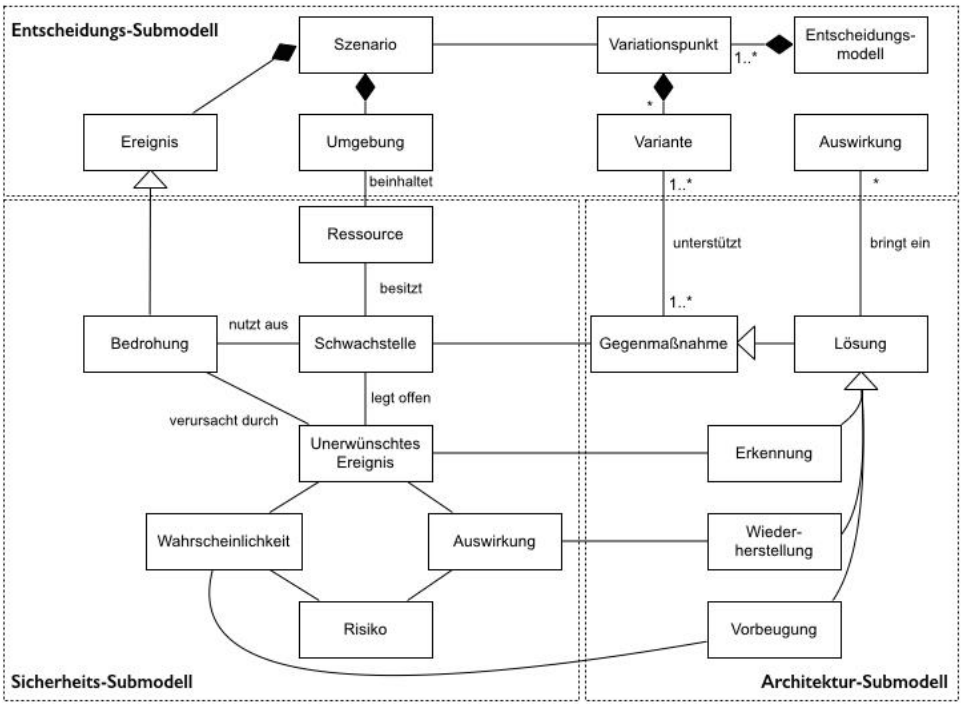

Abbildung 17: Konzeptionelles Modell einer Sicherheits-Referenzarchitektur [FH06]

Ableitung von verfeinerten Sicherheitsmaßnahmen ermöglicht wird. In Abbildung 18 wird ein Ausschnitt der Architektursprache für die Bereiche Zugriffskontrolle und Authentifizierung dargestellt.

Die Elemente Variation und Variationspunkt sind Teil des Submodells zur Entscheidungsunterstützung, in welchem die Variationspunkte als generische Szenarien erarbeitet werden. Diese werden dabei in Kooperation mit Industriepartnern erarbeitet sowie aus der Fachliteratur abgeleitet. Die Szenarien repräsentieren die wesentlichen sicherheitsbezogenen Qualitätsattribute Integrität, Vertraulichkeit, Verfügbarkeit und Verantwortlichkeit eines Qualitätsmodells. Durch sie lassen sich geeignete Variationspunkte und Variationen als Antwort auf die durch das Szenario gegebenen Bedrohungen auswählen. Die Bedrohungen werden dabei durch ihren Kontext und ihren Auslöser beschrieben.

Bewertung des Ansatzes

Mit ihrem Ansatz verfolgen die Autoren das Ziel, den über die Jahre gewachsenen Wissenstands über die verschiedenen Möglichkeiten zur Umsetzung von Sicherheitsanforderungen in ein iterativ verfeinertes Entscheidungsmodell zu komprimieren. Damit liefern Fægri und Hallenstein einen wichtigen Beitrag in der Unterstützung von Software-Architekten und -Entwicklern in der Einbeziehung von Sicherheitsanforderungen und den darauf bezogenen Entwurf von sicheren Software-Architekturen.

Das Qualitätsmodell bildet zusammen mit den beschriebenen Sicherheitstaktiken und -mustern eine Sprache, mittels welcher ein einheitliches Vokabular zwischen den Beteiligten eines Software-Projektes aufgebaut wird. Jedoch beinhaltet es lediglich die klassischen und sicherlich wichtigen Schutzziele Vertraulichkeit, Integrität, Verfügbarkeit und Verantwortlichkeit. Diese bilden nicht mehr die Gegebenheiten der sich an ständig neuen Bedrohungen anpassenden Informationstechnologie ab und müssen durch weitere Attribute ergänzt werden [WM05:9].

Den Einstieg in das Entscheidungsmodell bilden die sehr generisch aufbereiteten Szenarien, welche durch Literaturstudium und Industrieprojekte entlang der Qualitätsattribute erarbeitet wurden. Trotz der Allgemeingültigkeit ist anzuzweifeln, dass Szenarien so beschrieben werden können, dass auch

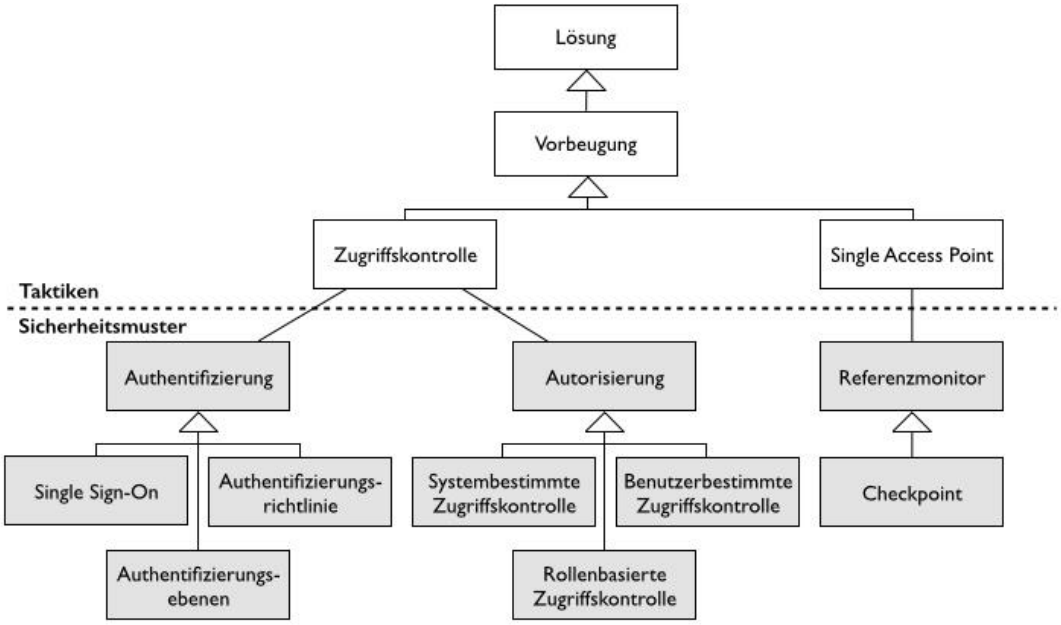

Abbildung 18: Ausschnitt aus der Sicherheitsarchitektursprache [FH06]

ungeübte Entwickler aus der Beschreibung der Szenarien die Relevanz für das eigene Projekt erkennen können.

Die durch das Entscheidungsmodell vorgeschlagenen Sicherheitsmuster haben oft Abhängigkeiten untereinander und es gibt zu einem Sicherheitsmuster mehrere alternative Umsetzungsmöglichkeiten. Diese Abhängigkeiten sind durch die mehrheitlich textuelle Beschreibung nicht deutlich hervorgehoben, wodurch Entwickler in ihrer Entscheidungsfindung nicht unbedingt unterstützt werden. Es besteht sogar die Gefahr der Verwirrung und des Einsatzes eines ungeeigneten Musters.

Die Referenzarchitektur ist als Ergänzung zu einer Software-Produktlinie gedacht und soll dabei die Ermittlung von Sicherheitsmaßnahmen unterstützen. Da SPL hauptsächlich die Entwicklung von ähnlichen Software-Produkten zum Ziel hat, lässt sich dieser Ansatz nicht direkt auf die Gegebenheiten eines dienstorientierten Ansatzes anwenden. Ebenso ist zu bemerken, dass sich der Ansatz lediglich auf den Entwurf der SPL bezieht. Die Anforderungsanalyse und die Implementierungsphase werden ausgeblendet. Wie die Referenzarchitektur in einer existierenden SPL oder einer Anwendungsentwicklung eingesetzt wird, wird ebenfalls nicht gezeigt.

3.3 Zusammenfassung und Handlungsbedarf

Die Bewertung der einzelnen Arbeiten im vorangegangenen Kapitel 3.2 ist in der nachfolgenden Tabelle 4 zusammengefasst. Ein ausgefülltes Kreissymbol („●“) symbolisiert eine Erfüllung, während ein nichtausgefülltes Kreissymbol („○“) entsprechend die Nichterfüllung einer Anforderung repräsentiert. Ist eine Anforderung nur zum Teil erfüllt, wird dies durch ein zur Hälfte ausgefülltes Kreissymbol („◑“) dargestellt.

Diskussion der untersuchten Ansätze

Die Integration von Artefakten und Aktivitäten einer sicherheitsbasierten Software-Entwicklung wird durch die untersuchten Ansätze im Allgemeinen nicht zufriedenstellend behandelt. Sicherheitsbezogene Entwicklungsartefakte wie die wiederverwendbaren Missbrauchsanwendungsfälle von Sindre und

Opdahl sowie die Vorlagen für Sicherheitsanforderungen von Firesmith werden zwar unabhängig von konkreten Prozessmodellen definiert, jedoch wird die Anpassung der Ansätze auf spezifische Entwicklungsprozessmodelle vernachlässigt. Ähnlich verhält es sich mit Aktivitäten für die Entwicklung von Sicherheitsartefakten. Hierbei werden die Abhängigkeiten zu einem fachlichen Entwicklungsprozess meist nur implizit angedeutet oder ganz vernachlässigt, wodurch die Anpassung und Einbettung der Aktivitäten an konkrete Entwicklungsprozesse nicht überprüft werden kann.

Tabelle 4: Bewertung der untersuchten Ansätze

Bewertungskriterien	**Mellado et al.: Sicherheitsanforderungen in Software-Produktlinien**	**Sindre et al.: Wiederverwendbare Missbrauchsfälle.**	**Firesmith: Wiederverwendbare Sicherheitsanforderungen**	**Fernandez et al.: Sicherheitsmuster und Musterdiagramme**	**Fægri und Hallenstein: Sicherheitsreferenzarchitektur für Software-Produktlinien**
AK1: Integrationsfähigkeit					
Unabhängigkeit von Entwicklungsprozessen	○	●	●	◑	○
Anpassungsfähigkeit	○	●	●	◑	○
AK 2: Nachvollziehbarkeit					
Trennung von Entwicklungsphasen	●	●	◑	◑	◑
Abbildung zwischen Phasen	○	○	◑	○	●
Verfeinerung von Artefakten	○	○	◑	●	●
AK 3: Wiederverwendbarkeit					
Definition von Sicherheitskonzepten	●	◑	◑	○	●
Kategorisierung von Maßnahmen	○	◑	○	◑	●
Unterstützung bei Entscheidungen	◑	○	○	◑	◑
Berücksichtigung von Infrastrukturen	○	○	○	◑	○
AK4: Praktikabilität					
Standardisierung	○	◑	○	◑	○
Werkzeugunterstützung	○	◑	○	◑	○
Automatisierung	◑	○	○	○	◑

Das Ergebnis der Bewertung zeigt außerdem, dass die Ansätze die Nachvollziehbarkeit der sicherheitsrelevanten Entwicklungsartefakte vernachlässigen. Hierbei wird die Zugehörigkeit der Artefakte zu Entwicklungsphasen zwar zumeist eindeutig geklärt, jedoch wird bei keinem der Ansätze eine durchgängige Unterstützung von allen Entwicklungsphasen berücksichtigt, sondern lediglich einzelne Phasen betrachtet. Hierdurch kann die geforderte durchgängige Abbildung zwischen Artefakten verschiedener Entwicklungsphasen durch die Ansätze nicht gewährleistet werden. Ebenso wird für die Ableitung von Entwicklungsartefakten innerhalb einer Entwicklungsphase nur unzureichende Hilfestellung gegeben. Im Falle der Sicherheitsmuster und Musterdiagramme von Fernandez et al. wird beispielsweise die iterative Verfeinerung der Artefakte nur informell beschrieben, wodurch keine klare Unterstützung bei deren Einsatz und Auswahl bereitgestellt wird. Dies lässt sich auch für die wiederverwendbaren Sicherheitsanforderungen und Bedrohungsbeschreibungen von Sindre et al. bzw. Firesmith feststellen.

Alle betrachteten Ansätze behandeln die Wiederverwendung von sicherheitsbezogenen Entwicklungsartefakten und Aktivitäten. Hierbei wird teilweise existierendes Sicherheitswissen aufgegriffen und für den Einsatz in einem Software-Entwicklungsprozess aufbereitet. Dies trifft zum Beispiel für die Sicherheitsmuster von Fernandez et al. und der Sicherheitsarchitektursprache von Fægri und Hallenstein zu. Jedoch werden dabei zumeist einzelne Standards und Modelle betrachtet, ohne einen Bezug zu einer bestehenden Sicherheitsinfrastruktur zu berücksichtigen, welche die Standards unterstützen muss. Des Weiteren werden teilweise Sicherheitskonzepte mittels Domänenmodellen spezifiziert, welche zumeist jedoch zu restriktiv sind und sich nicht anpassen bzw. erweitern lassen. Auffällig sind dabei auch die teilweise unterschiedlichen Auffassungen über verschiedene Sicherheitsbegriffe. Die Berücksichtigung von Varianten wird vor allem bei den Ansätzen für Software-Produktlinien von Mellado et al. sowie Fægri und Hallenstein aufgegriffen. Auch die Musterdiagramme von Fernandez et al. behandeln die unterschiedlichen Ausprägungen von Sicherheitsmodellen. Jedoch wird bei diesen Ansätzen keine Unterstützung bei der Auswahl der unterschiedlichen Varianten bereitgestellt, sondern lediglich eine Übersicht gegeben.

Der praktische Einsatz der sicherheitsbezogenen Artefakte und Aktivitäten wird von den betrachteten Ansätzen unterschiedlich behandelt. Der Einsatz von UML-basierten Diagrammen und Modellen ist beispielsweise in den Ansätzen von Sindre et al. sowie Fernandez vorgesehen, um Bedrohungen und Sicherheitsanforderungen in Anwendungsfalldiagrammen bzw. Sicherheitsmuster in UML-Klassen- bzw. -Komponentendiagramme darzustellen. Im letzteren Ansatz werden jedoch die Musterdiagramme nur informell beschrieben. Ebenso werden in den Ansätzen für Software-Produktlinien keine bekannten Modellierungssprachen verwendet. Dies wirkt sich negativ auf den Einsatz der Ansätze aus, da die Verwendung von Entwicklungswerkzeugen nur eingeschränkt möglich ist. Ebenso ist die automatische Generierung von Entwicklungsartefakten im Allgemeinen in der Software-Produktlinienentwicklung vorgesehen, um eine effiziente Entwicklung zu ermöglichen. Jedoch wird dies in den betrachteten Ansätzen nicht weiter berücksichtigt.

Zusammenhang zu Beiträgen der vorliegenden Arbeit

Trotz der aufgezeigten Defizite der untersuchten Arbeiten bieten die darin behandelten Ansätze eine gute Ausgangsbasis zur Umsetzung der Ziele der vorliegenden Arbeit, welche in der strukturierten Aufbereitung und Integration von bestehendem Sicherheitswissen in Software-Entwicklungsprozesse liegen. Aus diesem Grund werden die Vorteile der untersuchten Arbeiten als Grundlage genutzt und so erweitert, dass die Beiträge der Arbeit die Ziele umsetzen.

Zur Aufbereitung von bestehendem Sicherheitswissen zu wiederverwendbaren sicherheitsbezogenen Entwicklungsartefakten wurde aus den untersuchten Ansätzen deutlich, dass diese Aktivität unabhängig von dem Einsatz der Artefakte in einem Software-Entwicklungsprozess durchgeführt

werden sollte. Im Fall der Ansätze bezüglich Software-Produktlinien ist dies ein elementarer Bestandteil des Paradigmas, um gemeinsame Sicherheitsanforderungen und -maßnahmen in verschiedenen Software-Systemen einzusetzen. Aus den Beobachtungen von Fernandez et al. sowie Sindre et al. geht weiterhin hervor, das sicherheitsbezogene Entwicklungsartefakte einen hohen Wiederverwendungsgrad aufweisen, da unterschiedliche Software-Systeme ähnlichen Bedrohungen ausgesetzt sind. Wird der fachliche Kontext der Software-Systeme vernachlässigt, so unterscheiden sich deren Sicherheitsanforderungen und -maßnahmen daher nur minimal [SF+03] [FP+08].

Aus diesem Grund ist für das im Beitrag B1 behandelte sicherheitsbasierte Entwicklungsvorgehen eine solche Trennung zwischen Aufbereitung und Einsatz von sicherheitsbezogenen Entwicklungsartefakten vorgesehen. Das Ziel ist jedoch nicht, eine spezifische Software-Produktlinie zu unterstützen, sondern ein generisches Entwicklungsvorgehen zu beschreiben, welches an beliebige Entwicklungsprozessmodelle bzw. -paradigmen angepasst werden kann. Die Aufteilung des Entwicklungsvorgehens wird somit zur Umsetzung der Anforderungskategorie AK1 sowie AK 3 durchgeführt. Ebenso soll im Gegensatz zu den betrachteten Ansätzen eine durchgängige Unterstützung aller relevanten Entwicklungsphasen bereitgestellt werden, indem eine Verknüpfung zwischen den Entwicklungsaktivitäten und -artefakten aufeinanderfolgender Phasen hergestellt wird.

Zur Beschreibung der aufbereiteten Entwicklungsartefakte werden zum einen die Grundzüge der Ansätze von Vorlagen für Sicherheitsanforderungen von Sindre et al. und Firesmith sowie die Sicherheitsmuster und Musterdiagramme von Fernandez et al. übernommen. Durch die Beschreibung von bekannten Angriffen und Definition von Sicherheitsanforderungen, welche den Angriffen entgegenwirken, wird die Bedrohungsanalyse von Software-Systemen erleichtert. Ebenso gelten Sicherheitsmuster und deren kombinierter Einsatz als akzeptierte Werkzeuge in der Software-Entwicklung. Durch die Nähe zu Architektur- und Entwurfsmustern lassen sie sich in den Entwurfsprozess integrieren.

Als Erweiterung dieses Ansatzes wird in dieser Arbeit eine Formalisierung der Entwicklungsartefakte in den Beiträgen B2 und B3 angestrebt. Zum einen soll die Struktur von Sicherheitsanforderungen durch ein flexibles und adaptives Domänenmodell beschrieben werden, so dass die Anpassung an verschiedene Entwicklungsprozessmodelle ermöglicht wird. Zum anderen wird durch diese Strukturierung die Trennung zwischen Syntax und Semantik der Sicherheitsanforderungsvorlagen erreicht, so dass unterschiedliche Spezifikationsformen, wie um Beispiel Missbrauchs- und Sicherheitsanwendungsfälle unterstützt werden können.

Des Weiteren sollen zur Verbesserung der Nachvollziehbarkeit der Entwurfsentscheidungen für Sicherheitsmaßnahmen (Anforderungskategorie AK 2) die Beziehungen zwischen Sicherheitsmustern durch eine formalere Struktur unterstützt werden. Dabei wird das Ziel verfolgt, eine Hilfestellung bei der Auswahl zwischen verschiedenen alternativen Sicherheitsmaßnahmen zu ermöglichen. Ausgehend von den Musterdiagrammen als auch der Sicherheitsarchitektursprache von Fægri und Hallenstein werden hierzu Funktionsmodelle (engl. feature models) eingesetzt, um die Assoziationen zwischen alternativen Sicherheitsmustern zu beschreiben sowie die Verbindung zwischen Mustern unterschiedlicher Abstraktionsstufen herzustellen. Durch den Einsatz von standardisierten Modellierungssprachen, wie zum Beispiel der UML soll dabei die Praktikabilität der Entwicklungsartefakte gewährleistet werden (Anforderungskategorie AK4).

Zusätzlich werden existierende Sicherheitsinfrastrukturen bei der Aufbereitung von Entwicklungsartefakten berücksichtigt. Diese Infrastrukturen stellen Sicherheitsfunktionalität durch eingesetzte Sicherheitskomponenten bzw. -produkten bereit, welche auf die Entwicklungsartefakte abgebildet werden müssen. Hierdurch wird zum einen die Aufbereitung von Sicherheitswissen durchgeführt, welches bereits aktiv eingesetzt wird, sowie zum anderen die Durchgängigkeit des Entwicklungsvorgehens von Analyse und Entwurf zur Implementierung ermöglicht.

4 Entwicklungsvorgehen für sichere Software

Die Entwicklung von sicheren Anwendungen bedarf der Berücksichtigung von Sicherheitseigenschaften in allen Phasen eines Software-Entwicklungsprozesses [An08]. Da der Fokus eines Software-Entwicklungsprozesses hauptsächlich auf der Analyse und Umsetzung der fachlichen Funktionalität liegt, muss dieser durch Aktivitäten zur Analyse und Umsetzung von Sicherheitsfunktionalität erweitert werden. Hierbei werden in einem strukturierten Vorgehen die notwendigen Sicherheitsanforderungen einer Anwendung identifiziert und spezifiziert sowie entsprechende Sicherheitsmaßnahmen entworfen und durch angemessene Technologien implementiert.

Wie im vorhergehenden Kapitel aufgezeigt, besteht ein Handlungsbedarf bei der Spezifikation und Durchführung eines solchen sicherheitsbasierten Entwicklungsvorgehens. Vor allem eine notwendige Wiederverwendung von bestehendem Sicherheitswissen wurde zumeist nur unzureichend betrachtet. Ebenso bedarf es einer Durchgängigkeit der sicherheitsrelevanten Entwicklungsartefakte. In diesem Kapitel werden die identifizierten Anforderungen aufgegriffen und ein Vorgehen zur Entwicklung von sicherer Software vorgestellt.

Hierzu wird zunächst in Kapitel 4.1 die Einordnung des Entwicklungsvorgehens in einen fachlichen Software-Entwicklungsprozess aufgezeigt. Darin werden die primären Aktivitäten und deren Zusammenhänge sowie die Ergebnisartefakte der Aktivitäten beschrieben. Die nachfolgenden Abschnitte beschreiben das Vorgehen im Detail. Das in diesem Kapitel beschriebene Vorgehen ist dabei in zwei Phasen unterteilt. Die in Kapitel 4.2 beschriebene Phase zur Aufbereitung von existierendem Sicherheitswissen behandelt die Entwicklung von wiederverwendbaren Entwicklungsartefakten. Im anschließenden Kapitel 4.3 wird die Phase zur Anwendungsentwicklung betrachtet, in welcher die Verwendung der wiederverwendbaren Entwicklungsartefakte in der Entwicklung eines konkreten Software-Systems fokussiert wird.

4.1 Einordnung in einen Software-Entwicklungsprozess

Das in dieser Arbeit vorgestellte Entwicklungsvorgehen für sichere Software fokussiert die Wiederverwendung von bestehendem Sicherheitswissen und die Durchgängigkeit der sicherheitsrelevanten Entwicklungsartefakte. Als Quellen von bestehendem Sicherheitswissen lassen sich sowohl existierende Sicherheitsprodukte und -rahmenwerke als auch dokumentierte bewährte Methoden identifizieren. Sicherheitsprodukte stellen dabei ausgereifte und stabile Sicherheitsfunktionalität bereit, welche zur Umsetzung von Sicherheitsmaßnahmen eingesetzt werden sollte. Konzeptionelle Sicherheitsmodelle dagegen abstrahieren von technologischen Details und stellen bewährte Methoden zur Lösung von Sicherheitsproblemen bereit. Idealerweise werden diese Modelle durch die Produkte umgesetzt.

In dieser Arbeit werden diese wiederverwendbaren sicherheitsrelevanten Entwicklungsartefakte explizit verknüpft und als Teil eines Referenzmodells für Sicherheitsarchitekturen betrachtet, welche die Artefakte für die Anwendung in einem Software-Entwicklungsprozess bereitstellt. Aufgrund der Komplexität der Sicherheitsdomäne lassen sich diese Quellen nicht ohne größeren Aufwand direkt durch fachliche Software-Entwickler einsetzen. Somit muss das bestehende Sicherheitswissen von Sicherheitsexperten zunächst so aufbereitet werden, dass eine effiziente Integration in einen Software-Entwicklungsprozess ermöglicht wird. Daher ist es sinnvoll, eine Trennung zwischen der Aufbereitung des Sicherheitswissens und dem Einsatz des Sicherheitswissens in einem Entwicklungs-

prozess durchzuführen. Hierdurch wird zudem eine Trennung von Zuständigkeiten zwischen Sicherheitsexperten und Software-Entwicklern ermöglicht.

Im Folgenden wird der Zusammenhang zwischen dem sicherheitsbasierten Entwicklungsvorgehen und einem fachlichen Software-Entwicklungsprozess präsentiert. Als zentrales Artefakt wird hierzu zunächst ein Referenzmodell für Sicherheitsarchitekturen vorgestellt, in welchem wiederverwendbare Entwicklungsartefakte spezifiziert werden. Durch verschiedene Sichten werden die Artefakte zweckorientiert aufbereitet und zur Verfügung gestellt. Das Referenzmodell stellt eine erweiterte Betrachtung einer Sicherheitsarchitektur dar, welche die Artefakte der Architektur hinsichtlich der sicherheitsbasierten Entwicklung darstellt und als Basis für die Anwendung des aufbereiteten Sicherheitswissens dient. Im Anschluss wird der grobe Ablauf des Entwicklungsvorgehens präsentiert und der Zusammenhang zu den Aktivitäten sowie den Artefakten eines fachlichen Entwicklungsprozesses aufgezeigt.

4.1.1 Referenzmodell für Sicherheitsarchitekturen

Im Gegensatz zu den betrachteten Ansätzen zur sicherheitsbasierten Software-Entwicklung wird in dem in dieser Arbeit vorgestellten Entwicklungsvorgehen der Fokus auf die Aufbereitung und Berücksichtigung von existierendem Sicherheitswissen gelegt. Dabei wird vor allem eine Sicherheitsarchitektur als Quelle für das in einem Software-Entwicklungsprozess eingesetzte Wissen angesehen. Die Entwicklung von sicheren Software-Systemen führt zum Entwurf von Sicherheitsmaßnahmen, welcher analog zu einer fachlichen Funktionalität in logischen Komponenten gekapselt und lose gekoppelt ist. Die Komponenten werden dabei durch existierende Sicherheitsprodukte implementiert, um fehleranfällige Eigenimplementierungen zu vermeiden.

Wird ein Software-System als eigenständige Lösung entwickelt, so kann sich die Auswahl von geeigneten Produkten an den Sicherheitsanforderungen des Software-Systems orientieren. Im Fall einer Integration des Software-Systems in eine bestehende Anwendungslandschaft, zum Beispiel in einem betrieblichen Umfeld, müssen auch die vorhandenen Sicherheitsprodukte berücksichtigt werden. Im Kontext der Aufbereitung und der Anwendung von existierendem Sicherheitswissen führt diese Beobachtung zu einer erweiterten Betrachtung von Sicherheitsarchitekturen. Diese nimmt neben der Spezifikation von Sicherheitsfunktionalität in Form von Diensten und der Integration der Dienste in eine Anwendungslandschaft [EB+07] [KB+07] insbesondere auch eine Abstraktion der Sicherheitsfunktionalität hinsichtlich des Einsatzes in einem Software-Entwicklungsprozess vor.

Dies führt zu verschiedenen Sichten auf eine betrachtete Sicherheitsarchitektur, wobei jede Sicht einen Ausschnitt der in der Architektur vorhandenen Informationen in angemessener Form für unterschiedliche Zwecke zur Verfügung stellt. Die Sichten der Sicherheitsarchitektur dienen somit als Referenz bei der Entwicklung von Sicherheitsfunktionalität, woraus sich das im Folgenden vorgestellte Referenzmodell ergibt. Die klassischen Architektursichten unterteilen eine Software-Architektur in ihre statische Struktur und betrachten die Dynamik sowie Verteilung der Komponenten [CB+10] [RH+06: 37ff]. Im Gegensatz dazu werden im Referenzmodell hierzu orthogonale Sichten berücksichtigt, welche die Aufbereitung des Sicherheitswissens unterstützen.

Wie in Abbildung 19 dargestellt, beinhaltet das Referenzmodell eine Integrationssicht, eine Dienstsicht und eine Entwicklungssicht. Dabei ist insbesondere die letzte Sicht für das sicherheitsbasierte Entwicklungsvorgehen relevant, da sie das hierfür notwendige Wissen der Sicherheitsarchitektur in angemessener Form bereitstellt. Diese Sicht basiert auf den Informationen, welche in der Integrationssicht und der Dienstsicht bereitgestellt werden. Diese behandeln die Integration von Sicherheitsprodukten in die IT-Infrastruktur sowie den Zugriff auf die Sicherheitsfunktionalität über dedizierte

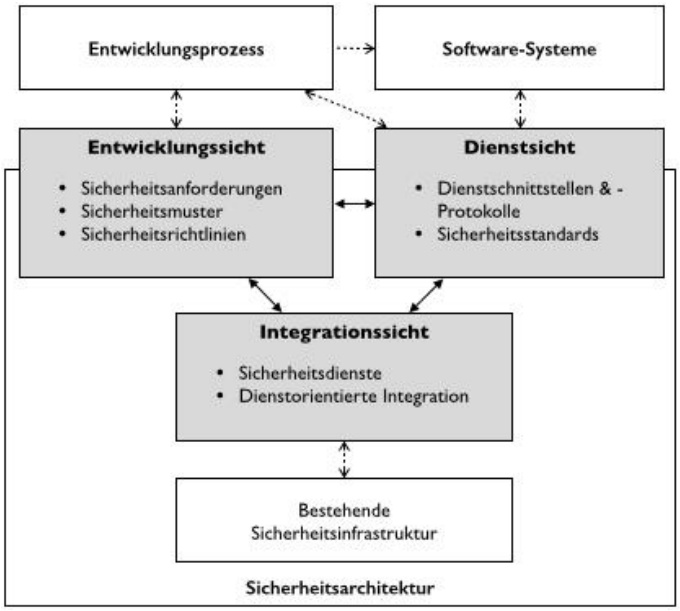

Abbildung 19: Sichtenbasiertes Referenzmodell für eine Sicherheitsarchitektur

Schnittstellen und Protokolle. Da die Sichten eine Einsicht auf die gleiche Sicherheitsarchitektur bieten, sind sie aufeinander abgestimmt. Diese Eigenschaft kann zur zukünftigen Aufbereitung des Sicherheitswissens genutzt werden, da eine Spezifikation der Sicherheitsarchitektur gemäß der Integrations- bzw. Dienstsicht Rückschlüsse auf die Spezifikation der Entwicklungssicht zulässt.

Integrationssicht

Die Integrationssicht des Referenzmodells einer Sicherheitsarchitektur befasst sich mit der Zentralisierung und konsistenten Verwendung von bestehenden Sicherheitsprodukten. Bei der Implementierung von Sicherheitsfunktionalität sollte auf fehleranfällige Eigenimplementierungen verzichtet und stattdessen existierende Sicherheitsprodukte eingesetzt werden. Eine Eigenimplementierung ist nur dann sinnvoll, wenn die erforderliche Sicherheitsfunktionalität nicht bereits durch bestehende Produkte umgesetzt wird. Hinzu kommt, dass aufgrund der Komplexität von Sicherheitsfunktionalität der Kostenaufwand für die Entwicklung oft nicht gerechtfertigt ist.

Des Weiteren führt die Eigenentwicklung von Sicherheitsmaßnahmen zumeist dazu, dass das Prinzip einer offenen Implementierung nicht befolgt wird (engl. open design [Sc01:334] [VG02:70ff]). Eine offene Implementierung von Sicherheitsmaßnahmen setzt den Einsatz von bekannten und bewährten Methoden voraus, indem vor allem existierende Standards und Sicherheitsmodelle verwendet werden. Durch eine offene Umsetzung von Sicherheitsmaßnahmen können darin enthaltene Sicherheitslücken von Sicherheitsexperten analysiert und anschließend behoben werden, welches bei einer verdeckten Implementierung nicht der Fall ist. Eine solche offene Implementierung wird von bestehenden Sicherheitsprodukten und -rahmenwerken verfolgt, welche durch ihren langjährigen Einsatz in vielfältigen Bereichen eine ausgereifte und getestete Sicherheitsfunktionalität bereitstellen.

Ausgehend von dieser Prämisse ist weiter zu beachten, dass üblicherweise mehrere heterogene Sicherheitsprodukte eingesetzt werden. Durch eine im Idealfall verfolgte mehrschichtige Sicherheitsstrategie wird Sicherheitsfunktionalität in verschiedenen Netz-, System- und Anwendungsschichten bereitgestellt [Ra02:72]. Die Vielzahl dieser Sicherheitsprodukte wird in dieser Arbeit als Sicherheitsinfrastruktur bezeichnet. Für eine strukturierte Verwendung dieser Sicherheitsprodukte sowie deren Berücksichtigung in einem Software-Entwicklungsprozess ist jedoch eine homogene Ansicht erforderlich, wodurch die Integration der Produkte benötigt wird. Diesbezüglich hat sich insbesondere

bei der fachlichen Funktionalität ein dienstorientierter Ansatz etabliert, bei welchem die durch verschiedene Produkte bereitgestellte Funktionalität durch einheitliche Dienstschnittstellen gekapselt wird [KB+07]. Eine solche dienstorientierte Integration ist auch für die Sicherheitsfunktionalität sinnvoll. Eine Integration von Sicherheitsfunktionalität in eine dienstorientierte Architektur wurde bereits in anderen Arbeiten vorgestellt [EB+07] [ES+06] [DE+09], welche die Grundlage für die Integrationssicht des Referenzmodells darstellen.

In diesem Zusammenhang ist, wie in Abbildung 20 dargestellt, das Ziel einer Integrationssicht eine dienstorientierte Restrukturierung der bestehenden Sicherheitsinfrastruktur in zentralisierte Sicherheitsdienste. Eine Zentralisierung der Dienste fördert die von Parnas geforderte Trennung von Zuständigkeiten [Pa72] zwischen fachlichen und utilitären Diensten, zu welchen Sicherheitsdienste gezählt werden [KB+07]. Eine solche Restrukturierung bedarf eines sehr hohen technischen Wissens über die Sicherheitsdomäne und über die Sicherheitsprodukte, weshalb sie durch entsprechende Experten durchgeführt werden muss. Folglich wendet sich die Integrationssicht einer Sicherheitsarchitektur an Sicherheitsexperten. Eine zentrale und homogene Sicherheitsinfrastruktur ist die Grundlage zur Extraktion von Sicherheitswissen aus den Sicherheitsprodukten sowie zur Integration der Produkte in einen Software-Entwicklungsprozess.

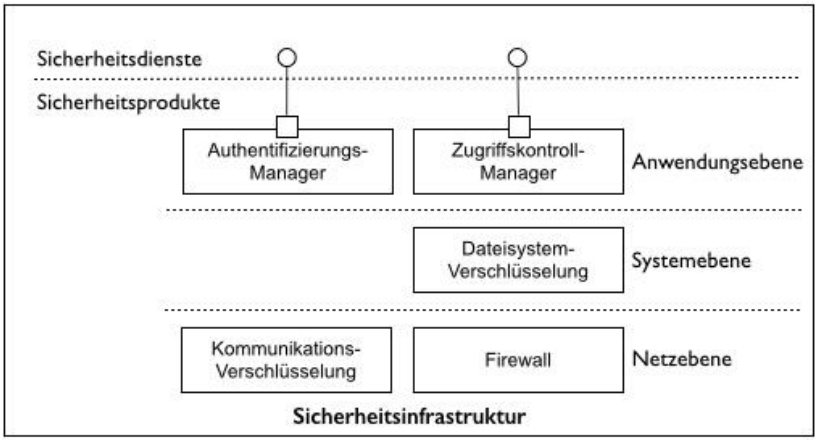

Abbildung 20: Elemente der Integrationssicht

Dienstsicht

Die in der Integrationssicht zentralisierte Sicherheitsfunktionalität ist Grundlage und Voraussetzung für eine detaillierte Betrachtung der Sicherheitsdienste. In der Dienstsicht wird von der technologiespezifischen Sicherheitsfunktionalität abstrahiert und die Verwendung sowie der Einsatzzweck der Sicherheitsdienste spezifiziert. Hierzu werden die zur Verfügung stehenden Schnittstellen der Dienste spezifiziert und deren Nutzung in Form von bewährten Methoden erläutert. Zur Beschreibung der Schnittstellen sollten dabei vorzugsweise angemessene Sicherheitsstandards eingesetzt werden. Dieser Sachverhalt ist in Abbildung 21 dargestellt.

Der Einsatz von Standards ermöglicht zum einen die Interoperabilität zwischen verschiedenen Sicherheitsprodukten und verschiedenen Software-Systemen. In einem betrieblichen Umfeld ist dies insbesondere notwendig, da mehrere Systeme innerhalb und außerhalb von Organisationsgrenzen miteinander interagieren und hierzu Sicherheitsinformationen austauschen müssen. Zum anderen lässt der Einsatz von Sicherheitsstandards den Austausch von Sicherheitsprodukten zu, welche diese Standards umsetzen. Ebenso wie die technischen Details der Sicherheitsprodukte sind Sicherheitsstandards ebenfalls von komplexer und umfangreicher Natur, da sie für eine Vielzahl von unterschiedlichen Einsatzszenarien konzipiert sind. Daher muss die korrekte Verwendung der Standards

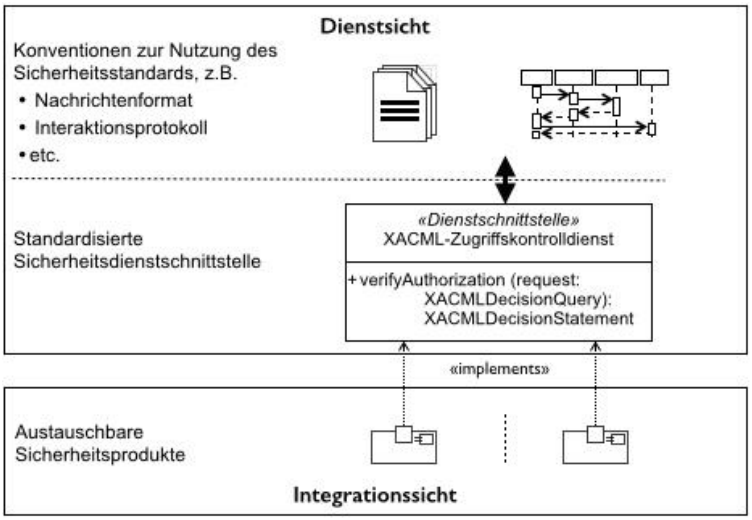

Abbildung 21: Elemente der Dienstsicht

durch Sicherheitsexperten so spezifiziert werden, dass Entwickler von fachlichen Software-Systemen die Sicherheitsdienste mittels der dokumentierten Standards direkt nutzen können.

Darüber hinaus lässt sich eine Unterscheidung zwischen Sicherheitsprodukten und -standards hinsichtlich der Einsatzart in Software-Systemen durchführen. Einerseits werden Sicherheitsprodukte bzw. -dienste zur Auslagerung von Sicherheitsfunktionalität aus Software-Systemen genutzt, wodurch diese eine direkte Assoziation zu den genutzten Sicherheitsdiensten haben. Ein typisches Beispiel hierfür ist die Auslagerung von Authentifizierungs- und Zugriffskontrollfunktionalität. Andererseits können Sicherheitsdienste identifiziert werden, welche nicht direkt durch fachliche Software-Systeme genutzt werden, jedoch zusätzliche Sicherheitsfunktionalität bereitstellen. Solche Dienste befinden sich üblicherweise auf einer niedrigeren technologischen Abstraktionsschicht und betreffen verschiedene Systemdienste, welche von Software-Systemen genutzt werden. Beispiele hierfür stellen Firewalls dar, welche den Zugriff auf Verbindungsprotokollebene regulieren, sowie verschlüsselte Kommunikationskanäle, welche die Integrität und Vertraulichkeit von übermittelten Daten garantieren.

Der Einsatz von Standards zur Spezifikation des Einsatzes von Sicherheitsdiensten ist in diesen Fällen oftmals optional, da für verschiedene Sicherheitsbereiche gegebenenfalls keine Standards und Schnittstellen existieren. In diesem Zusammenhang ist zunächst der Einsatz von Konventionen zur Konfiguration und Verwendung der Sicherheitsdienste sinnvoll. Werden Sicherheitsdienste nicht direkt durch ein Software-System genutzt und wirken sich die Effekte der Schutzmaßnahmen eines Sicherheitsdienstes indirekt auf die Funktionalität eines Software-Systems aus, so sollten diese Auswirkungen dokumentiert werden. Hierdurch werden fachliche Entwickler auf die Wirkungen aufmerksam gemacht und das Software-System kann darauf abgestimmt werden.

Die Dienstsicht stellt einem Software-Entwickler somit die grundlegenden Informationen über die Funktionalität von Sicherheitsdiensten für Software-Systeme bereit. Die Nutzung wird dabei in Form von standardisierten Schnittstellenbeschreibungen und der Spezifikation von Protokollen zur Interaktion mit den Schnittstellen der Sicherheitsdienste ermöglicht. Ebenso werden die zur Verfügung stehenden Dienste aufgezählt, wodurch Entwicklern eine Übersicht über die Sicherheitsdienste gegeben wird. Im Gegensatz zur Integrationssicht richten sich die in der Dienstsicht verfügbaren Artefakte an fachliche Software-Entwickler.

Entwicklungssicht

Die Integrations- und die Dienstsicht stellen vielfältige Informationen bereit, welche für den Einsatz in einem Software-Entwicklungsprozess genutzt werden können, um eine effiziente Durchführung einer

Anforderungsanalyse und die Modellierung von sicherheitsrelevanter Funktionalität zu ermöglichen. Da die Sicherheitsprodukte bereits vorhanden sind und gegebenenfalls bereits betrieben werden, reduziert sich in der Implementierungsphase eines Software-Systems die Entwicklung von Sicherheitsfunktionalität auf die Anpassung an die Sicherheitsdienste. Dies lässt sich zum Beispiel durch die Implementierung von geeigneten Adaptern zur Interaktion mit den Sicherheitsdiensten umsetzen. Abbildung 22 stellt die durch die Entwicklungssicht bereitgestellten Artefakte sowie den Zusammenhang zur Dienstsicht dar.

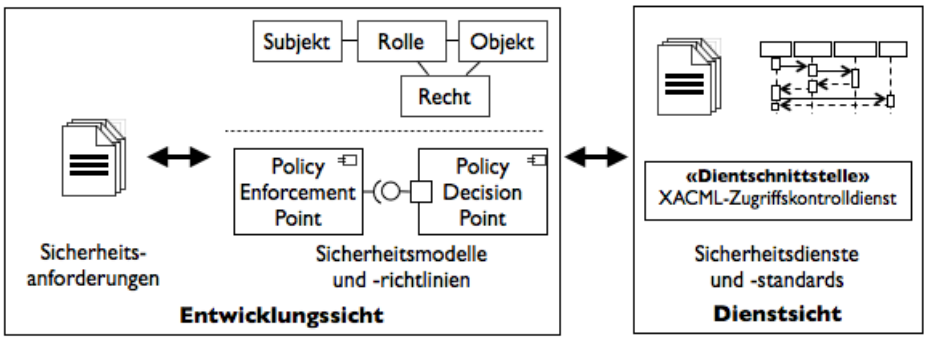

Abbildung 22: Elemente der Entwicklungssicht

Somit beschränkt sich die Aufbereitung von Sicherheitswissen für die sicherheitsbasierte Entwicklung auf Grundlage der Sicherheitsarchitektur zunächst auf die Beschreibung von Analyse- und Modellierungselementen, welche die wesentlichen Bestandteile der Entwicklungssicht des Referenzmodells darstellen. Ausgehend von der Dienstsicht ist es zur Aufbereitung von Sicherheitswissen naheliegend, von den technologischen Sicherheitsdiensten zu abstrahieren und die damit verbundenen Sicherheitskonzepte und -lösungen in Form von Entwurfselementen technologieunabhängig zu spezifizieren. Diese Elemente lassen sich anschließend in der Entwurfsphase als vorgegebene Strukturen in das Modell eines Software-Systems eingefügen.

Hierzu zählen zum einen statische Strukturen, wie beispielsweise Sicherheitskomponenten und deren Beziehungen untereinander, sowie deren Assoziation zu fachlichen Komponenten. Zum anderen sind die dynamischen Interaktionen zwischen den Komponenten zu modellieren, so dass die Protokolle für die Verwendung der Sicherheitskomponenten spezifiziert werden. Eine solche Ableitung bedarf detaillierter Kenntnisse von Sicherheitsexperten, welche durch eine rückwärtige Entwicklung (engl. reverse engineering) eine Abbildung von den in einem Sicherheitsprodukt implementierten Funktionen auf bestehende technologieunabhängige Sicherheitslösungen und -praktiken vornehmen müssen.

Sicherheitsrichtlinien stellen ein weiteres Modellierungselement für sicherheitsrelevante Funktionen dar, da sie die Funktionalität der Sicherheitskomponenten beeinflussen. So wird zum Beispiel die Entscheidung eines Zugriffskontrolldienstes durch die zugrundeliegenden Zugriffskontrollrichtlinien bestimmt. Eine Wiederverwendung von existierenden Sicherheitsrichtlinien, welche bereits in den existierenden Sicherheitsprodukten und -diensten eingesetzt werden, bietet eine Unterstützung für Entwickler bei der Umsetzung von Sicherheitsrichtlinien für neue Software-Systeme. Werden beispielsweise die Sicherheitsmodelle der rollenbasierten und der attributbasierten Zugriffskontrolle durch Sicherheitsprodukte unterstützt, so ist dies ausschlaggebend für das Sicherheitsmodell, welches in einem neu entwickelten Software-System eingesetzt wird. Hierzu muss entsprechend zu den Komponenten der konkrete Einsatz bzw. die Konventionen zur Spezifikation der Richtlinien sowie Vor- und Nachteile der Sicherheitsrichtlinien dokumentiert werden. Sicherheitskomponenten und -richtlinien werden hauptsächlich in der Entwurfsphase eines Entwicklungsprozesses eingesetzt und

werden, im Gegensatz zu den Sicherheitstechnologien der Implementierungsphase, im Rahmen dieser Arbeit als Sicherheitsmaßnahmen bezeichnet.

Parallel hierzu können Sicherheitsprinzipien, welche bewährte Grundregeln für einen sicherheitsbasierten Entwurf zusammenfassen, für die Konstruktion von Sicherheitsrichtlinien eingesetzt werden. Zu diesen Sicherheitsprinzipien gehören unter anderem die Prinzipien der minimalen Rechte, Sicherung des schwächsten Gliedes und gestaffelte Abwehr [Sc01:358ff]. Neben der eingesetzten Infrastruktur können in der Entwicklungssicht auch Sicherheitsrichtlinien vermerkt werden, welche durch geschäftliche bzw. rechtliche Vorgaben eingehalten werden müssen.

Um in einem Software-Entwicklungsprozess eine Analyse von Sicherheitsanforderungen zu unterstützen, ist zusätzlich die Ableitung von wiederverwendbaren Sicherheitsanforderungen aus den Entwurfselementen nötig. Sicherheitsanforderungen stellen dabei im Allgemeinen Einschränkungen an die fachliche Funktionalität dar und stellen noch keine Lösungsmöglichkeiten für vorhandene Sicherheitsprobleme bereit [Fi04] [SF+03]. Durch eine eindeutige Analyse werden die notwendigen Sicherheitsanforderungen sowie der Schutzbedarf eines Software-Systems festgelegt. Erst hieraus lassen sich effiziente und geeignete Schutzmaßnahmen ableiten, welche die Anforderungen umsetzen.

Eine Wiederverwendung von Sicherheitsanforderungen ermöglicht hierbei den Einsatz von gesammelten Erfahrungen und bewährten Praktiken bei der Erhebung und Analyse von Bedrohungen, wodurch der Analyseprozess effizienter gestaltet wird. Damit die Abbildung von Sicherheitsanforderungen auf Sicherheitsmaßnahmen in der Entwurfsphase definiert ist, müssen diese Elemente explizit in Verbindung gebracht werden. Dies resultiert in einer m:n-Beziehung zwischen Sicherheitsanforderungen und -maßnahmen, in welcher eine Sicherheitsanforderung durch mehrere Maßnahmen umgesetzt wird und eine Maßnahme mehrere Anforderungen abdeckt.

Die Analyse- und Entwurfsartefakte werden dabei separat von den fachlichen Modellen spezifiziert, da bei einer Vermischung die Komplexität der so entstehenden Modelle zunimmt. Eine separate Modellierung erlaubt zudem die Trennung von Zuständigkeiten und somit die unabhängige Weiterentwicklung von sicherheitsrelevanten und fachlichen Modellen. Die Modellbereiche, in welchen sich fachliche und sicherheitsrelevante Funktionalität überschneiden, werden als eigenständige Modelle bzw. Diagramme entworfen. Hierdurch bleibt die Komplexität handhabbar und die Aussage der Modelle ist eindeutig.

Die Entwicklungssicht des Referenzmodells für Sicherheitsarchitekturen stellt Entwicklungsartefakte bereit, welche auf bestehenden Sicherheitsinfrastrukturen fundiert sind. Somit wird eine effiziente Entwicklung von Sicherheitsmaßnahmen in Software-Systemen ermöglicht, da bestehende Sicherheitsanforderungen, -modelle und -richtlinien sowie Funktionalität in Form von Sicherheitsdiensten und -produkten wiederverwendet werden können. Durch die Ableitung der Entwicklungsartefakte aus einer betrachteten Sicherheitsarchitektur lassen sich die so entwickelten Software-Systeme in bestehende Sicherheitsinfrastrukturen integrieren.

4.1.2 Ablauf der Entwicklungsmethode

Eine sicherheitsbasierte Software-Entwicklung betrachtet die qualitative Eigenschaft der Sicherheit der Software entlang aller Phasen eines Entwicklungsprozesses. Wie in Kapitel 2 besprochen, wird hierzu die Sicherheitsfunktionalität parallel zur fachlichen Funktionalität entwickelt, in dem analog zur fachlichen Entwicklung die Anforderungen und Funktionen der Sicherheitskomponenten analysiert, modelliert und anschließend implementiert werden. Somit stellt eine sicherheitsbasierte Entwicklung nicht einen eigenständigen, von einer fachlichen Domäne losgelösten Entwicklungsprozess dar. Stattdessen gliedert sie sich mit ihren Aktivitäten, Maßnahmen und Methoden in Letzteren ein und erweitert ihn [Ec09] [An08]. Dadurch ergibt sich folglich eine Abhängigkeit zu den jeweils eingesetzten

fachlichen Entwicklungsprozessen, an welchen sich die Entwicklung der Sicherheitsfunktionalität orientieren muss. Um in verschiedenartigen Entwicklungsprozessen eingesetzt zu werden, müssen die Aktivitäten des im Folgenden vorgestellten sicherheitsbasierten Entwicklungsvorgehen entsprechend flexibel gestaltet werden.

Die Wiederverwendung von Sicherheitswissen in einem sicherheitsbasierten Entwicklungsvorgehen benötigt eine zu dem im Kapitel 2.2 beschriebenen generischen Ablauf erweiterte Vorgehensweise, in welcher die Aufbereitung des Sicherheitswissens behandelt wird. Die zuvor beschriebenen Sichten des Referenzmodells für eine Sicherheitsarchitektur sind hierbei Ausgangspunkt für das im Weiteren vorgestellte sicherheitsbasierte Entwicklungsvorgehen. Die darin durchgeführten Aktivitäten haben zum Ziel, die Entwicklungsartefakte gemäß der Entwicklungssicht aus der betrachteten Sicherheitsarchitektur zu spezifizieren. Im Folgenden wird eine konzeptionelle Übersicht über die Abläufe der Entwicklungsmethodik dargestellt. Ausgehend von dem Referenzmodell wird vorausgesetzt, dass Sicherheitsfunktionalität nicht erneut implementiert werden muss, so dass vorrangig die Analyse- sowie die Entwurfsphase betrachtet wird. Zudem wird auf eine Unabhängigkeit von konkreten fachlichen Entwicklungsprozessen Rücksicht genommen, so dass das Vorgehen an ein spezifisches Prozessmodell angepasst werden kann.

Wie in Abbildung 23 dargestellt, wird das Vorgehen dabei prinzipiell in zwei Phasen eingeteilt. In einer Aufbereitungsphase wird existierendes Sicherheitswissen durch Experten analysiert und für die Anwendung in einem Software-Entwicklungsprozess gemäß der diskutierten Entwicklungssicht vorbereitet. Durch diese Aufbereitung wird das Referenzmodell aufgebaut und kontinuierlich weiterentwickelt. Anschließend wird dieses aufbereitete Wissen in einer Anwendungsphase in einem konkreten Software-Entwicklungsprozess eingesetzt, um eine strukturierte und effiziente Entwicklung der Sicherheitsmaßnahmen des zu entwickelnden Software-Systems zu ermöglichen. Im folgenden Abschnitt wird insbesondere das Zusammenspiel der beiden Phasen und der konzeptionelle Rahmen des Entwicklungsvorgehens vorgestellt. Die detaillierte Beschreibung der einzelnen Phasen inklusive der darin ablaufenden Aktivitäten und der Entwicklungsartefakte wird in den darauffolgenden Kapiteln 4.2 und 4.3 dargestellt.

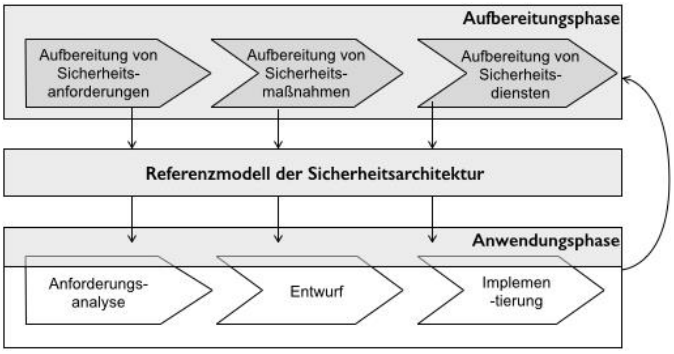

Abbildung 23: Zusammenspiel der Aufbereitungs- und Anwendungsphase

Aufbereitungsphase

Die Aufbereitungsphase fokussiert die Erstellung sowie die darauffolgende Pflege und Evolution des Referenzmodells der Sicherheitsarchitektur. Zur Erstellung lassen sich dabei verschiedene Ansätze verfolgen. Zunächst kann das bestehende Sicherheitswissen unabhängig von den zu entwickelnden Software-Systemen gesammelt werden. Hierdurch wird das initiale Spektrum der Sicherheitsanforderungen und -maßnahmen der Architektur sehr breit angelegt, wodurch eine Vielzahl von unterschiedli-

chen Software-Systemen unterstützt werden kann. Der Nachteil ist dabei der initiale Aufwand der Aufbereitung, welcher zu hohen Kosten führt.

Im Gegensatz dazu kann ein erstes Referenzmodell auch während der Entwicklung der Sicherheitsarchitektur eines bestimmten Software-Systems erstellt werden. Hierdurch werden zunächst die Kosten zur Erstellung des Referenzmodelles reduziert. Jedoch reduziert sich ebenfalls der Umfang der sicherheitsbezogenen Entwicklungsartefakte, welche lediglich auf das spezifische Software-System ausgerichtet und somit gegebenenfalls nicht für eine Wiederverwendung in anderen Software-Systemen geeignet sind. Die Kosten und der Aufwand zur Erstellung werden somit auf die zukünftige Evolution der Architektur verteilt und die Vorteile einer umfangreichen Sammlung an wiederverwendbaren Sicherheitsmaßnahmen bei der Software-Entwicklung eingeschränkt.

Ein hybrider Ansatz hierzu ist die Untersuchung einer eingeschränkten Zahl an bereits entwickelten Software-Systemen hinsichtlich der ausgewählten Sicherheitsmaßnahmen und -anforderungen. In diesem Fall werden aus den untersuchten Software-Systemen die gemeinsamen sowie alternativen Sicherheitsmaßnahmen extrahiert und wiederverwendbar aufbereitet. Gemeinsamkeiten deuten dabei auf Sicherheitsmaßnahmen hin, welche typischerweise in jedem der Software-Systeme eingesetzt wurden. Alternativen oder optionale Elemente sind dazu abweichende Elemente, welche zur Umsetzung von speziellen Sicherheitsmaßnahmen oder in Ausnahmefällen in einzelnen Software-Systemen eingesetzt werden, um erweiterte Sicherheitsanforderungen umzusetzen. Hierbei ist die Prämisse, dass die bereits entwickelten Software-Systeme einen ähnlichen fachlichen Kontext besitzen. Somit wird ein Referenzmodell anhand einer Sicherheitsarchitektur erstellt, welche eine breitere Palette an wiederverwendbaren Sicherheitsmaßnahmen und -anforderungen umfasst, jedoch nach wie vor auf einen fachlichen Kontext spezialisiert ist. Die Kosten und der Umfang der wiederverwendbaren Elemente des Referenzmodells hängen dabei von der Anzahl der zusammen untersuchten Software-Systeme ab.

In dem sicherheitsbasierten Entwicklungsvorgehen wird zunächst keiner der Ansätze präferiert. Die Auswahl eines Ansatzes zur initialen Erstellung der Architektur und des Referenzmodells hängt von verschiedenen individuellen Faktoren ab, zu welchen an dieser Stelle keine Aussage gemacht werden kann. Hierzu zählen unter anderem die initial vorhandenen Mittel und Ressourcen zur Erstellung des Referenzmodells sowie des geplanten Einsatzes der Sicherheitsarchitektur. Sollte beispielsweise nur eine beschränkte Anzahl von ähnlichen Software-Systemen entwickelt werden oder nur geringe verfügbare Mittel vorhanden sein, so ist der zweite Ansatz eventuell vorzuziehen. Aus den verschiedenen Ansätzen entstehen für ein Software-System zunächst unterschiedlich umfangreiche Referenzmodelle, welche jedoch durch die kontinuierliche Evolution und Pflege der Sicherheitsarchitektur angeglichen werden.

Zur Evolution der Architektur muss neues Sicherheitswissen kontinuierlich aufbereitet und eingepflegt werden. Neues Sicherheitswissen lässt sich einerseits durch die in der Entwicklung eines neuen Software-Systems gesammelten Erfahrungen hinsichtlich der benötigten Sicherheitsfunktionalität erfassen. Dies ist insbesondere der Fall, wenn die im Referenzmodell vorhandenen Entwicklungsartefakte nicht den Schutzbedarf des Software-Systems umfassen. In diesem Fall müssen alternative Sicherheitsmaßnahmen entwickelt werden, welche im Anschluss aufbereitet und in das Referenzmodell zur zukünftigen Nutzung in weiteren Software-Systemen integriert werden. Bei der Integration werden diese zusätzlichen Maßnahmen als Alternativen zu den bestehenden Sicherheitsmaßnahmen dokumentiert. Folglich müssen die Vor- und Nachteile der eingesetzten Maßnahmen sowie deren Einsatzkontext, welcher zur Auswahl der Sicherheitsmaßnahmen führte, dokumentiert werden.

Andererseits sind neben der internen Revision der Sicherheitsanforderungen und -maßnahmen eines entwickelten Software-Systems auch externe Impulse bei der Evolution der Sicherheitsarchitektur und des zugehörigen Referenzmodells ausschlaggebend. Hierzu gehört zunächst die Überprüfung auf bekanntgewordene Sicherheitslücken bei den eingesetzten Sicherheitsprodukten. Dies umfasst zudem die Einspielung von veröffentlichten Aktualisierungen zur Behebung von Sicherheitslücken. Ebenso zählt bei Bedarf auch der Einsatz eines Standards hinzu, welcher für einen Sicherheitsdienst spezifiziert, jedoch noch nicht in der Sicherheitsarchitektur eingesetzt wurde. Hierbei ist die Umsetzung des Standards mit den vorhandenen Sicherheitsprodukten zu prüfen, wodurch diese gegebenenfalls aktualisiert oder ausgetauscht werden müssen.

Ebenso sind technologieunabhängige Sicherheitsmodelle als Alternativen oder als Ersatz zu den verwendeten Modellen zu überprüfen. Dies kann zum einen durch eine effizientere Umsetzung von Sicherheitsanforderungen in einem bereits entwickelten Software-System motiviert sein. Zum anderen kann auch die effizientere Durchführung von sicherheitsrelevanten Prozessen durch alternative Sicherheitsmodelle ausschlaggebend sein. Ein Beispiel hierfür ist die erzielte Komplexitätsreduktion bei der Verwaltung von Zugriffsrechten bei einem Wechsel von Zugriffskontrolllisten auf eine rollenbasierte Zugriffskontrolle. Die Sicherheitsmodelle müssen zudem mit den bestehenden Sicherheitsanforderungen in Verbindungen gebracht werden, so dass deren Einsatz im Zusammenhang mit bestehenden Anforderungen spezifiziert wird.

Neben den Sicherheitsmodellen, welche hauptsächlich in der Entwurfsphase eingesetzt werden, müssen auch sich ändernde oder neue Sicherheitsanforderungen ergänzt werden. Die Notwendigkeit zur Aktualisierung der Sicherheitsanforderungen kann wiederum von einem zu entwickelnden Software-System ausgehen, falls das bestehende Referenzmodell nicht die nötigen Anforderungen beinhaltet. Ebenso können zusätzliche Anforderungen aus Änderungen an den geschäftlichen Sicherheitsrichtlinien sowie aus gesetzlichen Regelungen stammen, welche die Umsetzung von spezifischen Sicherheitsmaßnahmen erfordern. Die Aktivitäten zur Aufbereitung von Sicherheitswissen sowie die relevanten Quellen zur Aufbereitung sind in Abbildung 24 zusammenfassend dargestellt.

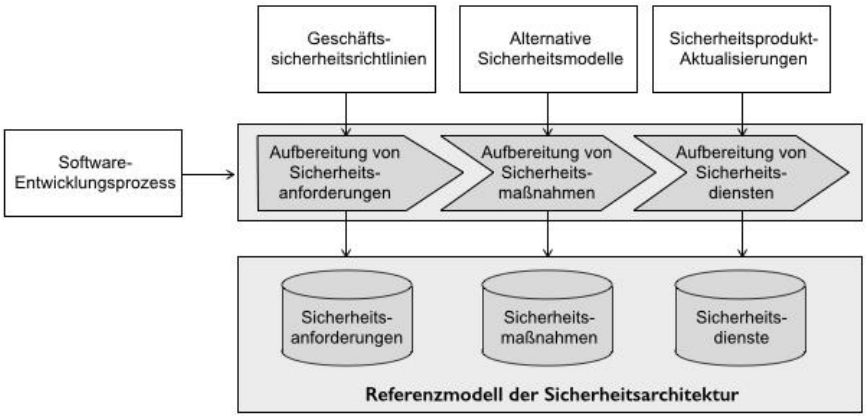

Abbildung 24: Aktivitäten und Artefakte der Aufbereitungsphase

Das Hinzufügen von neuen Sicherheitsanforderungen zum Referenzmodell erfordert die Inspektion der Sicherheitsarchitektur hinsichtlich der Sicherheitsmaßnahmen und der Sicherheitsprodukte, mittels welchen die Anforderungen umgesetzt werden können. Sind entsprechende Maßnahmen nicht bereits zur Verwendung vorhanden, so ist es erforderlich, die passenden Sicherheitsmaßnahmen zu modellieren und in die Sicherheitsarchitektur zu integrieren. Ebenso müssen verfügbare Sicherheitsprodukte

auf Umsetzbarkeit der neuen Maßnahmen überprüft und wenn nötig die Sicherheitsinfrastruktur durch Erweiterung der eingesetzten Sicherheitsprodukte ergänzt werden.

Anwendungsphase

Während die Aufbereitungsphase die Bereitstellung von wiederverwendbaren Entwicklungsartefakten in einem Referenzmodell einer Sicherheitsarchitektur fokussiert, ist das Ziel der Anwendungsphase der Einsatz dieser Artefakte zur Entwicklung von Sicherheitsfunktionalität für ein Software-System. In einem traditionellen Ansatz wird Sicherheitsfunktionalität parallel zur fachlichen Funktionalität entwickelt. Im Gegensatz hierzu konzentriert sich das vorgestellte Entwicklungsvorgehen auf die Wiederverwendung von sicherheitsrelevanten Entwicklungsartefakten und deren Einbettung in den fachlichen Entwicklungsprozess.

Zur Einbettung der Aktivitäten der Anwendungsphase wird in dieser Arbeit kein konkreter Software-Entwicklungsprozess vorausgesetzt, sondern es wird von einem abstrakten Entwicklungsprozess ausgegangen. Dieser abstrakte Prozess teilt die fachliche Entwicklung in die generischen Phasen Anforderungsanalyse, Entwurf und Implementierung ein [So07]. Diese Phasen werden üblicherweise in allen Software-Entwicklungsprozessen durchlaufen, wobei die Länge und Intensität der in den Phasen durchgeführten Aktivitäten unterschiedlich ausfallen sowie die Phasen auch mehrmals durchlaufen werden können. Das sicherheitsbasierte Entwicklungsvorgehen ist somit darauf ausgelegt, an einen spezifischen Entwicklungsprozess angepasst zu werden. Die Phasen sowie deren Ergänzung durch die Artefakte des Referenzmodells sind in Abbildung 25 dargestellt.

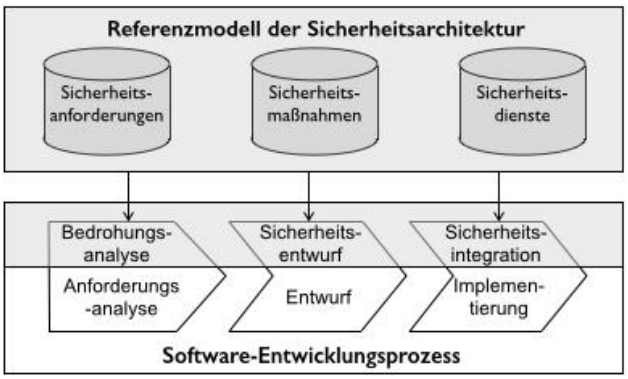

Abbildung 25: Aktivitäten und verwendete Artefakte in der Anwendungsphase

Die Test- und die Betriebsphase werden in dem vorgestellten Entwicklungsvorgehen nicht weiter untersucht. In einer Testphase wird nach wie vor die Erfüllung der Sicherheitsziele und die Umsetzung der Sicherheitsanforderungen durch die vorgegebene Sicherheitsfunktionalität der Architektur überprüft. Hierzu können bekannte Verfahren, wie die Simulation von Angriffen (engl. penetration testing [MC04]), eingesetzt werden. Durch das Referenzmodell wird die Prämisse vorgegeben, dass die benötigten Sicherheitsprodukte in einer Sicherheitsinfrastruktur bereitgestellt werden. Zudem wird davon ausgegangen, dass für die Betriebsphase die notwendigen IT-Managementprozesse für die Produkte etabliert und durchgeführt wurden. Wie im vorangegangenen Abschnitt beschrieben, ist die Rückmeldung von Bedrohungen und Angriffen auf ein Software-System sowie die nachfolgende Behandlung des Sicherheitsproblems ein Bestandteil der Aufbereitungsphase und dient der Evolution des Referenzmodells.

In der Phase der Anforderungsanalyse werden die für das zu entwickelnde Software-System notwendigen Sicherheitsanforderungen bestimmt. Hierzu werden zunächst die zu schützenden fachlichen Ressourcen in dem Software-System bestimmt. Anschließend werden der Schutzbedarf und die

verfolgten Schutzziele für jede Ressource festgelegt. Dieser aufwendige Prozess wird durch die wiederverwendbaren Sicherheitsanforderungen des Referenzmodelles unterstützt. Hierbei werden die Sicherheitsanforderungen, welche durch die Sicherheitsarchitektur bereits umgesetzt werden, auf Anwendbarkeit auf die identifizierten Ressourcen untersucht. Mit den Sicherheitsanforderungen sind weitere Informationen verbunden, wie beispielsweise der zugehörige Schutzbedarf und die umzusetzenden Schutzziele, wodurch die Spezifikation von Sicherheitsanforderungen vereinfacht wird.

Die Sicherheitsanforderungen sind innerhalb des Referenzmodells mit geeigneten Sicherheitsmaßnahmen verknüpft, welche in der Entwurfsphase zur Modellierung von Sicherheitsfunktionalität eingesetzt werden können. Bietet das Referenzmodell hierbei alternative Maßnahmen an, so können je nach Kontext des zu entwickelnden Software-Systems eine oder mehrere Maßnahmen ausgewählt werden. Hierzu werden die Spezifikationen zum Einsatz der Sicherheitsmaßnahmen aus dem Referenzmodell mit dem Kontext des Software-Systems verglichen. Nach Auswahl der Alternative wird diese anschließend als Teil der konkreten für das Software-System genutzten Sicherheitsarchitektur instanziiert. Im Gegensatz dazu stellen die zuvor angesprochenen gemeinsamen Sicherheitsmaßnahmen des Referenzmodells in diesem Fall keine optionalen Maßnahmen dar, sondern müssen zur Umsetzung der Sicherheitsanforderungen eingesetzt werden. Auf diese Weise lassen sich globale Sicherheitsstrategien, wie sie beispielsweise in einer Organisation vorgegeben sind, in allen Software-Systemen umsetzen.

Da in dem Referenzmodell existierende Sicherheitsprodukte und -rahmenwerke als technologische Basis eingesetzt werden, reduziert sich die Umsetzung von Sicherheitsmaßnahmen auf die Integration von Sicherheitsfunktionalität durch Implementierung von geeigneten Adaptern. Hinsichtlich der Sicherheitsrichtlinien müssen zudem noch die entsprechenden standardisierten bzw. produktspezifischen Richtlinien in die jeweiligen Produkte eingepflegt werden. Das Referenzmodell gibt in beiden Fällen konkrete Vorgaben, mit Hilfe derer diese Artefakte sowohl technologieunabhängig modelliert als auch technologiespezifisch umgesetzt werden sollten. Hierdurch lassen sich auch Verfahren zur automatischen Generierung der Artefakte, wie sie zum Beispiel in der modellgetriebene Software-Entwicklung (engl. Model-Driven Software Development, MDSD [EK+08] [LB+02]) zum Einsatz kommen, nutzen, um Fehler bei der manuellen Implementierung zu vermeiden. Diese Ansätze werden in dieser Arbeit jedoch nicht weiter betrachtet. Das Ziel der Implementierungsphase ist somit die Integration des Software-Systems in die bestehende Sicherheitsinfrastruktur.

4.2 Aufbereitung von Sicherheitswissen

Das Ziel der Aufbereitungsphase der Entwicklungsmethode ist die Aufbereitung von Sicherheitswissen aus verschiedenen Quellen, so dass es in einem Software-Entwicklungsprozess eingesetzt werden kann. Als Quellen zählen hierbei unter anderem die in der Literatur dokumentierten sowie durch Sicherheitsexperten gesammelten Erfahrungen an bewährten Methoden zur Lösung von Sicherheitsproblemen. Ebenso müssen die existierenden und gegebenenfalls bereits eingesetzten Sicherheitsprodukte bei der Aufbereitung des Sicherheitswissens berücksichtigt werden. Das Ergebnis der Aufbereitung ist eine Strukturierung des Sicherheitswissens und der Sicherheitsprodukte in aufeinander abgestimmte Sichten im Sinne des vorgestellten Referenzmodells für Sicherheitsarchitekturen. In Anlehnung an den Domänen-Entwicklungsprozesses für Software-Produktlinien (vgl. Kapitel 2.3 und [PB+05:23f]) wird in der Aufbereitungsphase somit eine Plattform für wiederverwendbare Sicherheitslösungen entwickelt. In dieser Arbeit wird diese Plattform durch das im vorangegangenen Abschnitt beschriebene Referenzmodell bereitgestellt.

Bei der Aufbereitung des Sicherheitswissens kann dabei zunächst auf eine vollständige Spezifikation aller Sicherheitsmaßnahmen verzichtet werden. Stattdessen wird eine iterative Erweiterung des

Referenzmodells durchgeführt, welche abhängig von den Sicherheitsbedürfnissen der vorhandenen bzw. zu entwickelnden Software-Systeme ist. Es lässt sich somit zunächst ein notwendiger Ausschnitt der Sicherheitsfunktionalität als wiederverwendbare Entwicklungsartefakte beschreiben, welche in sich anschließenden Iterationen erweitern werden. Mittels dieses Vorgehens lassen sich zudem auch Sicherheitslösungen aufbereiten, welche durch neu entstandene Bedrohungen erarbeitet wurden und noch nicht zu bewährten Methoden zählen. Eine solche Vorkehrung ist aufgrund der zahlreichen Bedrohungen sinnvoll und berücksichtigt zukünftige Sicherheitsprobleme sowie zugehörige Lösungen.

Durch die Aufbereitung des Sicherheitswissens ergeben sich Entwicklungsartefakte, welche die Sicherheitslösungen im Sinne der Entwicklungssicht des Referenzmodells für Sicherheitsarchitekturen repräsentieren. Diese liefern eine Plattform, auf welcher konkrete Entwicklungsprozesse aufbauen können. Die Entwicklungsartefakte umfassen hierbei sowohl gemeinsame und variable Elemente, welche wiederverwendbar sind und die Durchgängigkeit zwischen mehreren Entwicklungsphasen ermöglichen. Gemeinsame Artefakte stellen dabei verbindlich einzusetzende Sicherheitslösungen dar, wogegen variable Elemente eine Auswahlmöglichkeit zwischen mehreren alternativen Optionen zur Lösung eines Sicherheitsproblems liefern.

4.2.1 Spezifikation von wiederverwendbaren Sicherheitsanforderungen

Die Festlegung von Sicherheitsanforderungen ist eine wesentliche Voraussetzung für die Umsetzung von Sicherheitsfunktionalität, welche an die Schutzbedürfnisse einer Anwendung angepasst ist [TJ+08]. Eine fachliche Anforderung stellt eine von einem Nutzer zur Lösung eines Problems oder zur Erreichung eines Ziels benötigte Bedingung oder Fähigkeit des Software-Systems dar [IEEE-610.12-1990] in [PB+05:91]. Dagegen stellt eine Sicherheitsanforderung eine Einschränkung einer Fähigkeit dar, sodass die Einhaltung von Schutzzielen, wie beispielsweise Integrität und Vertraulichkeit, von im Software-System verarbeiteten Ressourcen gewährleistet werden kann [Fi03a]. Sicherheitsanforderungen beziehen sich somit auf eine fachliche Funktionalität oder Ressource. Dementsprechend ist die Erhebung von Sicherheitsanforderungen eine Herausforderung für unerfahrene Entwickler. Um eine Unterstützung zur strukturierten Analyse von Sicherheitsanforderungen zu ermöglichen, ist es sinnvoll den Entwicklern einen Katalog von vorgegebenen Sicherheitsanforderungen zur Verfügung zu stellen.

Im Folgenden wird hierfür ein Vorgehen aufgezeigt, welches von Sicherheitsexperten angewendet werden kann, um in der Aufbereitung von Sicherheitswissen einen solchen Katalog zu erstellen. Wie bereits beschrieben, lassen sich hierzu im Wesentlichen drei Richtungen identifizieren, welche bei der Erstellung angewendet werden können. Zum einen lassen sich die wiederverwendbaren Sicherheitsanforderungen iterativ aus Software-Systemen extrahieren, welche für einen ähnlichen Kontext entwickelt wurden. Hierdurch umfasst der Katalog nur solche Sicherheitsanforderungen, welche bereits eingesetzt wurden. Zum anderen können Anforderungen unabhängig von konkreten Anwendungen aus bestehenden Verzeichnissen, wie zum Beispiel dem Common-Criteria-Evaluierungshandbuch [CC09] oder dem Open Web Application Security Project (OWASP, [OWASP]) aufbereitet werden, wodurch der Umfang des Katalogs zunächst nicht eingeschränkt ist. In der Praxis hat sich eine Mischung aus beiden Ansätzen etabliert, worauf sich das hier vorgestellte Vorgehen stützt. Die Spezifikation der Sicherheitsanforderungen erfolgt dabei in Form von Vorlagen, welche im Detail im folgenden Kapitel 5 beschrieben werden.

Beziehungen zu anderen Aktivitäten

Das Ergebnis der Aufbereitung von Sicherheitsanforderungen sind wiederverwendbare Vorlagen, welche Sicherheitsanforderungen abstrakt und kontextunabhängig spezifizieren. Diese werden in der

Aufbereitungsphase als Basis für die Beschreibung von wiederverwendbaren Sicherheitsmaßnahmen und somit zur Spezifikation der Sicherheitsfunktionalität innerhalb der Entwicklungssicht des Referenzmodells für Sicherheitsarchitekturen verwendet. Hierzu werden zu jeder der Vorlagen für Sicherheitsanforderungen ein oder mehrere Sicherheitsmaßnahmen ausgewählt, welche die Anforderungen umsetzen. Dabei kann eine Sicherheitsmaßnahme zur Umsetzung von ein oder mehreren Sicherheitsanforderungen eingesetzt werden.

Durch die iterative Verfeinerung der Sicherheitsmaßnahmen steigt der Bedarf nach detaillierteren Sicherheitsanforderungen. Ebenso können die erfassten Sicherheitsanforderungen zu anspruchsvoll definiert sein, so dass keine Sicherheitsmaßnahmen dazu abgeleitet werden können. In diesen Fällen wird eine Rückkopplung ausgelöst und die Aufbereitung der Sicherheitsanforderungen erneut angestoßen. Wird diese Rückkopplung mehrmals durchlaufen, ergeben sich sowohl detailliertere Anforderungsvorlagen als auch verfeinerte und umsetzbare Sicherheitsmaßnahmen [Nu01] in [PB+05:195f].

Die Vorlagen werden zudem in einem Software-Entwicklungsprozess eingesetzt, um eine effiziente Ableitung von spezifischen Sicherheitsanforderungen eines zu entwickelnden Software-Systems zu ermöglichen. In einem Software-Entwicklungsprozess wird eine Bedrohungs- und Risikoanalyse durchgeführt, um die zu schützenden Ressourcen, deren Schutzbedarf und die möglichen Bedrohungen zu bestimmen. In diesem Kontext dienen die Vorlagen als Richtlinie, um aus dem zuvor bestimmten Schutzbedarf und den Bedrohungen geeignete Sicherheitsanforderungen abzuleiten.

Hier kann es ebenfalls dazu kommen, dass die vorhandenen Vorlagen nicht ausreichend sind und zusätzliche Sicherheitsanforderungen für ein Software-System bestimmt werden müssen. Ebenso ist der spezifische Kontext des zu entwickelnden Software-Systems zu berücksichtigen, wodurch die eingesetzten Vorlagen auf diesen Kontext angepasst werden müssen. In diesen Fällen fließen diese Ergebnisse ebenfalls in einer Rückkopplung in die Aufbereitungsphase zurück und es wird entschieden, ob die neuen und abgeänderten Sicherheitsanforderungen als Vorlagen aufbereitet und in den Vorlagenkatalog mit einfließen. Diese Zusammenhänge sind in Abbildung 26 dargestellt.

Aktivitäten zur Aufbereitung

Im Gegensatz zur traditionellen Analyse von Sicherheitsanforderungen, befasst sich die Aufbereitung von Sicherheitsanforderungen mit der Extraktion von gemeinsamen und optionalen Anforderungen aus bestehenden Software-Systemen. Zur Spezifikation von gemeinsamen Sicherheitsanforderungen

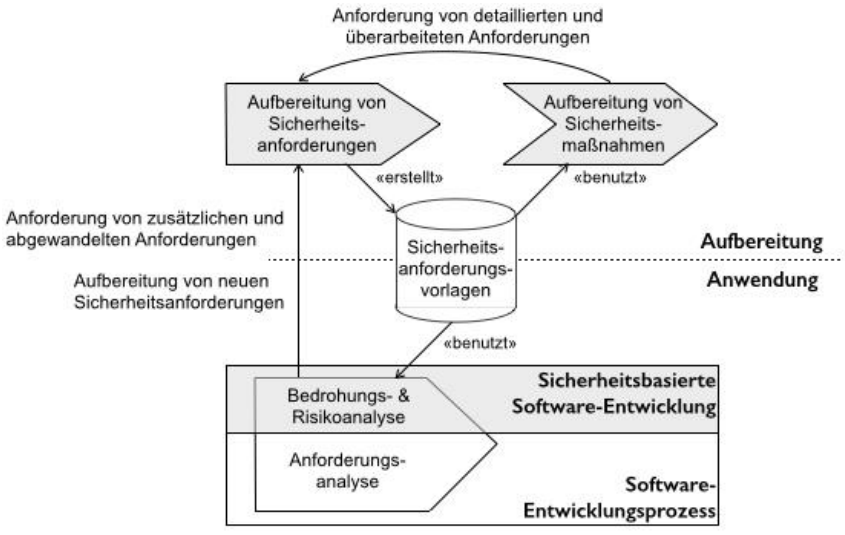

Abbildung 26: Kontext der Aufbereitung von Sicherheitsanforderungen

ist zunächst eine Identifikation einer initialen Menge an gemeinsamen Sicherheitsanforderungen aus verschiedenen Software-Systemen nötig. Anschließend werden diese in Form von Vorlagen aufbereitet und entsprechend dokumentiert. Durch die zuvor beschriebenen Rückkopplungen wird die Aufbereitung iterativ durchlaufen, um die Anforderungen zu präzisieren sowie die Qualität der Anforderungen zu erhöhen. Hierzu müssen die Anforderungen auf dem neuesten Stand gehalten werden. In dieser iterativen Vorgehensweise können gemeinsame Anforderungen auch zu optionalen Anforderungen werden. Dies ist zum Beispiel der Fall, wenn eine Mehrzahl der untersuchten Software-Systeme eine zuvor als verbindlich dokumentierte Anforderung als optional einstufen [PB+05:199f]. Der Ablauf der Analyse ist in Abbildung 27 dargestellt.

Zur Analyse von gemeinsamen Anforderungen existieren nach [PB+05:199f] mehrere Möglichkeiten,

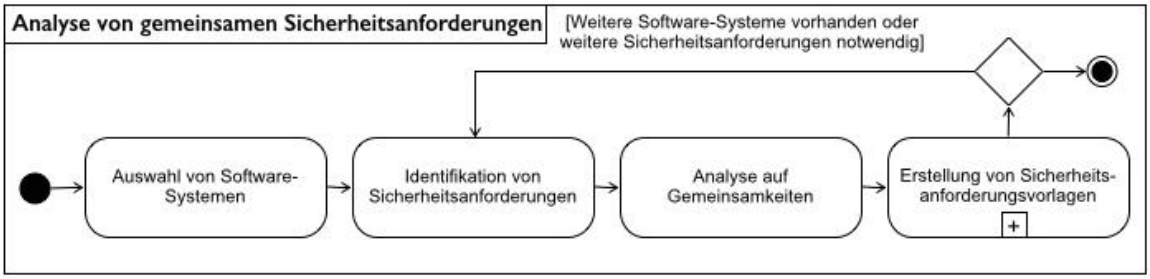

Abbildung 27: Basisaktivitäten zur Extraktion von Sicherheitsanforderungen

welche auch zusammen eingesetzt werden können. Diese Ansätze werden im Folgenden im Kontext der Aufbereitung von Sicherheitsanforderungen besprochen. Ausgangspunkt sind zunächst bereits existierende Software-Systeme, aus welchen die Anforderungen extrahiert werden. Hierbei kann es sich sowohl um Eigenentwicklungen als auch um Entwicklungen von Dritten handeln. Aus dieser Menge sollte zunächst eine begrenzte Anzahl an Software-Systemen ausgewählt werden, welche die fachliche Ausrichtung der Software-Systeme am geeignetsten widerspiegeln. Hierdurch lassen sich auch Sicherheitsanforderungen ableiten, die auf den fachlichen Kontext zugeschnitten sind.

Ein erster Ansatz zur initialen Erstellung und Analyse von Sicherheitsanforderungen ist die Abarbeitung von ein oder mehrere Prüflisten. Jeder Eintrag in einer Prüfliste stellt dabei eine Quelle bzw. eine Kategorie dar, aus welcher gemeinsame Sicherheitsanforderungen abgeleitet werden können. Beispielsweise lassen sich folgende Kategorien zur Erstellung und Analyse anwenden:

- **Sicherheitsanforderungen aus nationalen und internationalen Gesetzen und Standards:** Zahlreiche nationale und internationale Vorgaben zum Schutz von Informationen in Software-Systemen können zur Analyse von generell verpflichtenden Anforderungen herangezogen werden. Hierunter fallen zum Beispiel Vorgaben, welche durch das Grundschutzhandbuch des Bundesamtes für Sicherheit in der Informationstechnik (BSI) vorgeschlagen werden bzw. welche im Kriterienkatalog für Sicherheitsanforderungen des Common Criteria Recognition Arrangement (CCRA) spezifiziert sind.
- **Sicherheitsanforderungen aus Organisationsstandards:** Innerhalb einer Organisation, für welche Software-Systeme entwickelt werden, existieren Sicherheitsrichtlinien, welche den Schutz der organisationseigenen Ressourcen und Informationen beinhalten. Diese müssen bei der Aufbereitung der gemeinsamen Sicherheitsanforderungen berücksichtigt werden.

- **Ähnliche Sicherheitsanforderungen:** Bei der Untersuchung von verschiedenen Software-Systemen sollten die Sicherheitsanforderungen als verpflichtend analysiert werden, welche sich in ihrer Beschreibung und Spezifikation ähnlich sind.
- **Sich nicht widersprechende Sicherheitsanforderungen:** Ein weiteres Kriterium für gemeinsame Sicherheitsanforderungen sind die Anforderungen, welche nicht in Konflikt miteinander stehen und sich zum Beispiel nicht gegenseitig ausschließen.

Für die Untersuchung von verschiedenen Software-Systemen hinsichtlich gemeinsamer Sicherheitsanforderungen empfiehlt sich zudem der Einsatz einer Anforderungsmatrix. Hierbei werden die Sicherheitsanforderungen und die untersuchten Software-Systeme in die Spalten bzw. Zeilen eingetragen. In die Felder der Matrix wird anschließend vermerkt, welche Anforderung für welches Software-System eine verpflichtende Sicherheitsanforderung darstellt. Wird eine Sicherheitsanforderung in der Mehrzahl der Software-Systeme als verpflichtend angesehen, so ist dies ein Indiz zur Aufbereitung und Aufnahme der Sicherheitsanforderung in den Vorlagenkatalog.

Des Weiteren lassen sich auch die so identifizierten Sicherheitsanforderungen durch Interviews und Rücksprache mit den Kunden des Software-Systems priorisieren. Eine einfache Priorisierung teilt die Sicherheitsanforderungen dabei zum Beispiel in unnötige, grundlegende, wünschenswerte und wichtige Anforderungen ein, um eine niedrige, mittlere oder hohe Priorität auszudrücken [KS+96] in [PB+05:203]. Gemeinsame Sicherheitsanforderungen sollten dabei vorzüglich aus den Kategorien grundlegende Anforderungen und hoch priorisierten Anforderungen analysiert werden.

Abbildung 28 zeigt ein mögliches Zusammenspiel zwischen den Ansätzen. Mittels der verschiedenen Kategorien und der existierenden Software-Systeme wird eine initiale Menge an Sicherheitsanforderungen erstellt. Diese bilden zusammen mit den untersuchten Software-Systemen die Anforderungsmatrix. Anschließend werden die Einträge der Matrix bestimmt. Die so untersuchten Sicherheitsanforderungen werden im Anschluss mit dem Kunden besprochen und durch diesen priorisiert. Aus den so erzielten Ergebnissen werden die verpflichtenden Sicherheitsanforderungen bestimmt.

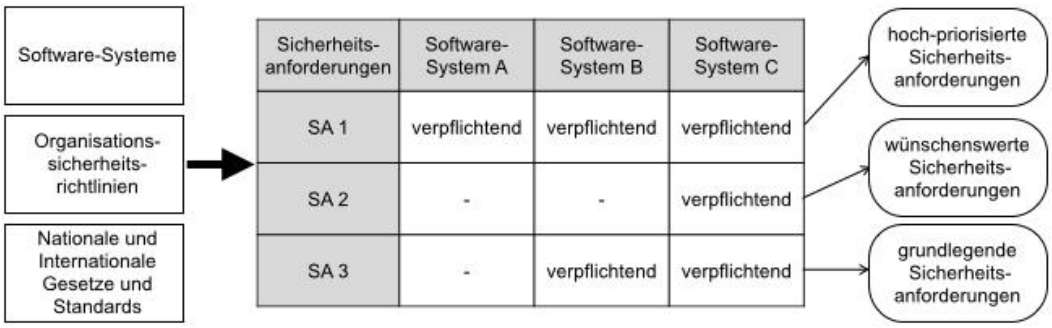

Abbildung 28: Verfahren zur Analyse von gemeinsamen Sicherheitsanforderungen

Wurden die verbindlichen Sicherheitsanforderungen mittels der besprochenen Verfahren extrahiert und dokumentiert, lassen sich aus den verbleibenden Elementen die optionalen Anforderungen auswählen. In der Anforderungsmatrix deuten Einträge auf optionale Anforderungen hin, wenn diese nur in einer kleinen Menge an Software-Systemen eingesetzt werden. Ebenso lassen sich stark abweichende Priorisierungen einer Anforderung durch verschiedene Kunden als optionale Anforderungen betrachten. Im Kontrast zu verpflichtenden Sicherheitsanforderungen müssen zusätzliche Informationen zu optionalen Sicherheitsanforderungen dokumentiert werden. Hierzu gehören zum einen die Vor- und Nachteile, die durch deren Einsatz erzielt werden, sowie generische

Informationen über den Verwendungszweck, wobei auch die widersprüchlichen Priorisierungen berücksichtigt werden können.

Nachdem die gemeinsamen bzw. optionalen Sicherheitsanforderungen identifiziert wurden, werden anschließend die Anforderungen in geeigneter Weise dokumentiert, so dass sie in einem zukünftigen Software-Entwicklungsprozess eingesetzt werden können. In dieser Arbeit werden zu diesem Zweck Sicherheitsanforderungsvorlagen eingesetzt, welche von den konkret zu schützenden fachlichen Ressourcen abstrahieren, jedoch die wesentlichen Informationen zu einer Sicherheitsanforderung beinhalten. Zu den grundlegendsten Informationen zählen dabei die mit der Sicherheitsanforderung verknüpften Schutzziele, der Schutzbedarf sowie die Bedrohungen und Angriffe. Entsprechend dieser Informationen werden die Schritte zur Spezifikation der Sicherheitsanforderungen, wie in Abbildung 29 dargestellt, unterteilt.

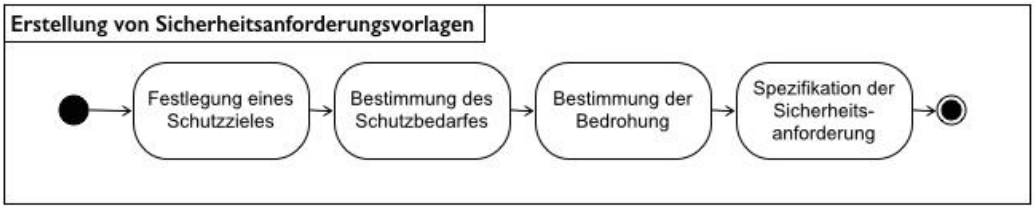

Abbildung 29: Aktivitäten zur Spezifikation von Sicherheitsanforderungsvorlagen

Zunächst werden die Schutzziele, welche durch die Sicherheitsanforderung umgesetzt werden, spezifiziert. Hierzu zählen die klassischen Ziele wie Diskretion, Integrität und Verfügbarkeit (engl. confidentiality, integrity, availability [Ec09:6ff]). Die Schutzziele geben einen ersten abstrakten Hinweis darauf, welchen Schutz eine Ressource benötigt. Des Weiteren wird der Schutzbedarf der Ressource abgeleitet und festgelegt. Dies ist ein zumeist qualitativer Wert, da ein quantifizierbarer Schaden an einer Ressource im Vorhinein schwer festzulegen ist [Ec09:176]. Neben dem Schutzbedarf wird auch die Bedrohung oder der konkrete Angriff spezifiziert, welcher zur Aufhebung des Schutzzieles führt und den Wert der Ressource mindert. Die anschließende Spezifikation der Sicherheitsanforderung beinhaltet Einschränkungen hinsichtlich der fachlichen Anforderungen, welche den Bedrohungen und Angriffen entgegenwirken und somit die Einhaltung der Schutzziele ermöglichen.

4.2.2 Spezifikation von wiederverwendbaren Sicherheitsmaßnahmen

Nach der Aufbereitung der Sicherheitsanforderungen setzt sich die darauf folgende Phase mit der Aufbereitung von wiederverwendbaren Sicherheitsmaßnahmenauseinander. Sicherheitsmuster haben sich als probate Methode zur Beschreibung von bewährten Lösungen zu bekannten Sicherheits-problemen durchgesetzt [SF+05], weshalb dieser Ansatz auch als Beschreibungsform in der vorliegenden Arbeit eingesetzt wird. Ebenso wie bei den Sicherheitsanforderungen ist hierbei ebenfalls ein Middle-Out-Vorgehen empfehlenswert, wobei dokumentierte Maßnahmen mit den in existierenden Sicherheitsprodukten umgesetzten Maßnahmen verglichen werden. Ziel der Aufbereitung ist es in diesem Fall, fachlichen Entwicklern ein methodisches Werkzeug zur Verfügung zu stellen, mit Hilfe dessen die strukturierte Ableitung von Sicherheitsmaßnahmen in einem konkreten Entwicklungsprozess ermöglicht wird.

Hierzu ist eine Kategorisierung anhand des Abstraktionsgrades sinnvoll, um zwischen formspezifischen bzw. plattformunabhängigen Maßnahmen unterscheiden zu können. Auf diese Weise lässt sich der Zusammenhang zwischen den Sicherheitsanforderungen und den gemeinsamen sowie

den optionalen Maßnahmen herstellen. Des Weiteren sollten die Zusammenhänge zwischen den Sicherheitsmaßnahmen untersucht und in der Musterbeschreibung vermerkt werden. Hierdurch lassen sich Abhängigkeiten identifizieren und sich ausschließende Verwendungen von Maßnahmen vermeiden.

Beziehungen zu anderen Aktivitäten

Die Erstellung von wiederverwendbaren Sicherheitsmaßnahmen stellt eine zentrale Aktivität in dem sicherheitsbasierten Entwicklungsvorgehen dar. Die Zusammenhänge bei der Erstellung sind in Abbildung 30 dargestellt. Wie bereits angesprochen, sind die zuvor in der Aufbereitungsphase erstellten Vorlagen für Sicherheitsanforderungen ein Ausgangspunkt für die Aufbereitung der Maßnahmen. Grund hierfür ist, dass Maßnahmen ausgewählt werden, welche die Sicherheitsanforderungen am geeignetsten umsetzen. Durch eine Verfeinerung der gewählten Maßnahmen werden detailliertere und notwendige Änderungen an Sicherheitsanforderungen nötig.

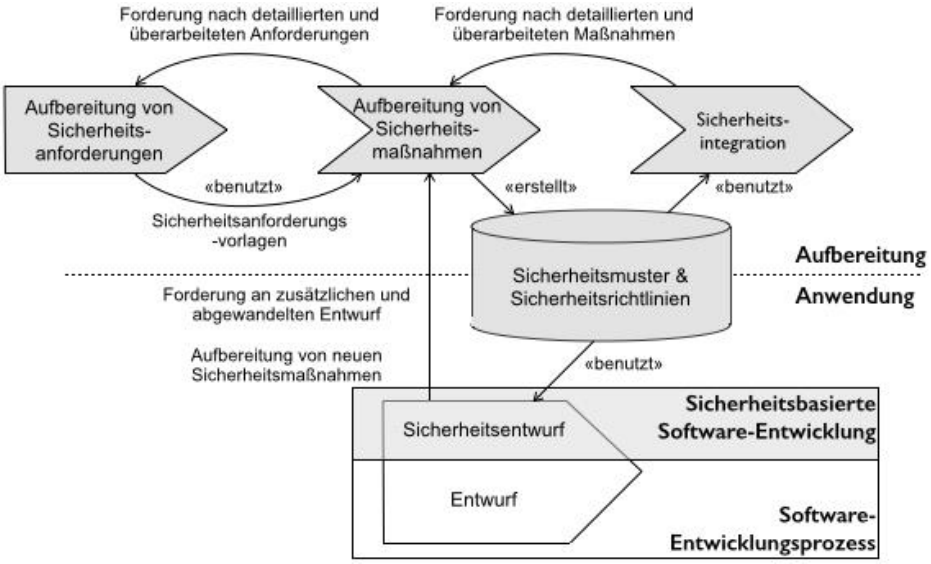

Abbildung 30: Kontext der Aufbereitung von Sicherheitsmaßnahmen

Bei der Verfeinerung der Sicherheitsmaßnahmen müssen dabei bestehende Sicherheitsprodukte berücksichtigt werden, da die Maßnahmen mittels der Produkte umsetzbar sein sollten. Sind spezifische Maßnahmen zwingend erforderlich, so ist die bestehende Sicherheitsinfrastruktur, durch beispielsweise Erweiterung bestehender oder Beschaffung neuer Sicherheitsprodukte, zu ergänzen, so dass die Sicherheitsmaßnahmen umgesetzt werden können. Die Abbildung der Sicherheitsmaßnahmen auf die in der Infrastruktur vorhandene Sicherheitsfunktionalität erfolgt dabei schrittweise. Ist keine direkte Abbildung einer Maßnahme auf eine existierende Sicherheitsmaßnahmemöglich, so ist eine detaillierte Überarbeitung und Verfeinerung der Maßnahmen in Betracht zu ziehen.

Neben der Abbildung von Maßnahmen auf Sicherheitsdienste und -produkte können diese gemäß dem zuvor besprochenen Referenzmodell als Ausgangspunkt zur Erhebung von gemeinsamen und variablen Sicherheitsmaßnahmen eingesetzt werden. Zu diesem Zweck werden die vorhandenen Sicherheitsprodukte auf die Umsetzung von bekannten Sicherheitsmaßnahmen untersucht. Dabei wird die auf eine Sicherheitsmaßnahme ausgerichtete Funktionalität eines Sicherheitsproduktes auf die konforme Umsetzung von bekannten Sicherheitsmustern bzw. Sicherheitsrichtlinien analysiert. Die Anzahl der zu untersuchenden Sicherheitsmuster und -richtlinien kann durch die fachliche Ausrichtung der Software-Systeme und der zuvor festgelegten Sicherheitsanforderungen eingeschränkt werden. Beispielsweise sollten Sicherheitsprodukte zur Umsetzung von Zugriffskontrollmaßnahmen

auf die Umsetzung von bekannten Modellen für Zugriffskontrollrichtlinien wie zum Beispiel rollenbasierte Zugriffskontrolle oder Zugriffskontrolllisten untersucht werden.

Die aufbereiteten Sicherheitsmuster und -richtlinien werden in der Entwurfsphase eines konkreten Software-Entwicklungsprozesses eingesetzt, um die spezifischen Sicherheitsmaßnahmen eines zu entwickelnden Software-Systems zu modellieren und an die existierenden Sicherheitsinfrastruktur anzupassen. Durch die Anpassung an den fachlichen Kontext müssen gegebenenfalls Maßnahmen abgeändert werden. Ebenso sind spezifische Sicherheitsmaßnahmen zu berücksichtigen, welche noch nicht aufbereitet wurden und Teil des Referenzmodells sind. In diesen Fällen werden die abgeänderten und zusätzlichen Maßnahmen in die Aufbereitung zurückgeführt und hinsichtlich ihrer Tauglichkeit zur zukünftigen Wiederverwendung in weiteren Software-Systemen analysiert.

Aktivitäten zur Aufbereitung von Sicherheitsmaßnahmen

Das Ziel der Aufbereitung von Sicherheitsmaßnahmen ist zum einen die Bereitstellung von wiederverwendbaren Lösungen zu bekannten Sicherheitsproblemen, so dass in einem Software-Entwicklungsprozess ein strukturierter Entwurf der benötigten Sicherheitsfunktionalität erfolgen kann. Zum anderen soll die durchgängige Abbildung von abstrakten Sicherheitsanforderungen auf konkrete Sicherheitsfunktionalität erfolgen. Daher besteht die wesentliche Aufgabe bei der Aufbereitung der Sicherheitsmaßnahmen in der Vermittlung zwischen den Sicherheitsanforderungsvorlagen und den Diensten der Sicherheitsinfrastruktur. Daher wird ein Middle-Out-Vorgehen eingesetzt, welches eine geeignete Abstraktion bzw. Verfeinerung der Sicherheitsmaßnahmen vornimmt. Die zugehörigen Aktivitäten der Aufbereitung sind in Abbildung 31 dargestellt.

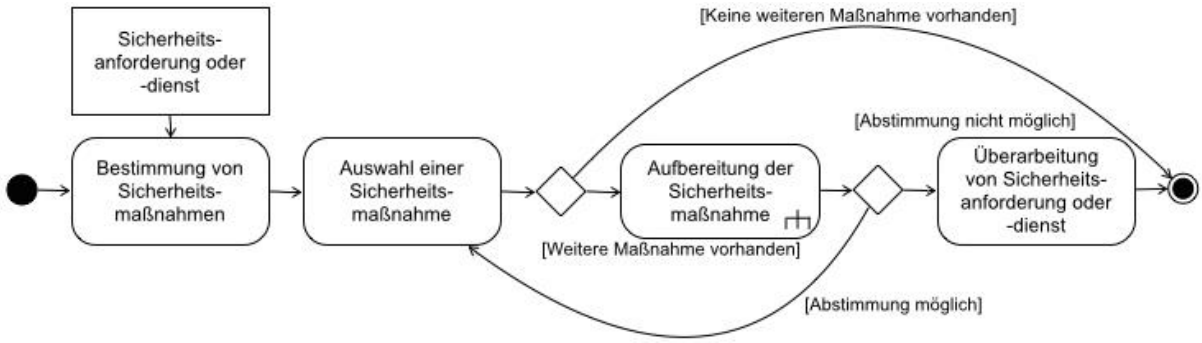

Abbildung 31: Basisaktivitäten zur Aufbereitung von Sicherheitsmaßnahmen

Das Vorgehen beginnt mit der Auswahl einer initialen Menge an Sicherheitsmaßnahmen, um entweder eine Sicherheitsanforderung umzusetzen oder um von dem technischen Kontext eines Sicherheitsdienstes zu abstrahieren. Im ersten Fall werden die initial ausgewählten Sicherheitsmaßnahmen im Allgemeinen sehr abstrakt sein, während sie im zweiten Fall zwar technologieunabhängig, aber trotzdem sehr konkret ausfallen. In der Regel wird eine Menge an Sicherheitsanforderungen ausgewählt, da eine Sicherheitsanforderung durch mehrere Maßnahmen umgesetzt wird sowie ein Sicherheitsprodukt mehrere Maßnahmen umsetzen kann. Die folgende Beschreibung des weiteren Vorgehens geht zur verständlicheren Darstellung von der Auswahl von Sicherheitsmaßnahmen zur Umsetzung einer Sicherheitsanforderung aus. Das Vorgehen lässt sich analog auf die initiale Auswahl von feingranularen Sicherheitsmaßnahmen übertragen.

Die ausgewählte Menge von Sicherheitsmaßnahmen wird nun sequentiell analysiert. Hierfür wird eine Maßnahme entweder zufällig oder nach einer Priorisierung geordnet ausgewählt. Die ausgewählte abstrakte Sicherheitsmaßnahme wird anschließend aufbereitet, in dem sie iterativ verfeinert wird, bis eine Abbildung auf eine technologienahe konkrete Sicherheitsmaßnahme möglich ist. Anschließend

wird diese aufbereitete Sicherheitsmaßnahme dokumentiert und die Aufbereitung für eine weitere Sicherheitsmaßnahme durchgeführt. Der Prozess endet, sobald die initiale Menge an Sicherheitsanforderungen abgearbeitet ist. Anschließend lässt sich der Prozess für eine weitere Sicherheitsanforderung durchführen. Führt die Aufbereitung zu keinem Ergebnis, so ist dies ein Indiz dafür, dass die zugehörige Sicherheitsanforderung detaillierter formuliert werden muss bzw. die vorhandene Sicherheitsinfrastruktur um Sicherheitsfunktionalität ergänzt werden sollte.

Bei der Aufbereitung einer einzelnen Sicherheitsmaßnahme wird zunächst überprüft, ob diese bereits aufbereitet wurde. Da eine Sicherheitsanforderung durch mehrere Sicherheitsmaßnahmen umgesetzt werden kann, können bei folgenden Iterationen der Abstimmung die bereits aufbereiteten Sicherheitsmaßnahmen eingesetzt werden, wodurch der Aufwand zur Aufbereitung reduziert wird. In diesem Fall wird die Sicherheitsmaßnahme als mögliche Umsetzung der Sicherheitsanforderung im Referenzmodell assoziiert und die nächste Sicherheitsmaßnahme ausgewählt. Ist die Sicherheitsmaßnahme noch nicht im Referenzmodell vorhanden, so wird sie in den weiteren Schritten aufbereitet.

Die Entscheidung über die ausgewählte Sicherheitsmaßnahme sowie der Einsatz der Sicherheitsmaßnahme werden dabei dokumentiert. Hierdurch wird zum einen die Nachvollziehbarkeit der Auswahl gewährleistet, wodurch auch in zukünftigen Iterationen die Eignung der ausgewählten Sicherheitsmaßnahme überprüft werden kann. Zum anderen ermöglicht dies die Validierung der Zweckmäßigkeit der Maßnahme beim Einsatz der Sicherheitsmaßnahme in einem konkreten Entwicklungsprozess. Diese Aktivitäten sind in Abbildung 32 dargestellt.

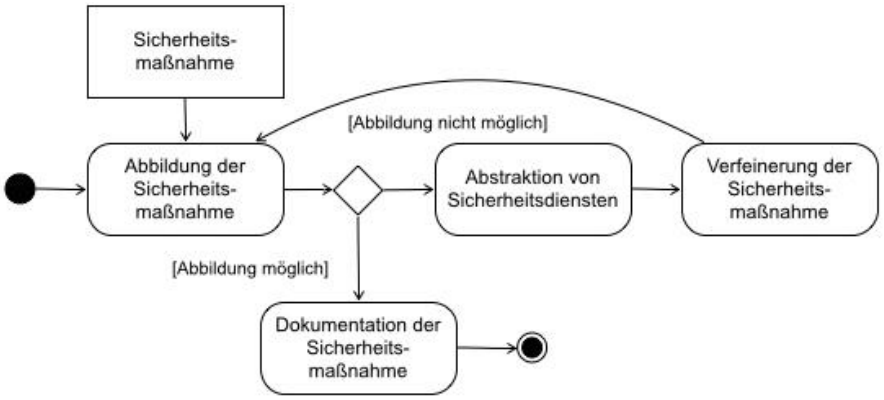

Abbildung 32: Aktivitäten zur Abstimmung von Sicherheitsmaßnahmen

Der Kern der Aufbereitung liegt in der Abstimmung von abstrakten und konkreten Sicherheitsmaßnahmen. Dieses Vorgehen wird insbesondere bei der initialen Erstellung eines Referenzmodells für Sicherheitsarchitekturen benötigt, da in diesem Fall noch keine Abstimmung erfolgt ist. Zur Abstimmung werden iterativ abstrakte Sicherheitsmaßnahmen spezialisiert sowie konkrete Sicherheitsmaßnahmen abstrahiert bis eine direkte Assoziation hergestellt werden kann.

Dabei wird zunächst überprüft, ob die gewählte Maßnahme durch einen oder mehrere Sicherheitsdienste direkt umgesetzt werden kann. Ist dies der Fall, so wird diese Abbildung ebenfalls dokumentiert und die nächste Sicherheitsmaßnahme wird ausgewählt. Ist dies nicht der Fall, wird zunächst eine initiale Auswahl an Sicherheitsdiensten erstellt, mittels welcher die abstrakte Sicherheitsmaßnahme umgesetzt werden kann. In diesem Zug wird dabei ebenfalls eine erste Abstraktion der Sicherheitsdienste auf feingranulare, jedoch technologieunabhängige Sicherheitsmaßnahmen erstellt. Dabei können abstrakte Sicherheitsmaßnahmen sehr viel Freiraum bei der Interpretation und Umsetzung beinhalten, wodurch ein direkte Abbildung auf die feingranularen

Sicherheitsmaßnahmen erschwert wird. In diesem Fall wird daher eine Verfeinerung der abstrakten Maßnahme vorgenommen, indem diese beispielsweise in mehrere spezialisiertere Sicherheitsmaßnahmen aufgeteilt wird.

Dokumentation der Sicherheitsmaßnahmen

Bei der Aufbereitung der Sicherheitsmaßnahmen werden die Entscheidungen über deren Auswahl sowie die Verfeinerung bzw. Abstraktion dokumentiert (vgl. Abbildung 33). Diese Dokumentation ist wesentliche Grundlage für die weitere Evolution der Sicherheitsmaßnahmen in dem Referenzmodell sowie für den Einsatz der Sicherheitsmaßnahmen in einem konkreten Software-Entwicklungsprozess.

Zur Einordnung von neuen Sicherheitsmaßnahmen in den Maßnahmenkatalog sind verschiedene Kategorisierungen notwendig. Zum einen muss das Abstraktionsniveau der Maßnahme festgelegt und zum anderen die Beziehungen zu weiteren Sicherheitsmaßnahmen verdeutlicht werden. Diese Beziehungen beinhalten die Spezialisierungs- bzw. Verfeinerungsbeziehung zu abstrakteren bzw. konkreteren Sicherheitsmaßnahmen. Ebenso können alternative Sicherheitsmaßnahmen zur Umsetzung eines Sicherheitsproblems dokumentiert werden, wodurch eine Variabilität im Referenzmodell geschaffen wird. Des Weiteren muss der Einsatzkontext der Sicherheitsmaßnahme sowie die Verwendung der Maßnahme zur Umsetzung einer Sicherheitslösung spezifiziert werden.

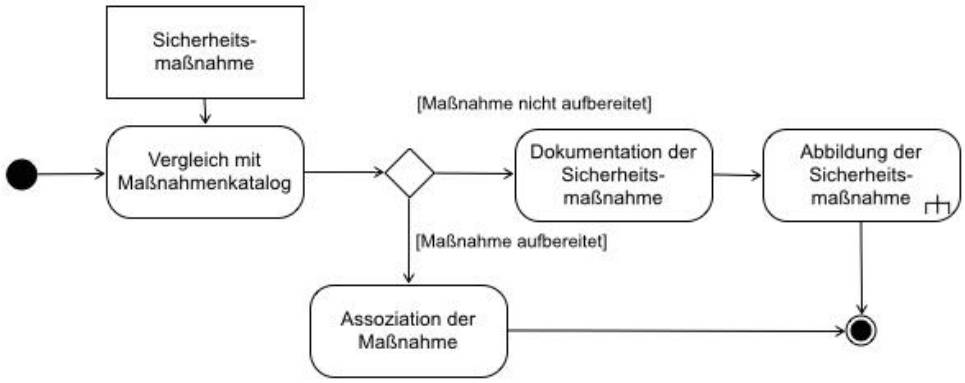

Abbildung 33: Aktivitäten zur Dokumentation von Sicherheitsmaßnahmen

Zur Beschreibung dieser Beziehungen wird in dieser Arbeit die Dokumentation und Spezifikation von Sicherheitsmaßnahmen in Form von Sicherheitsmustern [SF+05] durchgeführt. Aufgrund der konzeptionellen Nähe zum Entwurfsmusteransatz ermöglicht der Einsatz von Sicherheitsmustern eine Grundlage für die Integration der Sicherheitsmaßnahmen in den fachlichen Entwurf. Während Sicherheitsmuster eine Basis für die Dokumentation einer einzelnen Sicherheitsmaßnahme schaffen, sind sie weniger für die Spezifizierung von verschiedenen Beziehungen unter den Maßnahmen geeignet. Daher wird der Sicherheitsmusteransatz um ein Variabilitätsmodell erweitert, mittels welchem zum einen die Beziehungen zwischen verschiedenen Mustern, insbesondere der Spezialisierungsbeziehung zwischen verschiedenen Abstraktionsstufen spezifiziert werden kann. Zum anderen wird auch die Deklaration von alternativen Sicherheitsmaßnahmen ermöglicht.

Ausgehend von diesem Variabilitätsmodell führt die zuvor beschriebende Aufbereitung des Sicherheitswissens über Sicherheitsmaßnahmen zu einem Entscheidungsbaum, welcher die möglichen Sicherheitsmaßnahmen der Sicherheitsarchitektur im Referenzmodell widerspiegelt. Dabei sind abstrakte Sicherheitsmaßnahmen mit Sicherheitsanforderungen verknüpft, so dass eine Abbildung von den Anforderungsvorlagen auf die Sicherheitsmaßnahmen erfolgen kann. Eine abstrakte Maßnahme bildet dabei die Wurzel eines Entscheidungsbaumes, dessen Kindknoten die Spezialisierungen der Maßnahme darstellen. Eine Alternative in der Spezialisierung führt dabei zu einer Verzweigung

zwischen unterschiedlichen Kindknoten. Die so aufbereitete Dokumentation bietet fachlichen Software-Entwicklern eine Entscheidungsunterstützung bei der Auswahl und dem Einsatz der Sicherheitsmaßnahmen. Die grundlegenden Elemente des Variabilitätsmodells werden im Detail in Kapitel 6 vorgestellt.

4.3 Anwendung von Sicherheitswissen in der Software-Entwicklung

Die im vorherigen Abschnitt besprochene Aufbereitung von Sicherheitswissen in Form von wiederverwendbaren sicherheitsrelevanten Entwicklungsartefakten ist die Ausgangsbasis für die Entwicklung von Sicherheitsmaßnahmen in konkreten Software-Entwicklungsprozessen. Im folgendem Abschnitt wird die Entwicklung von Sicherheitsfunktionalität eines Software-Systems unter Einsatz von wiederverwendbarem Sicherheitswissen erläutert. Dabei wird von einem abstrakten Entwicklungsprozess ausgegangen, wodurch die Unabhängigkeit zu spezifischen Software-Entwicklungsprozessen gewährleistet wird. Da die Entwicklung von Sicherheitsfunktionalität abhängig von der fachlichen Funktionalität ist, wird insbesondere auf Voraussetzungen und Bedingungen eingegangen, welche zur Anwendung der wiederverwendbaren Sicherheitsartefakte notwendig sind. Diese Voraussetzungen sind dabei durch den Entwicklungsprozess zu gewährleisten, wodurch ggf. Abläufe eines Prozesses angepasst werden müssen. Dabei wird darauf geachtet, dass die auftretenden Auswirkungen auf den Prozess minimiert werden.

4.3.1 Analyse von Sicherheitsanforderungen

Die Anwendung der Vorlagen für Sicherheitsanforderungen erfordert zunächst die Identifizierung von schützenswerten Ressourcen in der fachlichen Anforderungsanalysephase. Anschließend werden auf jede identifizierte Ressource die Vorlagen angewendet, um abhängig von dem jeweiligen Schutzbedarf der Ressource die entsprechenden Sicherheitsanforderungen festzulegen. Die Vorlagen können dabei auch in Kombination mit bestehenden Verfahren zur Analyse von Sicherheitsanforderungen kombiniert werden.

Abstrakter Analyseprozess

Wie in Kapitel 2 besprochen existiert kein einheitlich akzeptiertes Vorgehen zur Erhebung und Analyse von Sicherheitsanforderungen. Daher wird in dieser Arbeit von einem abstrakten Analyseprozess für Sicherheitsanforderungen ausgegangen, welcher die dazu notwendigen grundlegenden Aktivitäten beinhaltet. Somit ist die Erhebung von Sicherheitsanforderungen mittels der zuvor angesprochenen Vorlagen auf konkrete Analyseprozesse anzupassen. Der im Folgenden vorausgesetzte Prozess ist in Abbildung 34 wiedergegeben.

Abbildung 34: Abstakter Analyseprozess für Sicherheitsanforderungen

Zunächst werden wie angesprochen, die zu schützenden Ressourcen der fachlichen Domäne identifiziert. Diese lassen sich zum Beispiel durch den Einsatz von Anwendungsfalldiagrammen und konzeptionellen Klassendiagrammen, welche die wesentlichen Domänenobjekte beinhalten, ableiten. Anschließend wird der Schutzbedarf der einzelnen Ressourcen bestimmt [Ec09:176ff]. Der Schutzbedarf kategorisiert die zu schützenden Ressourcen hinsichtlich ihres Wertes. Der Wert einer

Ressource wird dabei durch die Schadensauswirkungen bestimmt, welche im Falle einer Bedrohung verursacht werden. Mittels des Schutzbedarfes werden zudem auch die benötigten Schutzziele ermittelt, welche für die Ressource umgesetzt werden sollen.

Ausgehend von dem ermittelten Schutzbedarf und den Schutzzielen wird anschließend eine Bedrohungs- und Risikoanalyse durchgeführt. Hierbei werden die möglichen Bedrohungen untersucht, welche Schäden und Beeinträchtigungen an der Ressource vornehmen können. Ein durch eine Bedrohung verursachter Schaden an einer Ressource reduziert ihren Wert und umgeht die für die Ressource definierten Schutzziele. Das Ziel der Risikoanalyse ist die Bestimmung einer Wahrscheinlichkeit für das Auftreten von Bedrohungen. Ausgehend von diesen Werten werden im Abschluss Sicherheitsanforderungen spezifiziert, welche eine Bedrohung verhindern bzw. die Auswirkungen der Bedrohungen einschränken.

Einbeziehung von Sicherheitsanforderungsvorlagen

In dem zuvor vorgestellten Prozess werden die einzelnen Artefakte in jedem Entwicklungsprozess von Neuem bestimmt. Werden die Aktivitäten zudem von fachlichen Software-Entwicklern durchgeführt, welche nur geringen Hintergrund in der Sicherheitsdomäne besitzen, wird die Erhebung dieser Artefakte erschwert. Durch den Einsatz des aufbereiteten Sicherheitswissens in Form von Sicherheitsanforderungsvorlagen wird hierbei eine Unterstützung bei der Ableitung von Sicherheitsanforderungen ermöglicht.

Aufgrund der Struktur der Vorlagen, welche im Detail im Kapitel 5 besprochen wird, lassen sich in jedem der Schritte die Vorlagen nutzen, um die notwendigen Sicherheitsanforderungen in einem strukturierten Vorgehen zu bestimmen. Hierzu beinhalten die Vorlagen die wesentlichen Informationen zu Sicherheitsanforderungen, welche im abstrakten Analyseprozess bestimmt werden. Jedoch bieten die Vorlagen eine kontextunabhängige Darstellung der Informationen, in welcher die fachliche Ressource unberücksichtigt bleibt. Somit lassen sich in einer Vorlage konkrete Werte für einen Schutzbedarf, eine Bedrohung und die relevanten Schutzziele mit ein oder mehreren Sicherheitsanforderungen kombinieren. Aufgrund der Aufbereitung durch Sicherheitsexperten beruht diese Zusammenstellung dabei auf bewährten Kombinationen, wodurch durch die Vorlagen qualitativ hochwertige Anforderungen spezifiziert werden können.

Die Verwendung der Vorlagen basiert dabei auf einem Ausschlussverfahren. In jeder der zuvor beschriebenen Aktivitäten werden die ermittelten Werte mit den Vorlagen abgeglichen und die nicht zutreffenden Vorlagen zur Analyse verworfen. Gemäß des Analyseprozesses wird somit für eine ausgewählte schützenswerte Ressource der Schutzbedarf ermittelt und anschließend alle Vorlagen mit dem zutreffenden Schutzbedarf ausgewählt. Anschließend werden die ermittelten Schutzziele für die Ressource verwendet, um weitere Vorlagen zu filtern. Ebenso können die Vorlagen, welche anhand des Schutzbedarfes ausgewählt wurden, eingesetzt werden, um die Schutzziele für die Ressource zu bestimmen.

Die darauffolgende Bedrohungs- und Risikoanalyse wird ebenso durch die Vorlagen unterstützt. Einerseits können weitere Vorlagen durch die Betrachtung einer bestimmten Bedrohung ausgeschlossen werden. Andererseits können die möglichen Bedrohungen für eine Ressource anhand der verbleibenden Sicherheitsanforderungsvorlagen untersucht werden. Durch die Vorlagen wird eine Risikoanalyse unterstützt, da für die in den Vorlagen hinterlegten Bedrohungen bereits entsprechende Analysen durchgeführt wurden. Gegebenenfalls müssen diese noch an den fachlichen Kontext der Ressource angepasst werden, um genauere Angaben über Auswirkungen und Risiko der Bedrohung zu erhalten. Ausgehend von den ermittelten Werten lassen sich die zugehörigen Sicherheitsanforderungen aus den nach der Filterung verbliebenden Vorlagen ableiten.

Falls in einem der Schritte keine Vorlage auf die ermittelten Werte zutrifft, ist eine traditionelle Analyse durchzuführen. Das anschließende Ergebnis ist hinsichtlich der Aufnahme in den Katalog zu überprüfen und als wiederverwendbare Vorlage zu beschreiben. Dies ist dann praktikabel, wenn die Anforderung auch in weiteren Software-Systemen auftritt. Diese Einschätzung wird dann durch Sicherheitsexperten vorgenommen und ist Teil der Aufbereitungsphase.

4.3.2 Entwurf von Sicherheitsmaßnahmen

Die Integration von Sicherheitsmaßnahmen in den fachlichen Entwurf stellt eine Herausforderung für die Entwurfsphase dar, da eine Kombination von fachlichen und sicherheitsbezogenen Entwurfsmodellen zu einer Überladung des Modells führt. Die vorgestellten Sicherheitsmuster sollen eine Hilfestellung bei der strukturierten Ableitung und Spezifikation von Sicherheitsmaßnahmen geben.

Abstrakter Entwurfsprozess

Für die Entwicklung der Sicherheitsmaßnahmen wird in dieser Arbeit von einem spezifischen Entwicklungsprozessmodell abstrahiert, um die Abhängigkeit der Aktivitäten zum Einsatz der wiederverwendbaren Sicherheitsmuster zu ermöglichen. Demnach sollten die Aktivitäten des Entwicklungsvorgehens an spezifische Prozessmodelle angepasst werden. Unter Berücksichtigung von Sicherheitsmustern, welche mittels des in Kapitel 6 im Detail vorgestellten Variabilitätsmodells aufbereitet und dokumentiert werden, wird jedoch von einer iterativen Verfeinerung der Entwurfsartefakte ausgegangen. Dies stellt in der Regel keine Einschränkung dar, sondern entspricht der gängigen Praxis der Zerlegung von komplexen Software-Systemen.

Ausgangspunkt ist die Abbildung von Sicherheitsanforderungen auf abstrakte Sicherheitsmaßnahmen. In einem fachlichen Entwicklungsprozess führt dies zur Identifizierung von abstrakten fachlichen Komponenten, welche in ihrem Zusammenspiel die Architektur des Software-Systems darstellen [BD10]. Analog werden ausgehend von den beschriebenen Sicherheitsanforderungen sicherheitsbezogene Komponenten identifiziert, welche die Sicherheitsarchitektur bilden. Hierbei werden mittels Architekturstilen, wie beispielsweise Peer-to-Peer oder 3-Schichten-Architektur [BD10:277ff], robuste Vorgaben für den Aufbau der Architektur gemacht. Auf diesem Abstraktionsniveau lassen sich diese Architekturstile ebenfalls für die Strukturierung der Sicherheitsarchitektur einsetzen.

Im Weiteren wird die interne Struktur der sicherheitsbezogenen Komponenten entwickelt. Im Gegensatz zur Architekturbeschreibung, wird hierdurch die interne Funktionsweise der Komponenten festgelegt. Die Zerlegung erfolgt dabei nach dem Kriterium, eine möglichst lose Kopplung zwischen den Komponenten und eine hohe Kohäsion innerhalb der Komponenten zu erreichen. Auf diese Weise lassen sich gegebenenfalls auch die abstrakten Komponenten zunächst rekursiv in weitere interne Komponenten aufteilen. In einem objektorientierten Ansatz wird anschließend die interne Funktionalität einer Komponente in einzelne Objekte gekapselt. Hierdurch wird ein Feinentwurf vorgenommen, mittels welchem eine direkte Abbildung auf objektorientierte Programmiersprachen ermöglicht wird. Hierdurch wird die Implementierung von Adaptern zur Nutzung von existierenden Sicherheitsdiensten oder -produkten ermöglicht.

Einbeziehung von Sicherheitsmustern

Unter Einsatz des in dem Referenzmodell einer Sicherheitsarchitektur aufbereiteten Sicherheitswissens zur Modellierung von Sicherheitsmaßnahmen werden die Aktivitäten des Entwurfsprozesses unterstützt. Zunächst wird durch das aufbereitete Sicherheitswissen die Abbildung von Anforderungen auf Sicherheitsmaßnahmen im Referenzmodell vorgenommen. Hierbei werden zunächst sogenannte Architekturtaktiken eingesetzt, welche noch keine konkreten Lösungen vorgeben, sondern die generel-

le Lösungsweise zur Umsetzung der Anforderung beschreiben. Dies ist ein zusätzlicher Schritt vor der Definition der Sicherheitsarchitektur, um deren Ausrichtung genauer zu spezifizieren. Es werden dabei explizit Sicherheitstaktiken vorgegeben, welche ausgehend von dem Schutzbedarf und dem Wert der zu schützenden Ressource ausgewählt werden können. Anschließend werden Sicherheitsmaßnahmen ausgewählt, welche zu den ausgewählten Taktiken gehören und die Sicherheitsanforderungen umsetzen. Diese Sicherheitsmaßnahmen beschreiben Lösungen auf Architekturebene und bilden somit die Grundlage der Sicherheitsarchitektur.

Die anschließende Verfeinerung der Sicherheitsmaßnahmen lässt sich durch das aufbereitete Sicherheitswissen mittels des im Kapitel 6 beschriebenen Variabilitätsmodells für Sicherheitsmuster strukturiert durchführen. Die ist möglich, da die Assoziationen zwischen den verschiedenen Mustern im Modell dokumentiert sind. Somit ist bei der Verfeinerung lediglich die Auswahl von benötigten Sicherheitsmaßnahmen anhand der im Referenzmodell vorhandenen Artefakte notwendig. Hierbei werden die Sicherheitsmaßnahmen durch das Variabilitätsmodell gemäß ihres Abstraktionsgrades eingeteilt, so dass der Verfeinerungsprozess unterstützt wird. Da die Sicherheitsmaßnahmen mit existierenden Sicherheitsprodukten und -rahmenwerken abgestimmt sind, wird nach einer Verfeinerung die Abbildung auf diese Technologien unterstützt. Hierdurch reduziert sich zum einen der erforderliche Implementierungsaufwand und zum anderen lassen sich die so entwickelten Software-Systeme in eine bestehende IT-Infrastruktur integrieren.

Da die einzelnen Sicherheitsmaßnahmen als Sicherheitsmuster beschrieben sind, stellen diese nicht die gesamte Sicherheitslösung sondern lediglich einzelne Teile bereit. Daher müssen bei der iterativen Verfeinerung zudem die Assoziationen zwischen verschiedenen Sicherheitsmustern berücksichtigt werden. Bei jedem Iterationsschritt steigt dabei die Anzahl der zu kombinierenden Sicherheitsmuster, um eine Gesamtlösung zu entwickeln.

Die Anpassung des Vorgehens an ein spezifisches Prozessmodell lässt sich durch die Anzahl der Verfeinerungsschritte durchführen. Dies lässt sich aus der Beobachtung schließen, dass verschiedene fachliche Entwicklungsprozesse unterschiedliche Vorgehensweisen zur Verfeinerung einer fachlichen Software-Architektur einsetzen. So kann bei einem agilen Vorgehen die Anzahl so reduziert werden, dass eine direkte Ableitung von Sicherheitsanforderungen auf technische Sicherheitsmaßnahmen ermöglicht wird. Dies setzt jedoch eine sehr spezifische Beschreibung des Referenzmodells der Sicherheitsarchitektur voraus.

4.4 Resümee

In diesem Kapitel wurde ein Vorgehen zur Entwicklung von sicheren Software-Systemen vorgestellt. Das Ziel der Methode ist zum einen die Aufbereitung von bestehendem Sicherheitswissen in eine wiederverwendbare Form und zum anderen dessen Einsatz in einem fachlichen Entwicklungsprozess zur strukturierten Entwicklung von Sicherheitsfunktionalität. Die Wiederverwendung wird ergänzt um eine Durchgängigkeit und Kontinuität der Aktivitäten und Ergebnisartefakte des Vorgehens, wodurch eine Entscheidungsfindung bei der Auswahl von Sicherheitsfunktionalität unterstützt und die Nachvollziehbarkeit der Entwicklungsentscheidungen ermöglicht wird. Entsprechend dieser Ziele ist die Methode in zwei konzeptionelle Phasen, einer Aufbereitungsphase und einer Anwendungsphase, untergliedert, welche sich gegenseitig beeinflussen.

In der Aufbereitungsphase wird das existierende Sicherheitswissen für den Einsatz in einem Software-Entwicklungsprozess vorbereitet. Als Grundlage für die Aufbereitung wurde ein sichtenbasiertes Referenzmodell für Sicherheitsarchitekturen vorgestellt, um eine schrittweise Ableitung von Sicherheitswissen aus existierenden Sicherheitsprodukten und -rahmenwerken durchzuführen. Dabei wurde

ein Middle-Out-Vorgehen eingesetzt, welches dokumentierte und bewährte Methoden für Sicherheitsmodelle mit den in einer vorhandenen Sicherheitsarchitektur vorhandenen Funktionen abgleicht und eine Verknüpfung herstellt.

Als Ergebnisse dieser Aufbereitung wurden Sicherheitsanforderungsvorlagen sowie Sicherheitsmuster vorgestellt. Die Vorlagen dienen dabei zur Spezifizierung der Zusammenhänge zwischen bekannten Bedrohungen und Schutzzielen einer schützenswerten Ressource sowie den entsprechenden Sicherheitsanforderungen. Die Anforderungen stehen dabei in Verbindung zu abstrakten Sicherheitsmaßnahmen, welche Ausgangspunkt für die in der Entwurfsphase eingesetzten Sicherheitsmuster sind und eine Durchgängigkeit zwischen der Analysephase und der Entwurfsphase zulassen. Mittels der Sicherheitsmuster wird der strukturierte Entwurf von Sicherheitsmaßnahmen ermöglicht, wobei der hierarchische Zusammenhang zwischen abstrakten Sicherheitsmaßnahmen und konkreten technischen Implementierungen dokumentiert wird.

Die Anwendungsphase bezieht sich auf den Einsatz des aufbereiteten Sicherheitswissens in einem Software-Entwicklungsprozess. Die dokumentierten sicherheitsrelevanten Entwicklungsartefakte der Aufbereitungsphase werden instanziiert und an den jeweiligen Kontext der Entwicklung angepasst. In der Analysephase eines Entwicklungsprozesses werden hierfür zunächst die Sicherheitsanforderungsvorlagen auf die identifizierten fachlichen Ressourcen angewendet, um durch ein Ausschlussverfahren die passenden Vorlagen zu bestimmen und entsprechende Sicherheitsanforderungen zu spezifizieren. Da die Sicherheitsanforderungen aus dem aufbereiteten Sicherheitswissen stammen, spiegeln sie bewährte Anforderungen wieder, welche auch in der Sicherheitsarchitektur umgesetzt sind.

Durch die in der Aufbereitung gewährleistete Durchgängigkeit der Entwicklungsartefakte wird die Abbildung der Anforderungen auf abstrakte Sicherheitsmaßnahmen vorgenommen, welche in der sich anschließenden Entwurfsphase des zu entwickelnden Software-Systems iterativ verfeinert werden. Die Verfeinerungsstufe orientiert sich dabei an der Abstraktionsstufe der fachlichen Modelle. Die spezifizierten Auswahlmöglichkeiten zwischen mehreren alternativen Entwurfslösungen werden dabei selektiert, wodurch die Sicherheitsfunktionalität festgelegt und die Integration in die fachliche Funktionalität definiert wird. Durch die zuvor festlegte Verfeinerung bis zu einer technologischen Sicherheitsplattform wird der Übergang zu bestehenden Sicherheitsprodukten erleichtert.

Mit dem vorgestellten sicherheitsbasierten Entwicklungsvorgehen lässt sich eine effiziente Entwicklung von Sicherheitsfunktionalität erreichen. Sicherheit als qualitative Eigenschaft eines Software-Systems wird während der Anforderungsanalyse und der Entwurfsphase parallel zur fachlichen Funktionalität durchgängig betrachtet, wodurch eine auf die Sicherheitsbedürfnisse angepasste Implementierung von Sicherheitsfunktionalität erreicht wird. Die Berücksichtigung von bewährten Sicherheitsmethoden führt zu einer effizienten Analyse von Sicherheitsanforderungen und ermöglicht strukturierte Entscheidungen beim Entwurf von Sicherheitsmaßnahmen. Die Verwendung von vorhandener Sicherheitsfunktionalität in Form von stabilen und unter Einsatz getesteten Produkten und Rahmenwerken vermeidet fehleranfällige Eigenimplementierungen. Die Trennung in eine Aufbereitungsphase und eine Anwendungsphase fördert zudem die Trennung von Zuständigkeiten zwischen Sicherheitsexperten und fachlichen Software-Entwicklern.

Einem Software-Entwickler wird mit der vorgestellten Entwicklungsmethode ein unterstützendes Werkzeug zur Verfügung gestellt, um die Sicherheitsfunktionalität eines Software-Systems systematisch entwickeln zu können. Das vorgestellte Entwicklungsvorgehen fokussiert in der hier vorgestellten Übersichtsform lediglich die auszuführenden Aufgaben von Sicherheitsexperten und Software-Entwicklern. Auf die Struktur der in der Aufbereitungsphase entstehenden Entwicklungsartefakte und deren konkreter Einsatz in der Anwendungsphase wurde nicht näher eingegangen. Die Details der

Vorlage zur Analyse von Sicherheitsanforderungen werden im folgenden Kapitel 5 erläutert. Im Kapitel 6 wird mit einem Variabilitätsmodell für Sicherheitsmuster die Beschreibung von Sicherheitsmaßnahmen vorgestellt.

5 Vorlagen für Sicherheitsanforderungen

In der sicherheitsbasierten Software-Entwicklung ist die Analyse und die Spezifikation von Sicherheitsanforderungen grundlegender Ausgangspunkt für den sich anschließenden Entwurf und die Implementierung von Sicherheitsmaßnahmen. Ziel dabei ist es, den Schutzbedarf eines Software-Systems möglichst präzise zu beschreiben, um so die Implementierung von unzureichenden oder unpassenden Sicherheitsmaßnahmen zu vermeiden. Wie im vorangegangenen Kapitel beschrieben, ist die Erhebung und die Analyse von Sicherheitsanforderungen jedoch ein aufwendiges Verfahren. Insbesondere fachliche Entwickler fällt diese Analyse schwer, da sie eine andere Herangehensweise erfordert. Somit wird für diese Aktivität eine Unterstützung benötigt, welche eine Grundlage für die Sicherheitsanforderungsanalyse bietet. Diese Arbeit fokussiert dabei die Unterstützung durch die Aufbereitung und Verwendung von wiederverwendbarem Sicherheitswissen.

Zu diesem Zweck wird im folgenden Kapitel ein Ansatz zur Spezifikation von wiederverwendbaren Sicherheitsanforderungen vorgestellt. Kern des Ansatzes sind dabei Vorlagen für Sicherheitsanforderungen, in welchen die wesentlichen Artefakte der Anforderungsanalyse bereits definiert sind. Die Instanziierung und Anpassung der Vorlagen im Kontext der Entwicklung eines konkreten Software-Systems liefert eine Grundlage für die Spezifikation von Sicherheitsanforderungen. Diese Vorlagen beruhen dabei auf bewährten Sicherheitsanforderungen, welche bereits eingesetzt und als praktikabel eingestuft wurden. Allgemein sind die Sicherheitsanforderungsvorlagen unabhängig von einem konkreten Entwicklungsprozessmodell verwendbar. In dieser Arbeit wird jedoch der Einsatz der Vorlagen in Verbindung mit dem im Kapitel 4 vorgestellten Entwicklungsvorgehen behandelt.

Zunächst wird die Struktur der Vorlagen durch ein in Kapitel 5.1 beschriebenes Domänenmodell festgelegt. Dabei werden die wesentlichen Konzepte und deren Beziehungen untereinander definiert. Diese Festlegungen sind essentiell für den Aufbau und die Verwendung der Sicherheitsanforderungsvorlagen, da in dieser Domäne viele Unklarheiten bezüglich der verwendeten Begrifflichkeiten bestehen. Das Domänenmodell ist modular aufgebaut, so dass die Vorlagen unterschiedlich ausführlich beschrieben werden können. Hierdurch werden sowohl unterschiedliche Analyseverfahren unterstützt als auch das Domänenmodell erweiterbar gestaltet. Zudem kann bei der Festlegung der Vorlagen zwischen generischen, d.h. von einer Fachdomäne unabhängigen, sowie von domänenspezifischen Sicherheitsanforderungen unterschieden werden. Im Kapitel 5.2 wird diese Kategorisierung aufgegriffen, um konkrete Beispiele für Sicherheitsanforderungsvorlagen in einem Vorlagenkatalog vorzustellen.

5.1 Beschreibung des Domänenmodells

Das Ziel der Sicherheitsanforderungsvorlagen ist die Unterstützung einer Bedrohungs- und Risikoanalyse für ein Software-System durch den Einsatz von bewährten Sicherheitsanforderungen. In einem solchen Analyseprozess werden weitere Zwischenartefakte, wie beispielsweise Schutzbedarf und Schutzziele spezifiziert. Um eine möglichst hohe Wiederverwendung der Vorlagen zu erreichen, ist es notwendig, diese wesentlichen Konzepte und deren Beziehungen untereinander eindeutig festzulegen. Die Struktur der Vorlagen wird daher durch das im Folgenden definierte Domänenmodell festgelegt, welches die wesentlichen, zur Spezifikation der Sicherheitsanforderungsvorlagen notwendigen Informationsartefakte beschreibt.

Um die Integrationsfähigkeit der Vorlagen in unterschiedliche Entwicklungsprozesse zu gewährleisten, ist es notwendig, das Domänenmodell modularisiert und anpassungsfähig zu gestalten. Diese mit der Entwicklung des Domänenmodells verbundenen Entwurfsprinzipien und -entscheidungen werden im Abschnitt 5.1.1 präsentiert. Hierbei werden unter anderem auch die in Kapitel 3 vorgestellten verwandten Ansätze berücksichtigt. Das Domänenmodell als Resultat des Erstellungsprozesses wird anschließend in Abschnitt 5.1.2 erörtert.

5.1.1 Grundlegende Struktur

Das Ziel der Sicherheitsanforderungsvorlagen ist deren flexibler und vielfältiger Einsatz. So soll die Beschreibung von generischen als auch fachspezifischen Sicherheitsproblemen und -anforderungen unterstützt werden. Ebenso sollen leichtgewichtige als auch detaillierte Beschreibungen der Sicherheitsanforderungen möglich sein, um verschiedene Software-Entwicklungsprozesse zu unterstützen. Um dies zu ermöglichen, muss auch das zugrundeliegende Domänenmodell der Vorlagen flexibel gestaltet werden. Im Folgenden werden daher die Entwurfsziele des Domänenmodells vorgestellt, woraus dessen grundlegende Struktur abgeleitet wird.

Entwurfsziele

Die Domäne der Sicherheitsanforderungsanalyse lässt sich in verschiedene Subdomänen einteilen, welche den Aktivitäten in der Erhebung von Sicherheitsanforderungen entsprechen. Eine grundlegende Aktivität besteht in der Identifikation von schützenswerten Ressourcen. Des Weiteren werden in einer Bedrohungs- und Risikoanalyse die Gefahren bestimmt, welche zu Schäden an schützenswerten Ressourcen führen. Ebenso stellt die Spezifikation von Schutzzielen und Sicherheitsanforderungen eine weitere essentielle Aktivität dar. Unterschiedliche Vorgehensmodelle für die Erhebung von Sicherheitsanforderungen behandeln diese Aktivitäten auf verschiedene Art, indem zum Beispiel die Reihenfolge der Aktivitäten geändert wird.

Modularisierung: Um die Integrationsfähigkeit der Vorlagen in verschiedene Vorgehensmodelle zu gewährleisten, werden diese Bereiche durch lose gekoppelte Module innerhalb des Domänenmodells repräsentiert. In jedem Modul werden die kohärenten Konzepte der jeweiligen Subdomäne spezifiziert. Die Zusammenhänge zwischen zwei Modulen wird durch die Verbindung von geeigneten Konzepten aus den beteiligten Modulen hergestellt. Um die Unabhängigkeit der Module zu gewährleisten, werden dabei die Verbindungen möglichst reduziert. Die Modularisierung des Domänenmodells ermöglicht unter anderem die unabhängige Weiterentwicklung der einzelnen Bereiche.

Das Domänenmodell wird in Basismodule eingeteilt, welche die wesentlichen Teilbereich des Analyseprozesses widerspiegeln. Das Ressourcenmodul beschreibt die Eigenschaften von schützenswerten Ressourcen und Gütern. Das Bedrohungsmodul definiert die für die Beschreibung von Bedrohungen und Angriffen notwendigen Konzepte. Das Schutzmodul beinhaltet die Konzepte, welche für die Spezifikation von Sicherheitsanforderungen notwendig sind. Falls möglich oder notwendig, werden die Module in weitere Untermodule aufgeteilt. Zudem werden Stellen aufgezeigt, an denen eine Erweiterung der vorhandenen Module um weitere Konzepte möglich oder erwünscht ist. Hierdurch wird die unabhängige Entwicklung sowie der Austausch von einzelnen Modulen ermöglicht.

Anpassungsfähigkeit: Die in dieser Arbeit vorgestellten Konzepte für das Domänenmodell für Sicherheitsanforderungen spiegeln den derzeitigen Stand der Forschung und Technik wider. Ziel dabei ist es, ein möglichst einheitliches und generell anerkanntes Modell für die Analyse von Sicherheitsanforderungen zu schaffen. Hierdurch kann es auch als Grundlage für weitere Arbeiten dienen in existierende Ansätze angewendet werden. Jedoch gibt es in diesem Bereich selten eine

einheitliche Definition für die benötigten Konzepte und Begriffe. Wird das Werkzeug der Sicherheitsanforderungsvorlagen in anderen Ansätzen verwendet, kann daher davon ausgegangen werden, dass Anpassungen und Änderungen am Domänenmodell vorgenommen werden, um den jeweiligen spezifischen Anforderungen und Bedürfnissen der Ansätze gerecht zu werden.

Erweiterbarkeit: So bedarf es beispielsweise in einem agilen Entwicklungsvorgehen, wie beispielsweise Scrum [SK13], meist nur einer leichtgewichtigen und somit auf das notwendigste reduzierte Beschreibung von Bedrohungen, um passende Sicherheitsanforderungen zu spezifizieren. Dagegen wird in einem traditionellen und iterativen Vorgehen, wie zum Beispiel dem Rational Unified Process (RUP [So07]), meist eine detaillierte Analyse durchgeführt, in welcher die komplexen Zusammenhänge zwischen Bedrohungen, Risiko, Schutzziele, etc. genauer spezifiziert werden müssen. Zudem kann ebenfalls davon ausgegangen werden, dass sich in Zukunft die Domäne der Sicherheitsanforderungsanalyse weiterentwickeln wird. Daher ist das vorgestellte Domänenmodell so konzipiert, dass es durch weitere Begriffe ausgebaut und erweitert werden kann.

Meta-Metamodell

Da das Domänenmodell die wesentlichen Konzepte der Domäne der Sicherheitsanforderungsanalyse bereitstellt, entspricht es einem Metamodell gemäß des Paradigmas der modellgetriebenen Software-Entwicklung [SV+07]. Demnach ist für die formale Spezifikation des Domänenmodells ein zugrundeliegendes Meta-Metamodell notwendig, welches notwendige Sprachelemente für ein Meta- bzw. Domänenmodell bereitstellt. In dieser Arbeit wird zu diesem Zweck die Meta Object Facility (MOF [OMG-MOF-v2.4]) eingesetzt, welches ebenfalls das Meta-Metamodell der UML darstellt. Konkret wird dabei eine reduzierte Menge des MOF, der sogenannte Essential-Meta-Object-Facility-Standard (EMOF) eingesetzt, welcher eine Implementierung mittels des quelloffenen Projektes Ecore erfährt.

Bei der Modellierung des Domänenmodells werden dabei nicht alle zur Verfügung stehenden Modellierungselemente eingesetzt. Die einzelnen Konzepte werden mittels der Ecore-Klasse „EClass“ modelliert und als rechteckige Kästen dargestellt. Die einzelnen Module werden durch die Ecore-Klasse „EPackage“ repräsentiert, da durch deren Unterstützung von Sammlungen von „EClass“- und internen „EPackage“-Modellierungselementen die Modularisierung der Konzepte modelliert werden kann. In der Darstellung werden dabei die Module als abgerundete Rechtecke dargestellt. Zuletzt wird die Ecore-Klasse „EReference“ zur Modellierung der Assoziationen zwischen den Konzepten eingesetzt und als gerichtete oder ungerichtete Linien dargestellt.

Kernzusammenhänge

In den Modulen des Domänenmodells werden Konzepte definiert, welche die zentralen Begrifflichkeiten der jeweiligen Subdomänen darstellen. Innerhalb des Ressourcenmoduls ist dabei das Konzept der zu schützenden Ressourcen (engl. resource bzw. asset) essentieller Ausgangspunkt für die Beschreibung von fachlichen Ressourcen. Ressourcen sollen vor Bedrohungen geschützt werden, da diese Schäden an den Ressourcen verursachen und somit ihren Wert vermindern. Bedrohungen stellen daher das zentrale Konzept des Bedrohungsmoduls dar. Schutzziele definieren Eigenschaften der Ressourcen, welche eingehalten werden müssen, um den Wert der Ressource zu erhalten. Bedrohungen zielen darauf ab, die Schutzziele einer Ressource zu unterlaufen. Schutzziele sind somit zentrales Konzept des Schutzmodules. Abbildung 35 zeigt den Zusammenhang zwischen den Modulen und den Kernelementen.

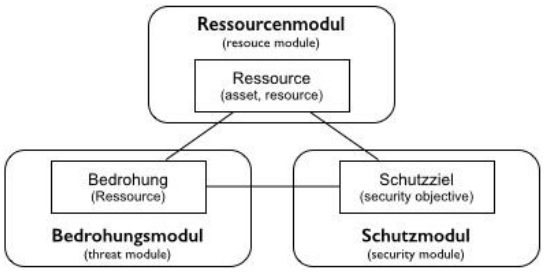

Abbildung 35: Module des Domänenmodells

Diese zentralen Konzepte sind über die Modulgrenzen hinweg verbunden und stellen somit den minimalen Kern des Domänenmodells dar, welcher zur Beschreibung von Sicherheitsanforderungen notwendig ist. Darüber hinaus stellen die Assoziationen zwischen den Konzepten die Hauptverknüpfungen zwischen den Modulen des Domänenmodells dar. Innerhalb der Module werden diese Konzepte jeweils um zusätzliche Konzepte erweitert, wodurch detailliertere Beschreibungen der Sicherheitsvorlagen ermöglicht werden. Dabei wird ebenfalls eine interne Strukturierung der Module durch Sub-Module vorgenommen. Ziel hierbei ist es, auch für die interne Struktur eine lose Kopplung zu erreichen, so dass die Sub-Module ebenfalls weiterentwickelt bzw. durch existierende Domänenmodelle ausgetauscht werden können. Ebenso wird durch die interne Modularisierung die Komplexität der einzelnen Sub-Module reduziert.

5.1.2 Ressourcenmodul

Die Spezifikation von Sicherheitsanforderungen bezieht sich auf die zu schützenden Ressourcen einer fachlichen Domäne. Im Ressourcenmodul werden die Konzepte zusammengefasst, welche für die Assoziation zwischen sicherheitsrelevanten Konzepten und fachlichen Ressourcen notwendig sind.

Wie bereits dargestellt, ist das zentrale Element des Ressourcenmoduls das Ressource-Element. Es stellt eine Abstraktion von den zu schützenden Ressourcen der fachlichen Domäne dar. Durch diese Abstraktion lassen sich beliebige Fachdomänen mit dem Domänenmodell verknüpfen, wodurch das Ressourcen-Element eine Form von Platzhalter (engl. proxy) darstellt. Existiert ein Domänenmodell für eine fachliche Domäne, so lassen sich die beiden Domänenmodelle durch den Einsatz des Ressource-Elementes als Oberklasse der fachlichen Konzepte verknüpfen. Diese Unabhängigkeit von einer konkreten fachlichen Domäne ist essentiell für die Wiederverwendung der Sicherheitsanforderungsvorlagen in unterschiedlichen fachlichen Entwicklungsprozessen.

Das Ressourcenmodul lässt sich in weitere Sub-Module unterteilen, welche verschiedene Aspekte einer Ressource aus Sicht der Sicherheitsanforderungsanalyse beschreiben. Hierbei sind insbesondere die Eigenschaften wie der Wert einer Ressource, die vorhandenen sicherheitsrelevanten Schwachstellen und der Schutzbedarf relevant. Die Einteilung in Submodule ist in Abbildung 36 dargestellt. Eine Verbindung zwischen den einzelnen Sub-Modulen ist aufgrund der in dieser Arbeit vorgestellten Detailstruktur zunächst nicht vorgesehen, kann aber bei einer zukünftigen Evolution hinzugefügt werden. Um die lose Kopplung auch bei der Evolution der Sub-Module zu ermöglichen, sind die Assoziationen auf ein notwendiges Minimum zu beschränken. Der Detaillierungsgrad der Spezifikation von Sicherheitsanforderungsvorlagen ist somit abhängig von der Ausprägung der Sub-Module. Eine detaillierte Beschreibung der Submodule führt zu einer komplexeren Sicherheitsanforderungsvorlage.

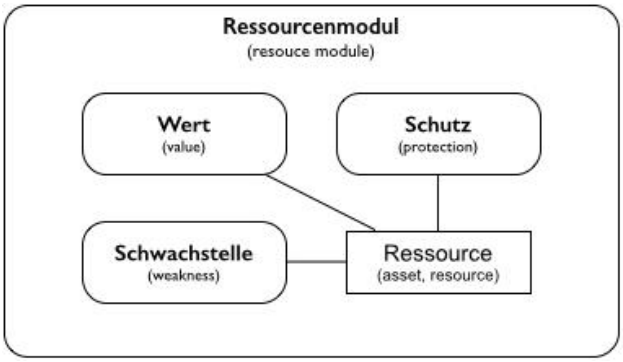

Abbildung 36: Interne Strukturierung des Ressourcenmoduls

Sub-Modul Wert

Ein Bedürfnis nach Sicherheit ergibt sich naturgemäß aus der Feststellung, dass die in einem Software-System eingesetzten Ressourcen einen Wert für die an dem System beteiligten Akteure darstellen. Dieser Wert kann dabei sowohl von materieller, wie beispielsweise der Verlust oder die Beschädigung von Personen, Geld, materielles Eigentum, etc., als auch immaterieller Natur, wie zum Beispiel der Verlust oder die Beschädigung von Informationen, Ansehen, Vertrauen, etc., sein [HL+08] [WM05:30]. Die Eigenschaft des Ressourcenwertes wird im Domänenmodell durch das Wert-Element dargestellt. Die Beziehung zwischen dem Ressource-Element und dem Wert-Element wird als Komposition modelliert, da der Wert abhängig von der Ressource ist.

Für unterschiedliche Akteure kann dabei der Wert der Ressource unterschiedlich hoch ausfallen, wodurch die Interessen der Beteiligten priorisiert werden müssen. Die Ressourcen sollen daher vor Schäden und Beeinträchtigungen geschützt werden. Oft müssen dafür die zu schützenden Ressourcen sowie ihr Wert zuerst ermittelt werden, da diese den beteiligten Akteuren nicht klar sind [SF+05:90]. Dieser Sachverhalt wird im Domänenmodell mittels der Assoziationsklasse „Interessenten“ in der Komposition zwischen Wert- und Ressource-Element modelliert. Die Elemente des Wert-Moduls sind in Abbildung 37 wiedergegeben.

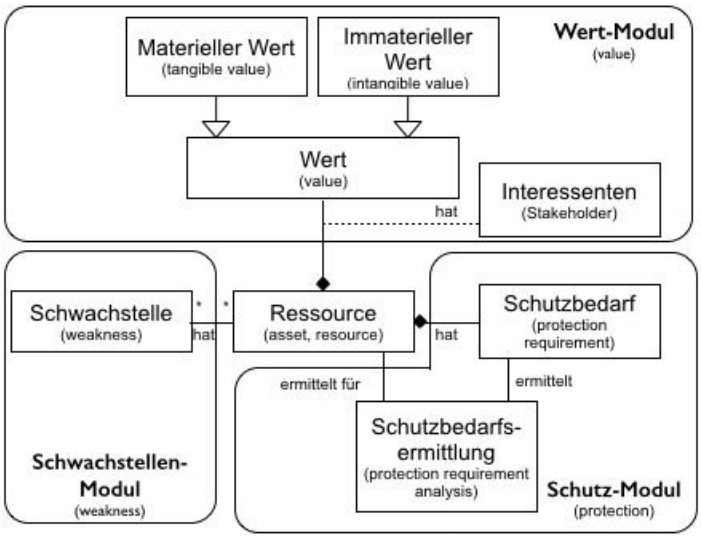

Abbildung 37: Elemente des Ressourcenmoduls

Sub-Modul Schutz

Abhängig von der zu schützenden Ressource und deren Wert kann deren Schutz unterschiedlich stark ausgeprägt sein. Der Schutzbedarf einer Ressource gibt darüber Auskunft, wie wichtig der Schutz einer Ressource ist [Ec09:176]. Somit lassen sich unterschiedliche Sicherheitsanforderungen für ähnliche Ressourcen mit jeweils abweichenden Schützbedürfnissen spezifizieren. Der Schutzbedarf wird ebenso wie der Wert der Ressource als Komposition zwischen dem Ressource-Element und dem Schutzbedarf-Element modelliert. Es ist dabei offen, ob der Schutzbedarf in quantifizierbaren Werten oder in qualitativen Kategorien gemessen wird, wobei die letztere Form üblicherweise eingesetzt wird [Ec09].

Der Schutzbedarf kann auf unterschiedliche Art und Weise ermittelt werden. Beispielsweise werden im Grundschutzhandbuch des Bundesamtes für Sicherheit in der Informationstechnologie (BSI, [BSI-GSHB]) der Schutzbedarfsermittlung für Ressourcen sogenannte Schutzbedarfskategorien zugrundegelegt, welche qualitative Angaben (niedrig bis mittel, hoch, sehr hoch) zur Schadenswirkung angeben. Um eine Ressource einer Schutzbedarfskategorie zuzuteilen, wird empfohlen, ein generisches Schadensszenario auf die jeweilige Ressource anzuwenden und mögliche Konsequenzen zu ermitteln [Ec09:176ff]. Eine andere Art den Schutzbedarf von Ressourcen zu ermitteln und festzulegen wird beispielsweise durch das Common-Criteria-Rahmenwerk vorgestellt [CC09].

Um die verschiedenen Arten der Schutzbedarfsermittlung zu unterstützen, wird im Domänenmodell das Element „Schutzbedarfsermittlung“ eingeführt, welches von konkreten Vorgehensweisen und Methoden abstrahiert. Das Schutzbedarfsermittlung-Element ist mit dem Ressource-Element assoziiert, welches als minimale Eingabe für das Ermittlungsverfahren verwendet wird. Ausgabe des Verfahrens ist der Schutzbedarf einer Ressource, welches ebenfalls als Assoziation zum Schutzbedarf-Element modelliert wird. Die Elemente sind ebenfalls in Abbildung 37 dargestellt.

Zwar können zur Ermittlung weitere Eingaben berücksichtigt werden, wie beispielsweise der ermittelte Wert einer Ressource und mögliche Bedrohungen, jedoch wird auf die explizite Modellierung dieser Assoziationen im Domänenmodell zunächst verzichtet. Grund hierfür ist hauptsächlich die Reduzierung von unnötigen Assoziationen, so dass das Modul weiter entwickelt werden kann. Insbesondere kann auf diesem Abstraktionsniveau keine allgemeine Aussage über die Verwendung dieser Eingabewerte bei der Schutzbedarfsermittlung gemacht werden.

Sub-Modul Schwachstellen

Konkrete Schäden an Ressourcen werden durch sogenannte Schwachstellen ermöglicht, welche von Bedrohungen ausgenutzt werden [Ec09:15] [WM05:13] [FH06:282] [OWASP]. Häufig wird auch der Begriff Verwundbarkeit synonym zum Begriff der Schwachstelle verwendet. Während in vielen englischsprachigen Literaturquellen meist nur der Begriff Verwundbarkeit verwendet wird, unterscheidet Eckert zusätzlich noch zwischen Schwachstellen (engl. weakness) und Verwundbarkeiten (engl. vulnerability). Schwachstellen werden dabei als allgemeine Schwächen oder verwundbare Stellen eines Systems definiert [Ec09:14]. Verwundbarkeiten hingegen sind spezielle Schwachstellen, welche die Sicherheitsmechanismen eines Systems betreffen [Ec09:15]. Aufgrund der Unabhängigkeit von einer konkreten Fachdomäne wird im Domänenmodell der generische Begriff „Schwachstelle” zur Repräsentation dieses Sachverhaltes eingesetzt. Durch Einsatz der Sicherheitsdomäne als betrachtete fachliche Domäne lassen sich somit auch Verwundbarkeiten untersuchen.

Schwachstellen stellen eine Eigenschaft von Ressourcen dar, wodurch das Schwachstelle-Element und das Ressource-Element assoziiert sind. Dabei kann eine Schwachstelle in mehreren Ressourcen auftreten, zum Beispiel durch die Verwendung einer fehlerhaften Programmbibliothek in mehreren Software-Produkten. Ebenso kann eine Ressource mehrere Schwachstellen aufweisen. Das Konzept

der Schwachstelle ist die Schnittstelle zum Bedrohungsmodul, in welchem die Begriffe der Bedrohung und des Angriffs genauer spezifiziert werden.

Schwachstellen und Verwundbarkeiten von Systemen werden häufig untersucht, dokumentiert und veröffentlicht, um andere Personen auf diese möglichen Schwachstellen hinzuweisen [WM05:31]. Werden Angriffe auf bekannte Schwachstellen eines Systems durchgeführt, so wird in diesem Zusammenhang auch von einer Ausnutzung (engl. exploit) der Verwundbarkeit gesprochen.

5.1.3 Bedrohungsmodul

Das Bedrohungsmodul umfasst wesentliche Konzepte, welche für die Durchführung und Ergebnisse einer Bedrohungs- und Risikoanalyse notwendig sind. Hierdurch werden Sicherheitsanforderungen mit denjenigen Bedrohungen verknüpft, welche für den Schutz von fachlichen Ressourcen relevant sind. Zentrales Element ist hierbei, wie in Abbildung 38 verdeutlicht, das Konzept der Bedrohung, welches als zentrale Schnittstelle zu den restlichen Modulen fungiert. Die detaillierte Beschreibung der Konzepte des Bedrohungsmodul wird in weiteren Sub-Modulen durchgeführt. Zum einen werden in einem Angriffs-Modul Angreifer sowie Angriffe abstrakt beschrieben. Zum anderen werden in einem Risiko-Modul die Konzepte zur Spezifikation von Risikoeigenschaften von Bedrohungen spezifiziert.

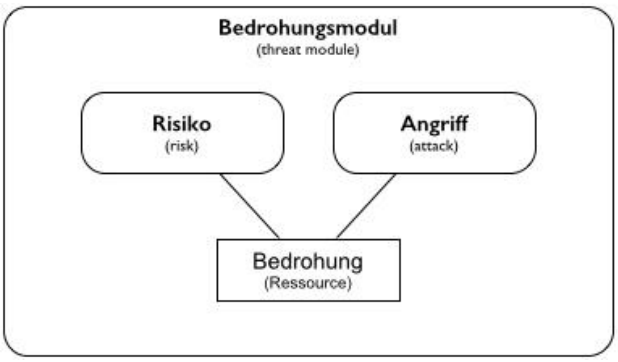

Abbildung 38: Innere Struktur des Bedrohungsmoduls

Die Modularisierung zur unabhängigen Beschreibung von Konzepten ist insbesondere im Fall von Angriffen und Bedrohungen vorteilhaft. So lassen sich diese zunächst unabhängig von konkreten Sicherheitsanforderungen mittels des Domänenmodells spezifizieren, wodurch neue entdeckte Angriffe und Bedrohungen für Software-Systeme kategorisiert und anschließend geeignete Sicherheitsanforderungen und -maßnahmen spezifiziert werden können.

Modulschnittstellen

Die Schnittstelle des Bedrohungsmoduls mit dem Ressourcenmodul bildet das Bedrohung-Element. Eine Bedrohung (engl. threat) stellt ein potentielles und unerwünschtes Ereignis dar. Ein solches Ereignis stellt eine Gefahr gegenüber den zu schützenden Ressourcen dar und könnte diese beschädigen, indem zum Beispiel ein Wertverlust herbeigeführt wird [FH06:282]. Bedrohungen sind mögliche, allgegenwärtige Ereignisse und können mit oder ohne Absicht, beispielsweise durch Unfälle und Naturkatastrophen, ausgelöst werden. Ebenso können unterschiedliche Personen und Objekte Bedrohungen hervorrufen [WM05:31]. Zudem stehen Bedrohungen rekursiv in Beziehung, wodurch eine Bedrohung der Anlass für weitere Bedrohungen sein kann [FE09].

Eine Erweiterung der Schnittstelle stellt das Angriff-Element dar. Der Begriff der Bedrohung ist eng verknüpft mit dem Begriff des Angriffs. In der Literatur finden sich unterschiedliche Interpretationen dieser Begriffe, welche verschiedene semantische Eigenschaften jeweils einer der beiden Begriffe zuordnen. Einigkeit besteht jedoch zumeist in der Abstraktion der Begriffe. Unter Bedrohungen werden zu meist alle potentiellen, Schäden verursachenden Ereignisse oder Personen verstanden. Dagegen stellen Angriffe (engl. attack) konkrete Ereignisse dar, welche zu einem Schaden an einer Ressource führen [Ec09:17] [FH06:282] [WM05:30]. Ein Beispiel für diese Beziehung ist, dass jeder Angreifer oder Hacker eine potentielle Gefahr und somit eine Bedrohung für ein Software-System darstellt. Erst die aktive unautorisierte Interaktion eines Angreifers oder einer Gruppe von Angreifern mit einem Software-System macht die Bedrohung zu einem Angriff [WM05:31]. Die Schnittstellen sind in Abbildung 39 dargestellt.

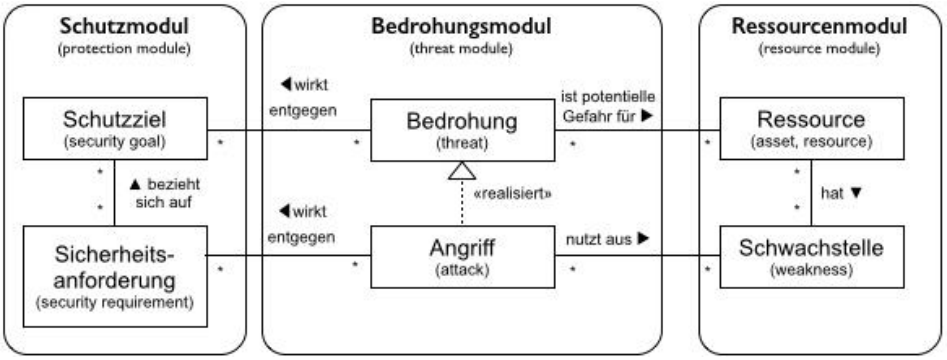

Abbildung 39: Schnittstellen des Bedrohungsmoduls

Sub-Modul Angriff und Angreifer

Das Angriff-Element ist Teil des Angriff-Sub-Moduls, in welchem Angriffe und Angreifer näher spezifiziert werden. Die hierbei spezifizierten Elemente des Domänenmodells dienen zur Analyse von Angriffstaktiken und Angreiferverhalten. Für die Spezifikation von Sicherheitsanforderungen ist dies notwendig, um die durch Angriffe entstehenden Schäden abzuschätzen und somit geeignete Anforderungen zu spezifizieren, welche das Software-System vor Angriffen schützen. Die hierfür notwendigen Konzepte werden in Abbildung 40 wiedergegeben.

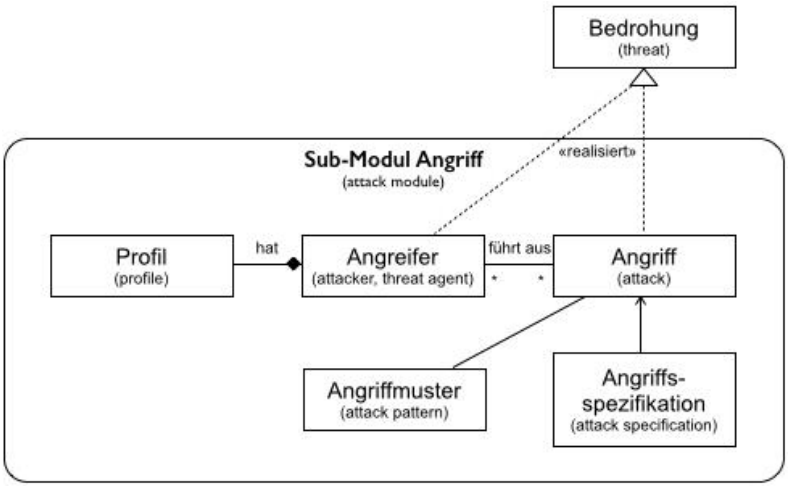

Abbildung 40: Elemente des Sub-Moduls Angriff

Für Angriffe lassen sich bereits konkrete Vorgehensweisen und Techniken identifizieren und dokumentieren. Solche wiederkehrenden Vorgehensweisen werden durch das Element „Angriffsmuster“ im Domänenmodell repräsentiert. Ein Angriff nutzt ein oder mehrere Schwachstellen eines Software-Systems aus, um Schäden an einer Ressource zu verursachen [OWASP]. In diesem Sinne stellt also ein Angriff eine Realisierung einer abstrakten Bedrohung dar. Im Domänenmodell wurde daher die Beziehung zwischen den Elementen „Angriff“ und „Bedrohung“ mittels einer Generalisierungsbeziehung modelliert.

Zur Spezifikation von Sicherheitsanforderungen ist es nötig, die Angriffe geeignet zu dokumentieren. Die Beschreibung eines Angriffs kann dabei durch verschiedene Methoden und Sprachen durchgeführt werden. In Kapitel 2 wurde mit Missbrauchsanwendungsfällen [SO05] bereits eine Beschreibungsmöglichkeit vorgestellt. Eine weitere Möglichkeit stellen Angriffsbäume dar [Sc01]. Die Kombination von ein oder mehreren dieser Methoden wird im Domänenmodell durch das Angriffsspezifikation-Element repräsentiert. Da Bedrohungen lediglich potentielle Gefahren darstellen und somit per Definition immer existieren, wurde auf eine Möglichkeit zur detaillierten Beschreibung von Bedrohungen mittels der oben genannten Methoden im Domänenmodell verzichtet. Die zuvor beschriebenen Angriffsmuster können bei der Spezifikation von Angriffen verwendet werden.

Um den Ursprung und die Gefahr eines Angriffes besser bestimmen zu können, werden häufig die den Angriff durchführenden Angreifer (engl. threat agent) untersucht. Ebenso wie das Angriff-Element stellt ein Angreifer eine Realisierung einer Bedrohung dar, welche diese mittels eines Angriffs umsetzt [WM05:31], [OWASP]. Unter dem Begriff des Angreifers werden analog zur Definition von Angriff sowohl Personen oder Gruppen verstanden, welche mutwillig und mit böswilligen Absichten Angriffe durchführen, als auch Personengruppen, welche durch unabsichtliche Fehlbedienung einen Schaden verursachen. Auch Naturereignisse wie Stürme und Überflutungen können hierzu gezählt werden [WM05:31].

Bei der Bestimmung der Gefährdungslage einer Ressource und möglicher Gegenmaßnahmen spielen Intentionen, Fähigkeiten und Ressourcen der Angreifer eine wesentliche Rolle [Sc01:39]. Häufig werden die Angriffe und Bedrohungen nicht in der Analysephase erkannt, da nur externe Angreifer betrachtet werden [Ec09:20]. Eine Vielzahl von Angriffen auf Software-Systeme wird jedoch durch interne Angreifer durchgeführt [Ec09:20] [Sc01:44].

Abhängig von den Fähigkeiten und den zur Verfügung stehenden Mitteln lassen sich Angreifer in verschiedene Kategorien einordnen, mittels denen eine Aussage über die Gefährdung durch die Angreifer gemacht werden kann. Häufig zu findende Kategorien sind zum Beispiel Hacker, Wirtschaftsspione, Kriminelle, aber auch ungenügende Kenntnisse besitzende Mitarbeiter [Sc01:20ff]. Die Eigenschaften eines Angreifers, welche bei der Analyse zu berücksichtigen sind, werden im Domänenmodell durch das Profil-Element (engl. profile) repräsentiert. Da das Profil eines Angreifers diesen eindeutig beschreibt, steht das Profil-Element in einer Kompositionsbeziehung zum Angreifer-Element.

Sub-Modul Risiko

Die alleinige Auflistung von möglichen Bedrohungen und Angriffen ist für die strukturierte Ermittlung von Sicherheitsanforderungen nicht hilfreich. Hierdurch kann nicht gewährleistet werden, dass die relevantesten Bedrohungen und Angriffe zur Erhaltung der Schutzziele in der Anforderungsanalyse berücksichtigt werden. Da jede Bedrohung nur eine potentielle Gefahr darstellt, ist es aus Kosten- und Aufwandsgründen für die später umgesetzten Sicherheitsmaßnahmen nötig, eine Filterung der Bedrohungen gemäß ihrer Priorität vorzunehmen. In der Praxis wird dies durch die Bestimmung des Risikos einer Bedrohung durchgeführt [Ec09:16], [FH06:282], [WM05:31]. Dieser Wert berechnet sich aus der Eintrittswahrscheinlichkeit und dem Schadensausmaß, d.h. der Höhe und

den Auswirkungen des Schadens, welcher beim Eintreten der Bedrohung an den zu schützenden Ressourcen verursacht wird.

Zu diesem Zweck wird im Risiko-Sub-Modul das Bedrohung-Element um die zur Bestimmung des Risikos notwendigen Elemente ergänzt, welche in Abbildung 41 dargestellt sind. Hierbei wird explizit das Bedrohung-Element erweitert, da per Definition Bedrohungen lediglich potentielle Ereignisse darstellen. Hierdurch ist es bei der Sicherheitsanforderungsanalyse möglich, die Bedrohungen zunächst gemäß ihres Risikos zu kategorisieren und priorisieren. Anschließend können aus den hoch priorisierten Bedrohungen konkrete Angriffe abgeleitet werden. Durch die Vererbungsbeziehung kann das Risiko auch für einen Angriff bestimmt werden, wodurch ebenfalls eine Kategorisierung vorgenommen werden kann.

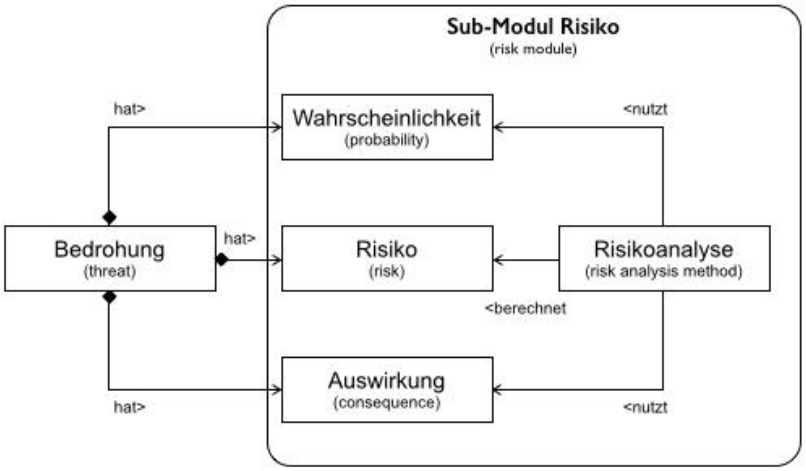

Abbildung 41: Elemente des Sub-Moduls Risiko

Die Wahrscheinlichkeit des Eintritts einer Bedrohung sowie die entsprechenden Schadensauswirkungen werden im Domänenmodell durch gleichnamige Elemente repräsentiert. Das Wahrscheinlichkeit- und das Auswirkung-Element stehen dabei in einer Kompositionsbeziehung zu einander, da die Auswirkung und die Wahrscheinlichkeit eindeutig einer Bedrohung zugeordnet sind. Das Risiko-Element repräsentiert entsprechend das Risiko einer Bedrohung und ist somit dem Bedrohungselement zugeordnet. Das Risiko kann in quantitativen und qualitativen Werten ausgedrückt werden, wie zum Beispiel 25%ige oder eine niedrige Wahrscheinlichkeit eines Angriffs [WM05:31]. Dabei ist die quantitative Bestimmung sicherlich aufgrund präziser Daten bevorzugt. In der Praxis wird allerdings häufig aufgrund von fehlenden Daten die zweite Variante verwendet.

Die Risikoermittlung ist eine besonders wichtige, wenn auch sehr schwere Aktivität in der Bedrohungs- und Risikoanalyse, da sie sehr viel Erfahrung und Kompetenz erfordert und sehr stark von der Fachdomäne einer Ressource abhängig ist [Ec09:17]. Das Verfahren zur Bestimmung des Risikos wird durch das Risikoanalyse-Element repräsentiert, wobei auch hier von konkreten Verfahren abstrahiert wird. Die zur Bestimmung des Risikos notwendigen Bedrohungseigenschaften der Wahrscheinlichkeit und der Auswirkung werden durch das Element referenziert. Das Ziel dieser Arbeit ist jedoch nicht die direkte Untersuchung und Unterstützung solcher Verfahren, weshalb die Ausgestaltung dieses Bereichs des Domänenmodells absichtlich offen gelassen wird und nur die grundlegenden auch in anderen Arbeiten zu findenden Konzepte und Zusammenhänge modelliert werden.

5.1.4 Schutzmodul

Während sich die vorhergehenden Module vor allem mit den Bedrohungen von Ressourcen eines Software-Systems beschäftigen, werden im Gegensatz dazu im Schutzmodul die Schutzbedürfnisse des Software-Systems fokussiert. Zentraler Bestandteil des Schutzmodules ist die Definition des Konzeptes der Sicherheitsanforderung, da es das relevanteste Ergebnis der Anforderungsanalyse eines Software-Systems darstellt. Daher wird deren Einordnung in den Kontext von weiteren essentiellen Konzepten, wie zum Beispiel Sicherheitsstrategie, Schutzziel, etc. durchgeführt, da diese für die Semantik einer Sicherheitsanforderung relevant sind. Das Sub-Modul Sicherheitsanforderung behandelt diese Einordnung.

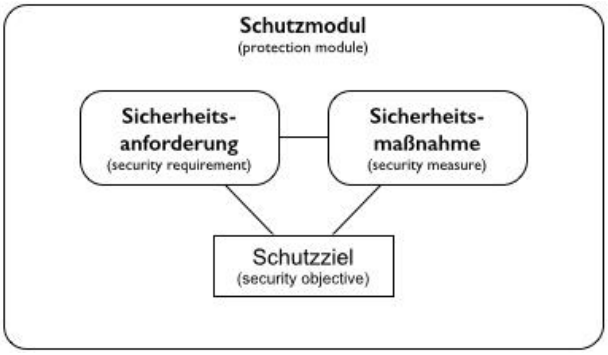

Abbildung 42: Interne Struktur des Schutzmoduls

Die Spezifikation von Sicherheitsanforderungen wird in dieser Arbeit eindeutig der Anforderungsanalysephase zugeordnet. Dennoch stellen sie einen wesentlichen Ausgangspunkt für den Entwurf von Sicherheitsmaßnahmen dar. Daher wird in dem vorgestellten Domänenmodell eine Erweiterung vorgenommen, mittels welcher der Bezug von Sicherheitsanforderungen zu Sicherheitsmaßnahmen hergestellt wird. In Rahmen des in Kapitel 4 vorgestellten Entwicklungsvorgehens dient diese Erweiterung zudem als Übergang zur dem in Kapitel 5 vorgestellten Variabilitätsmodell für Sicherheitsmaßnahmen. Hierdurch wird das Ziel verfolgt, die durch das Domänenmodell spezifizierten Vorlagen in ein durchgängiges Entwicklungsvorgehen zu integrieren. Diese Erweiterung wird dabei in dem Sub-Modul Sicherheitsmaßnahme gekapselt und kann bei Bedarf auch außer Acht gelassen werden.

Schutzziele

Schutzziele bringen den Schutzbedarf einer Ressource zum Ausdruck [Ec09:6], indem sicherheitsrelevante Eigenschaften der Ressource spezifiziert werden [WM05:9]. Das Ziel der Spezifikation von Sicherheitsanforderungen ist die Gewährleistung der Einhaltung dieser Eigenschaften. Unter Schutzzielen werden unter anderem die klassischen Schutzziele Vertraulichkeit, Integrität und Verfügbarkeit verstanden. Durch die zunehmende Komplexität der Bedrohungen und der Ressourcen werden in der Literatur weitere Eigenschaften, wie beispielsweise Authentizität, Verbindlichkeit, Anonymisierung und Pseudomisierung, definiert, welche zum Schutz der Ressource berücksichtigt werden. Abhängig vom Wert und dem Schutzbedarf können dabei mehrere Schutzziele für eine Ressource definiert werden.

Die Veränderung dieser Schutzziele bzw. Eigenschaften einer Ressource sind zudem an den Wert einer Ressource gekoppelt. Wird beispielsweise das Schutzziel der Authentizität, d.h. der ursprüngliche Zustand einer Ressource verändert, so kann dies zu einem Wertverlust der Ressource führen. Da der Wertverlust einer Ressource durch Bedrohungen verursacht wird, haben diese folglich einen negativen Einfluss auf die Schutzziele einer Ressource [WM05:9] [FE09].

Sub-Modul Sicherheitsanforderung

Die Umsetzung von Schutzzielen im Kontext der analysierten Bedrohungen einer Ressource führt zu der Spezifikation von Sicherheitsanforderungen. Um die Trennung von Analyse- und Entwurfsphase eindeutig zu beschreiben, ist die Abstraktionsebene der Anforderungen ausschlaggebend. In dem vorgestellten Domänenmodell stellen Sicherheitsanforderungen Einschränkungen hinsichtlich der Ressource und der mit ihr durchgeführten Aktionen dar. Damit orientiert sich das Domänenmodell an der von Haley et al. vorgegebenen Definition [HL+08] und vermeidet die häufig durchgeführte Spezifikation von Sicherheitsanforderungen mittels Sicherheitsmaßnahmen [TJ+08].

Die Spezifikation einer Sicherheitsanforderung hat zum Ziel, die zu erreichenden Schutzziele für eine Ressource präziser zu spezifizieren, so dass Software-Architekten diese in Form von Sicherheitsmaßnahmen umsetzen können [HL+08]. Somit stellen Sicherheitsanforderungen eine Abstraktionsstufe zwischen der Analyse- und der Entwurfsphase dar, da sie den gewünschten Effekt aber nicht die eigentliche Umsetzung beschreiben [HL+08] [SF+03]. Im Domänenmodell ist diese Beziehung durch eine Assoziation zwischen dem Sicherheitsanforderung-Element und dem Sicherheitsmaßnahme-Element umgesetzt. Letzteres ist dabei Teil der optionalen Ergänzung des Domänenmodells und repräsentiert ein in der Entwurfsphase eingesetztes Konzept. Dabei kann eine Sicherheitsanforderung durch eine oder mehrere Maßnahmen umgesetzt werden bzw. eine Maßnahme zur Umsetzung von mehreren Anforderungen eingesetzt werden.

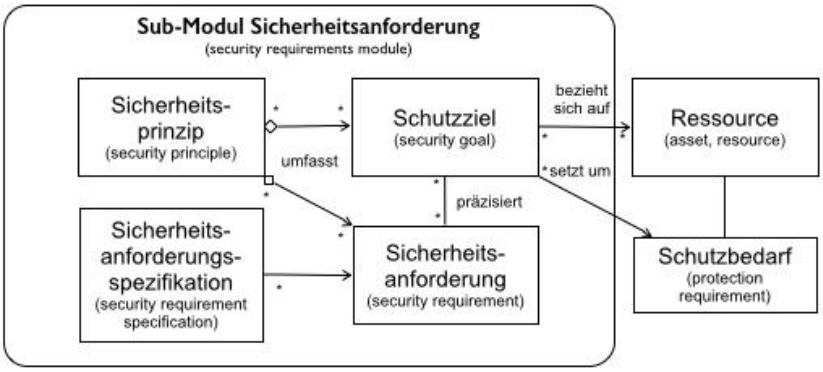

Abbildung 43: Interne Struktur des Sub-Moduls Sicherheitsanforderung

Wie in Kapitel 2 vorgestellt, können Sicherheitsanforderungen auf unterschiedliche Weise spezifiziert werden. Die konkrete Form der Spezifikation ist abhängig von dem eingesetzten Software-Entwicklungsprozess. Dies hat einen Einfluss auf die Sicherheitsanforderungsvorlagen, welche entsprechend auf die bevorzugte Form angepasst werden müssen. Um hierbei eine Vielzahl von Ansätzen zu unterstützen wird im Domänenmodell die Spezifikation von Anforderungen separat definiert und durch das Sicherheitsanforderungsspezifikation-Element repräsentiert.

Bei der Bestimmung der für eine Ressource zu verwendenden Schutzziele sowie zur Spezifikation von angemessenen Sicherheitsanforderungen lassen sich gängige Sicherheitsprinzipien verwenden. Beispiele für Sicherheitsprinzipien sind dabei das generische Prinzip der minimalen Rechte (engl. least privilege, [BM05] [HS75]). Sicherheitsprinzipien existieren auch für spezielle Fachdomänen, wie beispielsweise das Prinzip der Trennung von Verantwortlichkeiten (engl. separation of duties bzw. separation of privileges [HS75] [BM05]), welches häufig im Finanzwesen angewendet wird.

Diese Prinzipien spiegeln zum einen das aus der praktischen Erfahrung gesammelte Wissen von Sicherheitsexperten zur Konstruktion von sicheren Systemen wieder [BM05] [HS75]. Zum anderen können sie aus den Sicherheitsrichtlinien einer Organisation abgeleitet werden [HM+06]. Die Nutzung dieser Prinzipien hilft dabei, die Schutzziele und die Sicherheitsanforderungen für eine Ressource

besser zu bestimmen, da sie meist weitgehende Beschreibungen und Begründungen besitzen, die einfacher auf eine Fachdomäne übertragen werden können [BM05]. Demnach sind zur Umsetzung von Sicherheitsprinzipien mehrere Schutzziele und Sicherheitsanforderungen notwendig.

Sub-Modul Sicherheitsmaßnahmen

Wie angemerkt ist das Sicherheitsmaßnahme-Modul eine optionale Ergänzung des Schutzmodules, um die Verknüpfung von Sicherheitsanforderungen mit Sicherheitsmaßnahmen der Entwurfsphase zu ermöglichen. Um die Trennung zwischen den Phasen zu gewährleisten, werden lediglich die zur Abbildung von Anforderungen auf Maßnahmen notwendigen Konzepte beschrieben. Wird in einem Entwicklungsprozess bereits ein Domänenmodell für Sicherheitsmaßnahmen verwendet, so kann das Sub-Modul Sicherheitsmaßnahmen durch dieses ausgetauscht werden. Die im Rahmen dieser Arbeit verwendeten Konzepte werden durch Abbildung 44 wiedergegeben.

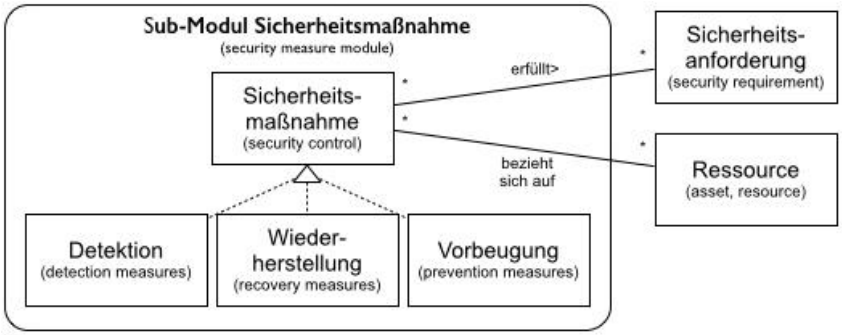

Abbildung 44: Interne Struktur des Sub-Moduls Sicherheitsmaßnahme

Das zentrale Element des Sub-Moduls stellt das Element Sicherheitsmaßnahme dar. Im Gegensatz zur Sicherheitsanforderung repräsentiert es eine abstrakte Sicherheitslösung, mittels welcher eine Sicherheitsmaßnahme umgesetzt werden kann. Eine Sicherheitsmaßnahme bezieht sich dabei auf eine oder mehrere Ressourcen, für welche die Sicherheitsanforderung definiert wird. Die Ressource ist idealerweise unabhängig von der Sicherheitsmaßnahme und wird durch sie nicht direkt beeinflusst. Diese Entwurfsstrategie ist jedoch nicht für jede Maßnahme umsetzbar. Aus den für die Ressourcen definierten Schutzbedürfnissen ergeben sich mehrere Möglichkeiten zur Umsetzung der Sicherheitsmaßnahmen. Diese als Taktiken bezeichneten Möglichkeiten umfassen Maßnahmen zur Detektion von Angriffen, Maßnahmen zur Wiederherstellung von Ressourcen nach einem Angriff sowie vorbeugende Maßnahmen, welche Angriffe verhindern sollen [FH06].

5.2 Sicherheitsanforderungskatalog

Mittels des im vorangegangenen Kapitel 5.1 vorgestellten Domänenmodells wurde die konzeptionelle Grundstruktur für Sicherheitsanforderungsvorlagen spezifiziert. Gemäß dem im Kapitel 4 beschriebenen Entwicklungsverfahren werden Sicherheitsanforderungen auf Grundlage dieses Domänenmodells zu Vorlagen aufbereitet. Hierdurch wird eine Sammlung von Vorlagen erarbeitet, welche in der konkreten Entwicklung eines Software-Systems eingesetzt werden. Das Ziel dabei ist, die Sicherheitsanforderungsanalyse durch die strukturierte Aufbereitung von bestehendem Sicherheitswissen und bewährten Methoden zu vereinfachen. Es sollen vor allem Entwickler ohne Hintergrundwissen bei der Erhebung, Analyse und Spezifikation von Sicherheitsanforderungen unterstützt werden.

Auf Basis des vorgestellten Domänenmodells wird in diesem Abschnitt die Katalogisierung von Vorlagen für Sicherheitsanforderungen behandelt. Ziel dabei ist es, die Verwendung des Domänenmodells zur strukturierten Spezifikation von wiederverwendbaren Sicherheitsanforderungen

zu demonstrieren. Die Vorlagen wurden dabei zunächst aus bestehenden Sicherheitsstandards, wie zum Beispiel dem Common-Critera-Anforderungskatalog [CC09] abgeleitet.

5.2.1 Zugrundeliegende Konzepte

Während das Domänenmodell die Struktur der Vorlagen definiert, müssen konkrete Sicherheitsanforderungsvorlagen für den wiederverwendbaren Einsatz in einem Entwicklungsprozess katalogisiert werden. Hierzu wird eine Ablagestruktur benötigt, mittels welcher die Vorlagen sowie deren Komponenten flexibel verwaltet werden können. Im folgenden Abschnitt werden die konzeptionellen Grundlagen einer solchen Ablage in Form eines Vorlagenkatalogs für Sicherheitsanforderungsvorlagen spezifiziert.

Modularität der Vorlagen

Durch den modularen Aufbau des zugrundeliegenden Domänenmodells lassen sich die Sicherheitsanforderungsvorlagen im Katalog in einzelne Komponenten zerlegen und zusammensetzen. Die einzelnen Komponenten spezifizieren dabei je nach Art konkrete Angriffe, Angreifertypen sowie Schutzziele und Sicherheitsanforderungen. Hierdurch wird die bereits angesprochene unabhängige Evolution der einzelnen Sub-Domänen in Form von einzelnen Komponenten erreicht. Durch die Aufteilung in einzelne Komponenten wird die Komplexität der Beschreibung der Vorlagen zunächst auf einzelne Komponenten reduziert.

In Abbildung 45 ist die modulare Struktur des Vorlagenkatalogs dargestellt. Die einzelnen Komponenten werden zunächst unabhängig voneinander spezifiziert. Dies ermöglicht es Sicherheitsexperten, Sicherheitswissen wiederverwendbar aufzubereiten und zunächst ohne Verbindung zu spezifischen Anforderungsvorlagen im Katalog abzulegen. Dies ist zum Beispiel hilfreich, um neue Angriffe zu katalogisieren, zu welchen noch keine Sicherheitsanforderungen spezifiziert wurden. Ebenso können neue Schutzziele hinzugefügt werden, ohne dass diese ebenso durch im Katalog vorhandene Sicherheitsanforderungen umgesetzt werden.

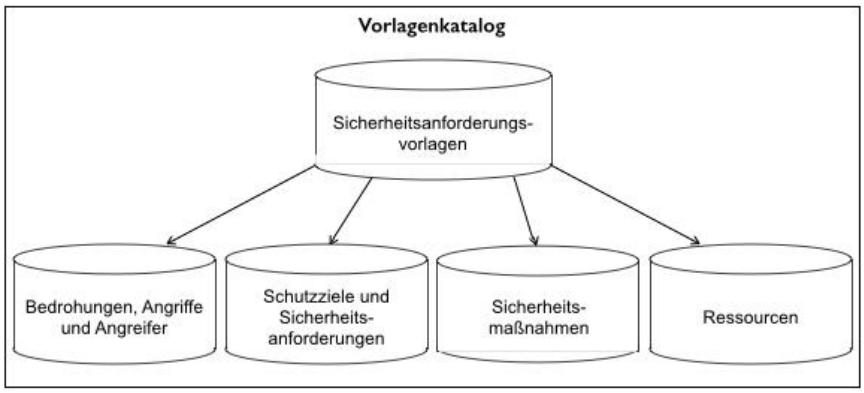

Abbildung 45: Grundlegende Struktur des Vorlagenkatalogs

Entsprechend der Module im Domänenmodell werden verschiedene Komponenten der Vorlagen unabhängig voneinander spezifiziert. Dabei werden die Elemente des Bedrohungsmoduls, d.h. Bedrohungen, Angriffe und Angreiferprofile, in einem entsprechenden Komponentenkatalog aufbereitet und zur Verfügung gestellt. Entsprechend werden die Elemente des Schutz- und des Ressourcenmoduls in entsprechenden Artefakten im Katalog repräsentiert. Abstrakte Sicherheitsmaßnahmen stellen optionale Artefakte dar und können ebenfalls im Vorlagenkatalog spezifiziert werden. Die Kombinationen der einzelnen Komponenten zu Vorlagen werden in einer eigenen Ablage hinterlegt.

Bei der Kombination der verschiedenen Komponenten werden die im Domänenmodell definierten Schnittstellen berücksichtigt, um die Kompatibilität der Komponenten zu gewährleisten. Durch die Aufteilung in einzelne unabhängige Komponenten, können diese in unterschiedlichen Vorlagen eingesetzt werden. Die Sicherheitsanforderungsvorlagen werden ebenfalls im Vorlagenkatalog abgelegt und stehen Entwicklern in einem Analyseprozess zur Verfügung.

Generische und anwendungsspezifische Komponenten

Bei der Ablage von elementaren Komponenten sowie der Konstruktion von Sicherheitsanforderungsvorlagen lassen sich generische sowie anwendungsspezifische Artefakte unterscheiden [SF+03]. Im ersten Fall lassen sich sowohl die einzelnen Komponenten als auch die Vorlagen auf beliebige Fachdomänen anwenden. Somit lassen sich Bedrohungen und Angriffe sowie Schutzziele und Sicherheitsanforderungen beschreiben, welche generell in allen Software-Systemen unabhängig von einer spezifischen fachlichen Domäne berücksichtigt werden sollten. Im letzteren Fall beziehen sich die Artefakte auf Bedrohungen sowie Sicherheitsanforderungen, welche nur in einer bestimmten Fachdomäne auftreten und sinnvoll eingesetzt werden. Diese können somit in anderen Fachdomänen vernachlässigt werden. Dieser Sachverhalt ist in Abbildung 46 dargestellt.

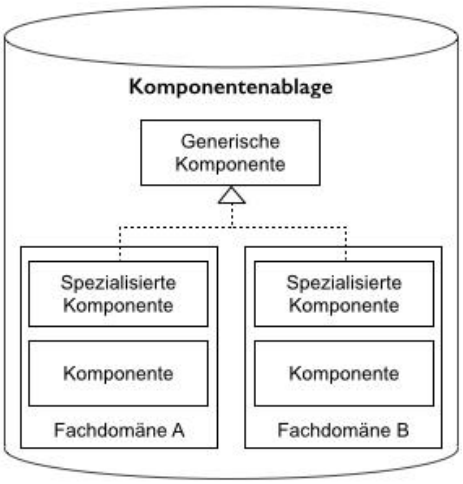

Abbildung 46: Generische und spezialisierte Komponenten im Vorlagenkatalog

Zur Beschreibung von anwendungsspezifischen Komponenten können generische Komponenten verwendet werden. Hierbei können letztere in ihrer bestehenden Beschreibung weiter spezialisiert werden, in dem die vorhanden Spezifikation weiter eingeschränkt werden. Hierdurch wird die Komponente auf den fachlichen Kontext eines Software-Systems angepasst. So lassen sich beispielsweise Interaktionen und Auswirkungen eines generischen Angriffs detaillierter beschreiben, sobald der Kontext der Fachdomäne mit berücksichtigt wird. Des Weiteren können die generischen Komponenten auch um zusätzliche Beschreibungen ergänzt werden.

Bei der Evolution der einzelnen Komponenten sowie der Sicherheitsanforderungsvorlagen ist es ebenfalls möglich, dass eine anwendungsspezifische Komponente zu einer generischen Komponente wird. Ausgangspunkt dabei sind anwendungsspezifische Vorlagen, welche in verschiedenen Software-Systemen in mehreren Fachdomänen angewendet wurden. In diesem Fall stellen die Komponenten dieser Vorlagen Kandidaten dar, für welche die Anwendbarkeit als generische Komponente zu überprüfen ist. Gegebenenfalls ist dabei die Spezifikation der Komponente zu überarbeiten, so dass die anwendungsspezifischen Bezüge aus der Beschreibung entfernt werden und die Allgemeingültigkeit der Komponente hergestellt wird. Dieses Vorgehen ermöglicht es, einen Katalog von zunächst

anwendungsspezifischen Komponenten und Vorlagen zu spezifizieren und anschließend generische Komponenten daraus abzuleiten.

5.2.2 Schema der Vorlagen

Das im vorherigen Abschnitt vorgestellte Konzept für eine Katalogstruktur ermöglicht die Ablage von Sicherheitsanforderungsvorlagen sowie deren einzelne Bestandteile. Diese Komponenten werden in dem Katalog in einem Format beschrieben, welches auf UML-Klassendiagrammen basiert und im folgenden Abschnitt spezifiziert wird. Dieses setzt das abstrakte Domänenmodell somit in eine konkrete Syntax um [SV+07]. Diese konkrete Syntax der Beschreibung der Komponenten der Sicherheitsanforderungsvorlagen wird dabei in Form von Schemata vorgenommen.

Schema für Bedrohungen, Angriffe und Angreifer

Für die Beschreibung von unerwünschten Ereignissen wird gemäß dem Domänenmodell eine Unterteilung von Bedrohungen und Angriffen im Vorlagenkatalog vorgenommen. Eine Bedrohung-Komponente entspricht dabei dem gleichnamigen Element des Domänenmodells und umfasst eine Beschreibung der Bedrohung sowie das Risiko der Bedrohung, welches sich aus dem Auswirkungsgrad sowie der Eintrittswahrscheinlichkeit ergibt. Die Methode, welche zur Ermittlung des Risikos eingesetzt wurde, wird ergänzend referenziert. Ebenso werden auch die Schutzziele, welche durch die Bedrohung verletzt werden, referenziert. Diese Zusammenhänge sind in Abbildung 47 dargestellt.

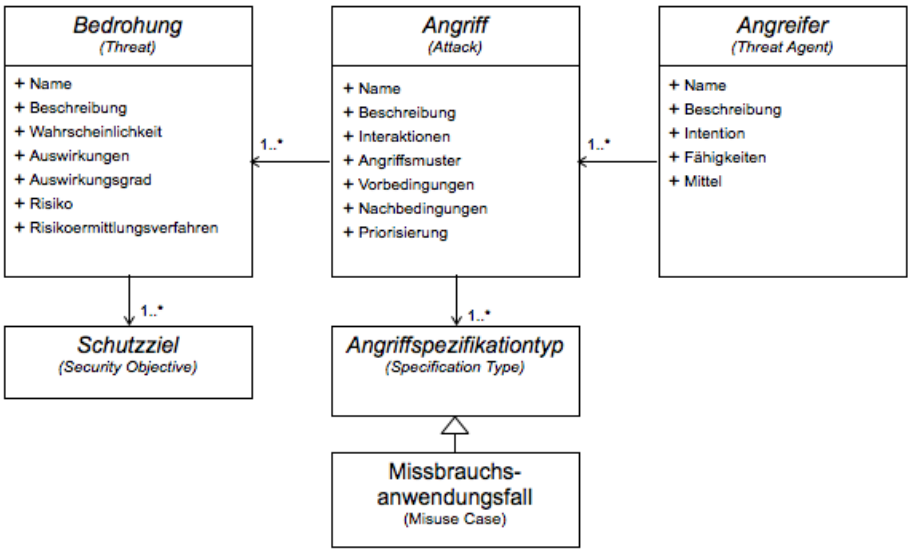

Abbildung 47: Schema für Bedrohungen, Angriffe und Angreifer

Da Bedrohungen potentielle Ereignisse darstellen, werden die Bedrohung-Komponenten als Kategorien von Angriffen angesehen. Da die Anzahl der Bedrohungskategorien überschaubar ist, kann davon ausgegangen werden, dass die Einträge in dem Vorlagenkatalog auf einem stabilen Niveau bleiben. Die Kategorisierung mittels Bedrohungen wird eingeführt, um eine Unterstützung für Entwickler mit wenig Kenntnissen in der Sicherheitsdomäne bei der Nutzung des Anforderungskatalogs bereitzustellen. Die Bedrohung-Komponenten ermöglichen es, Entwicklern einen Überblick über verschiedene Angriffe zu erhalten, so dass die Notwendigkeit der Berücksichtigung der Angriffe eingeschätzt werden kann.

Angriff-Komponenten referenzieren Bedrohungen, um sich in eine oder mehrere Bedrohungskategorien einzuordnen. Die Beschreibung von Angriffen setzt sich zunächst aus einer zusammen-

fassenden Beschreibung des Angriffs, welcher die Intention des Angriffs wiedergibt, und einem eindeutigen Namen zusammen. Gemäß des Domänenmodells gehört zudem eine Angriffsspezifikation, ein Angriffsmuster sowie die Zuweisung von Angreifern zur Angriffsbeschreibung. Die Angriffsspezifikation beschreibt die Interaktionen zwischen einem Angreifer und der Ressource, welche während des Angriffs von beiden Parteien durchgeführt werden.

Die Angriffspezifikation kann dabei mittels eines Angriffsmusters beschrieben werden, welche typische Interaktionen von Angriffen dokumentieren. Als Ergänzung zu den Interaktionen werden Vor- und Nachbedingungen spezifiziert, welche erfüllt sein müssen, bevor die Interaktion eintreten kann bzw. welche nach dem Angriff erfüllt sind. Diese Spezifikation von Bedingungen und Interaktionsschritten orientiert sich an der Beschreibung von Szenarien und Anwendungsfällen in der fachlichen Software-Entwicklung [BD10]. Wird ein Angriff durch die informelle Beschreibung als relevant angesehen, so bieten diese Einträge eine Möglichkeit zur Priorisierung des Angriffs. Die zur Modellierung eines Angriffs eingesetzte Spezifikationsform wird durch die Komponente „Angriffsspezifikationstyp“ referenziert. Hierbei kann es sich beispielsweise um einen Missbrauchsanwendungsfall handeln.

Die Spezifikation von Angreifer-Komponenten umfasst die Intention und die Fähigkeiten mit welchen ein Angreifer Angriffe durchführt. Ebenso werden die dem Angreifer zur Verfügung stehenden Mittel und Ressourcen beschrieben. Hierdurch wird ein klares Bild über die Absichten eines Angreifers spezifiziert, mittels welchen die Eintrittswahrscheinlichkeit bzw. die Auswirkungen eines Angriffs besser abgeschätzt werden können. Eine Angriffsbeschreibung wird von einer Angreiferbeschreibung referenziert. Grund hierfür ist die Verwendung von mehreren Angriffskomponenten in unterschiedlichen Angreiferbeschreibungen. Somit kann ein Angreifer je nach dessen Intention, Fähigkeiten und zu Verfügung stehenden Mitteln unterschiedliche Angriffe einsetzen.

Schema für Schutzziele und Sicherheitsanforderungen

Neben den Angriffsbeschreibungen stellt die Spezifikation von Sicherheitsanforderungen einen weiteren zentralen Bestandteil des Vorlagenkataloges dar. In Abbildung 48 ist die Umsetzung des Schemas für eine Sicherheitsanforderungsvorlage dargestellt. Eine Sicherheitsanforderung wird zunächst mit einem Namen versehen, welche die Semantik der Anforderung eindeutig wiedergibt. Anschließend wird die Anforderung selbst in Textform beschrieben. Hierbei werden Variablen verwendet, welche die Freiheitsgrade der Vorlage darstellen. Die Formulierung orientiert sich dabei an die von Firesmith vorgeschlagene Formulierung für Sicherheitsanforderungen [Fi04]:

„Die [Ressource] soll zur Erhaltung von [Schutzziel] vor [Angriff] geschützt werden.”

Die in den eckigen Klammern spezifizierten Begriffe stellen dabei typisierte Platzhalter dar, welche durch entsprechende Vorlagenkomponenten instanziiert werden müssen. Die Angriffsbeschreibung verweist dabei auf eine zuvor beschriebene Angriffskomponente. Durch die externe Spezifikation der Vorlagen der Komponenten werden diese in der Sicherheitsanforderungsvorlage referenziert. Dabei steht der Ressourcen-Platzhalter stellvertretend für die zu schützenden fachlichen Ressourcen und muss bei der Instanziierung der Vorlage konkretisiert werden.

Da die Schutzziele in den Beschreibungen von verschiedenen Sicherheitsanforderungsvorlagen verwendet werden, wird deren ausführliche Beschreibung ebenfalls in separate Komponenten ausgelagert. Das Schema zur Spezifikation von Schutzzielen ist dabei einfach gehalten. So wird ein Schutzziel durch einen eindeutigen Namen sowie durch eine allgemeine Definition beschrieben. Somit stellen Schutzziele eigenständige und abgeschlossene Komponenten dar, welche keine Beziehung zu anderen Komponenten besitzen.

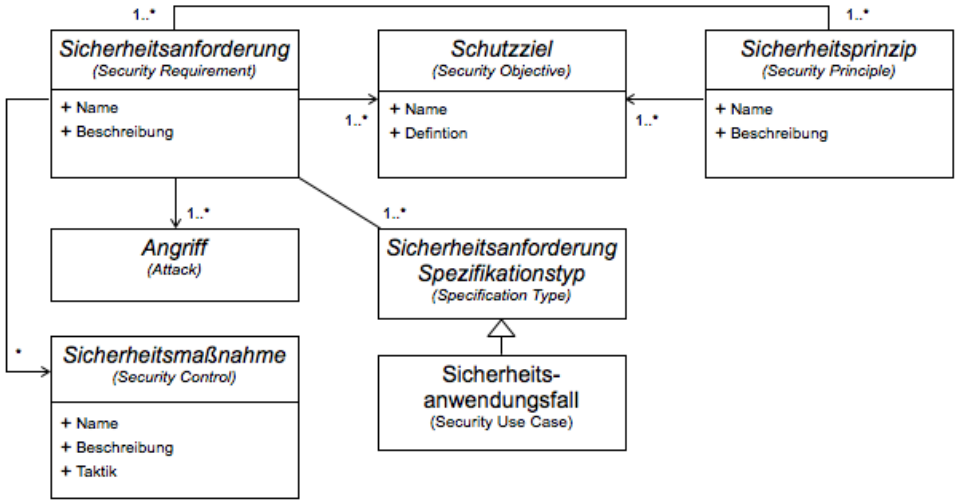

Abbildung 48: Schemata für Schutzziel, Sicherheitsanforderung und -prinzip

Die textbasierte Methode zur Spezifikation der Sicherheitsanforderung stellt eine abstrakte Form dar, welche häufig in Pflichtenheften vorzufinden ist [So07]. Neben dieser Form werden auch andere text- sowie vorallem modellbasierte Spezifikationensformen eingesetzt [Fi03b], um die Sicherheitsanforderung in Systemdiagrammen zu integrieren. Die Umsetzung der Sicherheitsanforderung mittels verschiedener weiterer Methoden kann ebenfalls in der Vorlage referenziert werden und setzt somit das Sicherheitsanforderung-Spezifikationtyp-Element des Domänenmodells um. Da über die Form a priori keine Aussage getroffen werden kann, wird hierfür im Katalog kein Schema vorgesehen. Eine Möglichkeit zur Umsetzung ist die Referenzierung von außerhalb des Kataloges abgelegten Komponenten, welche vor allem im Falle von modellbasierten Spezifikationen vorteilhaft ist.

Sicherheitsprinzipien werden im Vorlagenkatalog als eine Sammlung von Schutzzielen und Sicherheitsanforderungen repräsentiert, welche zur Vermeidung von einer Vielzahl von Angriffen eingesetzt wird. Zusätzlich sind ein eindeutiger Name sowie eine Beschreibung des Prinzips vorgesehen. Die mit dem Sicherheitsprinzip verbundenen Schutzziele und Anforderungen sowie die vermiedenen Angriffe werden entsprechend referenziert.

Schema für Sicherheitsmaßnahmen

Im Domänenmodell wurde die optionale Verbindung von Sicherheitsanforderungen mit Sicherheitsmaßnahmen vorgesehen, welche die Anforderungen umsetzen. Im Vorlagenkatalog wird diese Beziehung durch ein optionales Feld in der Vorlage für Sicherheitsanforderungen umgesetzt. Mittels dieses Feldes werden die Sicherheitsmaßnahmen anhand ihres Namens referenziert, welche zur Umsetzung der Anforderung eingesetzt werden. Somit werden auch die abstrakten Sicherheitsmaßnahmen als eigenständige Komponenten im Vorlagenkatalog spezifiziert, welche keine weiteren direkten Beziehungen zu weiteren Komponenten besitzen. Hierdurch lässt sich das optionale Vorliegen der Sicherheitsmaßnahmen umsetzen. Die Sicherheitsmaßnahmen werden durch Namen und Beschreibung charakterisiert. Zudem wird eine Einteilung in die angemerkten Taktiken Detektion, Wiederherstellung und Vorbeugung vorgenommen.

5.2.3 Beispiele für Vorlagen

Im folgenden Abschnitt wird eine Einordnung von bestehendem Sicherheitswissen zur Erhebung von Sicherheitsanforderungen in die vorgestellten Schemata vorgenommen. Ziel dabei ist es, zum einen eine Klassifizierung von dem in der Literatur behandelten Sicherheitswissen in den beschriebenen

Ansatz für einen Vorlagenkatalog vorzunehmen. Zum anderen soll demonstriert werden, wie aus den Einzelkomponenten eines Vorlagenkataloges Sicherheitsanforderungsvorlagen erstellt werden können, um einen initialen Katalog von wiederverwendbaren Vorlagen zu beschreiben, welcher Ausgangsbasis für weitere Entwicklungen darstellt.

Bedrohungen, Angriffe und Angreifer

In Abbildung 49 wird ein Auszug an möglichen Bedrohungen dargestellt, welche im Katalog abgelegt werden können [WM05] [Ra02]. Ein menschlicher Irrtum bezeichnet eine Bedrohung, welche ohne böswillige Absichten durch befugte Personen aufgrund von Fehlern in der Bedienung eintritt. Naturereignisse stellen Bedrohungen dar, welche auf Naturkatastrophen oder natürlichen Ereignissen, wie zum Beispiel Feuer, Flut, usw. beruhen. Ein Akt der Sabotage oder des Vandalismus liegt vor, wenn eine Schädigung einer Ressource oder des Ansehens einer Organisation erzielt wird. Die Bedrohung der Beeinträchtigung von Dienstgüte zielt auf die Unterbrechung einer Wertschöpfungskette ab. Anhand der Bedrohung „Diebstahl“ wird ein vollständiger Beispieleintrag gemäß der erläuterten Struktur des Vorlagenkataloges demonstriert.

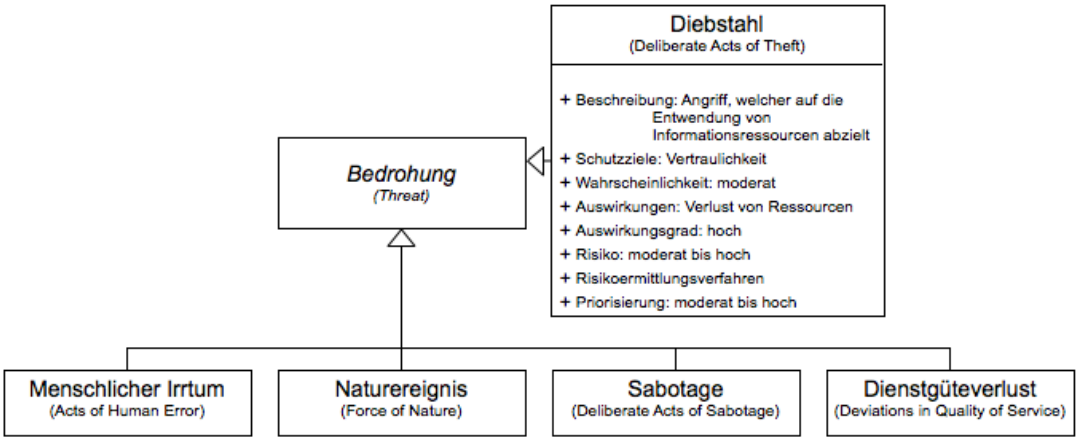

Abbildung 49: Auswahl an wiederverwendbaren Bedrohungskomponenten

Die Analyse von Angreiferprofilen ermöglicht die Auswahl von zu schützenden Ressourcen sowie die Priorisierung von Schutzzielen für ein Software-System. Neben externen Angreifern sind vor allem auch Innentäter zu berücksichtigen. Hierbei handelt es sich um Mitarbeiter, welche absichtlich oder unabsichtlich Schaden an einer Ressource verursachen In Abbildung 51 wird anhand des Angreifer-profils für Einzeltäter die entsprechende Komponente im Vorlagenkatalog beschrieben Abbildung 51 stellt ebenfalls eine einfache Übersicht über weitere Angreifertypen dar, welche die Einträge in einem Vorlagenkatalog strukturiert darstellt. Die Beschreibungen fassen dabei das in [Sc01:39-54] sowie [Ec09:20-25] gesammelte Wissen zusammen.

Prinzipiell sind alle Angriffe durch alle Angreifer durchführbar. Jedoch wird durch die Intentionen der Angreifer die Auswahl der Angriffe eingeschränkt. Zielen die Absichten eines Angreifers beispiels-weise auf den Diebstahl von Ressourcen ab, so werden andere Angriffe ausgewählt, als im Falle von Vandalismus mit der Absicht von Rufschädigung. In diesem Kontext ermöglicht also die Untersuchung und das Verständnis von existierenden Angriffsmethoden eine strukturiertere und effizientere Analyse von Schutzzielen und Sicherheitsanforderungen. Hierdurch wird gewährleistet, dass die Sicherheitsmaßnahmen die Angriffe abwehren bzw. mindern können.

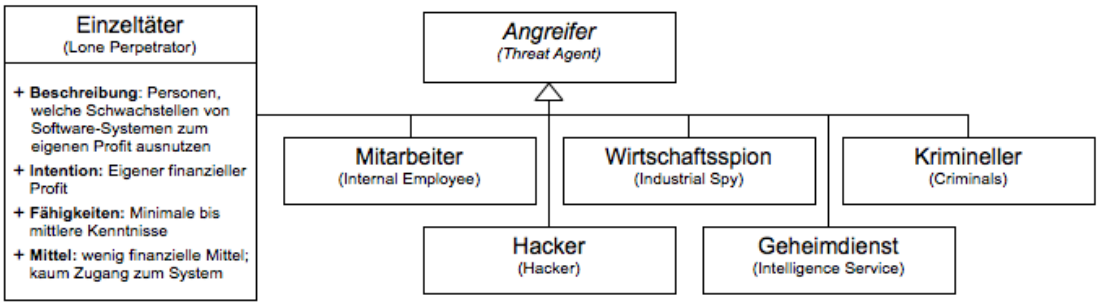

Abbildung 51: Auswahl an wiederverwendbaren Angreifer-Komponenten

In Abbildung 50 ist eine Beschreibung eines Angriffs für den Vorlagenkatalog für das Beispiel „Unbefugter Zugriff" [WM05:43-44] aufgezeigt. Weitere Beispiele, welche Einträge in einem Vorlagenkatalog darstellen können, werden ebenfalls als Übersicht in Abbildung 50 aufgezeigt [WM05:38-68] [Sc01:21-38]. Dabei ist zu bemerken, dass die angeführten Beispiele Angriffe auf abstrakter Ebene darstellen und somit in ihrem Abstraktionsgrad zu den Schutzzielen und Sicherheitsanforderungen passen. Somit wurden Angriffe, welche ein sehr technisches Niveau voraussetzen, aus dem Überblick ausgeschlossen.

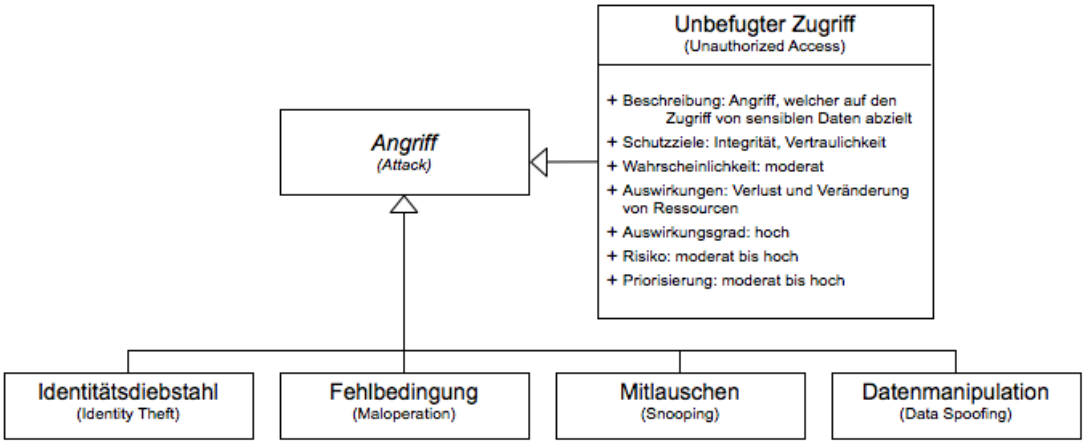

Abbildung 50: Auswahl an wiederverwendbaren Angriffen

Schutzziele und Sicherheitsanforderungen

Zu den klassischen Schutzzielen werden Vertraulichkeit (confidentiality, [Ec09:8]), Verfügbarkeit (availability, [Ec09:10]) sowie Integrität (integrity, [Ec09:7]) gezählt. In Abbildung 52 wird anhand des Schutzzieles Integrität die entsprechende Komponente in einem Vorlagenkatalog spezifiziert. Diese Schutzziele sind die minimal notwendigen Schutzziele, welche durch ein Software-System berücksichtigt werden müssen und sind somit als obligatorische Einträge in einem Vorlagenkatalag anzusehen. In Abbildung 52 werden weitere Schutzziele vorgestellt, welche abhängig von der Fachdomäne eines Software-Systems berücksichtigt werden können.

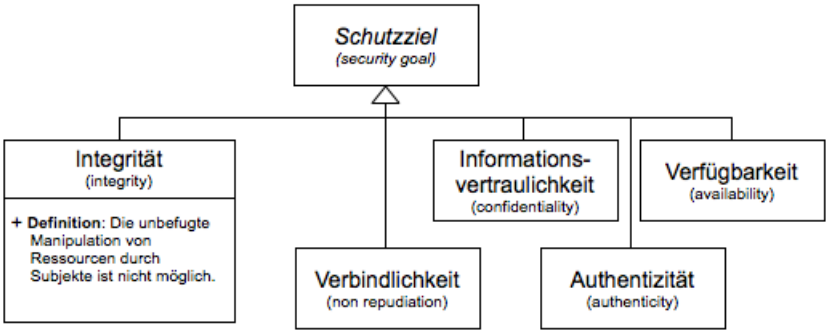

Abbildung 52: Auswahl an wiederverwendbaren Schutzzielen

Aus den bisher beschriebenen Beispielen für Einträge in Komponenten lässt sich eine wiederverwendbare Sicherheitsanforderung spezifizieren, welche dem durch einen Einzeltäter durchgeführten Angriff „Unbefugter Zugriff" vorbeugen soll. Ziel ist es, für eine Ressource das Schutzziel der Integrität zu gewährleisten. Abbildung 53 zeigt die Sicherheitsanforderung in Form des vorgestellten Schemas. Dabei wird die Sicherheitsmaßnahme „Zugriffskontrolle" referenziert, welche den Angriff „Unbefugter Zugriff" vermeiden soll. Wie in Abbildung 53 dargestellt, handelt es sich hierbei um eine vorbeugende Maßnahme.

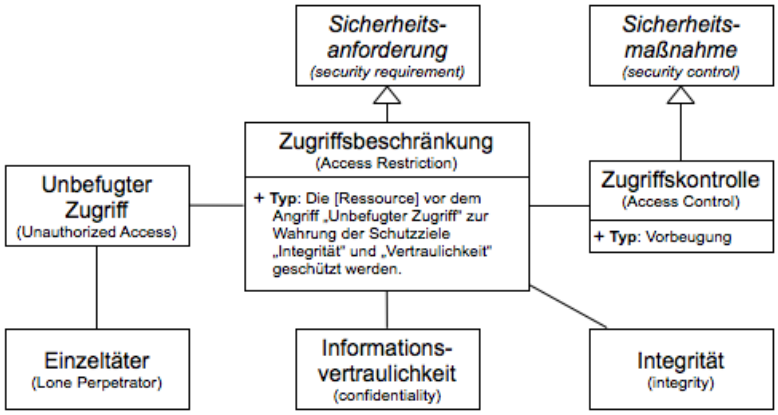

Abbildung 53: Beispiel-Sicherheitsanforderungsvorlage „Zugriffsbeschränkung"

5.3 Resümee

Dieses Kapitel behandelte die Unterstützung der Analyse von Sicherheitsanforderungen von Software-Systemen. Hierzu wurden Vorlagen für Sicherheitsanforderungen präsentiert, welche ein Werkzeug für fachliche Entwickler darstellen, um wiederverwendbare Sicherheitsanforderungen in der Anforderungsanalyse einzusetzen. Die Struktur der Vorlagen wurde zunächst durch ein Domänenmodell spezifiziert, welches aus drei Modulen besteht: einem Ressourcenmodul, welches den Bezug zur Ressourcen einer Fachdomäne herstellt, einem Bedrohungsmodul, welches die Kernkonzepte einer Bedrohungs- und Risikoanalyse definiert sowie einem Schutzmodul, welches die Konzepte der Ergebnisartefakte der Anforderungsanalyse beschreibt. Anhand des Domänenmodells wurde ein initialer Katalog von Sicherheitsanforderungsvorlagen beschrieben, welcher angepasst und erweitert werden kann. Die initialen Vorlagen basieren dabei auf existierenden Sicherheitsanforderungen.

Mit den vorgestellten Sicherheitsanforderungsvorlagen wird das Ziel verfolgt, eine Unterstützung für eine effiziente und strukturierte Bedrohungs- und Risikoanalyse bereitzustellen. Der Ansatz ist dabei zunächst unabhängig von einer konkreten Methode zur Anforderungsanalyse konzipiert. Durch das flexible Domänenmodell ist eine solche Anpassung möglich. In dieser Arbeit wird er jedoch hauptsächlich für die Aufbereitung von wiederverwendbaren Sicherheitsanforderungen in dem in Kapitel 4 vorgestellten Entwicklungsvorgehen verwendet. Durch die Sicherheitsanforderungsvorlagen werden bekannte und bewährte Sicherheitsanforderungen strukturiert aufbereitet und fachlichen Entwicklern bereitgestellt. Durch die Definition von abstrakten Konzepten im Domänenmodell wird die Praktikabilität des Prozesses zur Anforderungsanalyse erhöht.

Analog zu fachlichen funktionalen Anforderungen, stellen Sicherheitsanforderungen die essentielle Basis für die Spezifikation von Sicherheitsmaßnahmen dar. Sicherheitsmaßnahmen setzen dabei die Anforderungen um, wobei die Maßnahmen zunächst modelliert und anschließend auf konkrete Sicherheitstechnologien abgebildet werden. Die Festlegung von Sicherheitsanforderungen bestimmt somit den Umfang und die Art der einzusetzenden Sicherheitsmaßnahmen. Die Abbildung von Sicherheitsanforderungen auf Sicherheitsmaßnahmen wurde in diesem Kapitel bereits angesprochen. Eine detaillierte Auseinandersetzung mit diesem Thema liegt im Fokus des folgenden Kapitels.

6 Variabilitätsmodell für Sicherheitsmuster

Im Anschluss an die im vorangegangenen Kapitel beschriebene Spezifikation von Sicherheitsanforderungen in der Analysephase eines Software-Entwicklungsprozesses, müssen diese in der darauffolgenden Entwurfsphase auf Sicherheitsmaßnahmen abgebildet und iterativ verfeinert werden. Für den effizienten Entwurf von Sicherheitsmaßnahmen sieht das in Kapitel 4 beschriebene Vorgehen für sichere Software-Entwicklung den Einsatz von bewährten Lösungen in Form von Sicherheitsmustern vor.

Die Sicherheitsmuster abstrahieren dabei von den in einer Sicherheitsarchitektur vorhandenen Sicherheitsdiensten und -standards sowie den eingesetzten Sicherheitsrahmenwerken und -produkten. Für die nachvollziehbare Verwendung der Sicherheitsmuster in den Entwicklungsprozess ist eine Kategorisierung der Sicherheitsmuster notwendig, damit die Beziehungen zwischen den Mustern geklärt und diese gemeinsam eingesetzt werden können. Abbildung 54 zeigt die Einordnung des im folgenden Kapitel vorgestellten Beitrags zur Kategorisierung und Aufbereitung von Mustern in das sicherheitsbasierte Entwicklungsvorgehen auf.

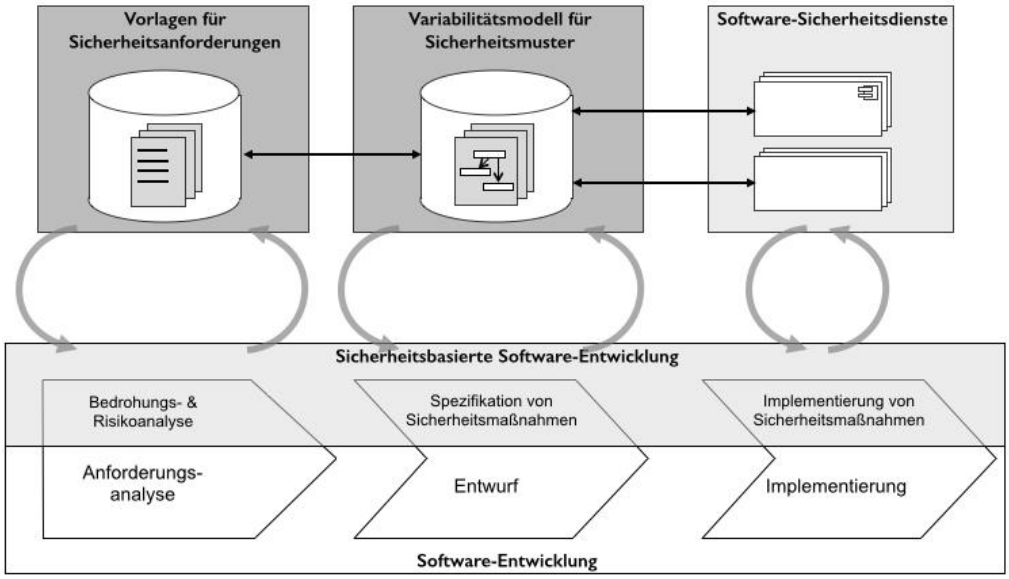

Abbildung 54: Sicherheitsmustersprachen im Kontext der Arbeit

Zum Zweck der Aufbereitung von Sicherheitsmustern wird in dem vorliegenden Kapitel zunächst ein Variabilitätsmodell vorgestellt, mittels dessen Sicherheitsmuster gemäß ihrer Spezifikation klassifiziert werden können. Insbesondere wird die Aufmerksamkeit in dieser Klassifikation auf die Berücksichtigung von hierarchisch angeordneten Abstraktionsstufen, den Beziehungen zwischen den Mustern sowie alternativen Auswahlmöglichkeiten zwischen mehreren Sicherheitsmustern gerichtet. Eine solche Einordnung ermöglicht eine strukturierte Verfeinerung von abstrakten Sicherheitsmaßnahmen zu einem technologienahen Feinentwurf unter Berücksichtigung der Sicherheitsanforderungen eines Software-Systems.

Des Weiteren wird ein Vorgehen zur Einordnung von Sicherheitsmustern mittels des Variabilitätsmodells aufgezeigt, wodurch die iterative Erfassung und Erweiterung von vorhandenen und benötigten Sicherheitsmaßnahmen einer Sicherheitsarchitektur unterstützt wird. Hierzu wird anhand von

ausgewählten Sicherheitstechnologien eine Sicherheitsarchitektur beschrieben, welche sich auf die Sicherheitsmaßnahmen Authentifizierung, Autorisierung und Zugriffskontrolle beschränkt.

6.1 Konzepte des Variabilitätsmodells

Wie in Kapitel 2.4.3 sowie in Kapitel 3 besprochen, stellen Sicherheitsmuster ein anerkanntes Hilfsmittel bei der Entwicklung und Modellierung von Sicherheitsmaßnahmen dar. Ein Grund hierfür ist die informelle Beschreibung von bewährten Lösungsansätzen zu Sicherheitsproblemen [HH+07a] [FP01] [DF+07]. In der Spezifikation von Sicherheitsmustern werden bereits deren Einsatzzweck, d.h. der Verwendungskontext des Musters und die zu lösende Problemstellung, sowie die eigentliche durch das Muster bereitgestellte Lösung und dessen Auswirkungen auf den Entwurf [GH+94] [SF+05] berücksichtigt. Durch die semantische Nähe zu Architektur- und Entwurfsmustern [So07] wird zudem die Integration in fachliche Entwurfsmodelle ermöglicht.

Für den effizienten Einsatz von Sicherheitsmustern beim Entwurf von Sicherheitsmaßnahmen ist eine angemessene Klassifikation notwendig, mit Hilfe derer die Verwendung von Mustern im Entwurfsprozess eindeutig spezifiziert wird. Ausgehend von den analysierten Sicherheitsanforderungen sollten unter Berücksichtigung von existierendem Sicherheitswissen nur die notwendigen Sicherheitsmaßnahmen für ein Software-System unter Einsatz der klassifizierten Sicherheitsmuster abgeleitet werden. Hierdurch wird ein ausgeglichenes Kosten-Nutzen-Verhältnis bei der Implementierung von Sicherheitsmaßnahmen ermöglicht.

Dabei werden in einem iterativen Vorgehen die Sicherheitsmaßnahmen von einem groben architekturellen Entwurf immer weiter konkretisiert, bis ein technologienaher Entwurf entsteht, welcher in der anschließenden Implementierungsphase mittels einer Sicherheitsplattform direkt umgesetzt werden kann. Die Klassifikation bietet dabei in jedem Iterationsschritt eine Unterstützung bei Entwurfsentscheidungen hinsichtlich der Auswahl und des Einsatzes der Sicherheitsmaßnahmen.

Kombinationen von Mustern ermöglichen es bereits, den Zusammenhang zwischen verschiedenen Mustern aufzuzeigen und deren kombinierten Einsatz zur Lösung komplexerer Probleme zu beschreiben [BH+07]. Jedoch fehlt hierzu ein formaler Ansatz zur Beschreibung der Kombinationen sowie eine Berücksichtigung bei der iterativen Verfeinerung der Sicherheitsmaßnahmen. Im Rahmen dieser Arbeit wird zu diesem Zweck der in Kapitel 2.3 vorgestellte Ansatz von Funktionsmodellen (engl. feature model, [KK+11] [CE02]) zur Klassifikation von Sicherheitswissen eingesetzt. Ausgehend von diesem Ansatz werden Kombinationen zwischen Mustern so beschrieben, dass Auswahlmöglichkeiten zwischen mehreren Sicherheitsmaßnahmen spezifiziert werden können.

Im Folgenden werden die zugrundeliegenden Konzepte des Variabilitätsmodells für Sicherheitsmuster vorgestellt. Hierzu wird zunächst auf die hierarchische Klassifizierung gemäß des Abstraktionsniveaus der Muster eingegangen. Anschließend wird die Grundstruktur der durch das Variabilitätsmodell beschriebenen Sicherheitsmustersprachen beschrieben. Abschließend werden die Beziehungen zwischen den einzelnen Mustern präsentiert.

6.1.1 Klassifizierung von Sicherheitsmustern

Der erste Schritt zur Aufbereitung des vorhandenen Sicherheitswissens besteht in der Klassifizierung der Sicherheitsmuster entsprechend ihres Abstraktionsniveaus sowie hinsichtlich ihres Einsatzes in einer mehrschichtigen Sicherheitsstrategie. Hierdurch wird eine Modularisierung der Sicherheitsmuster verfolgt, welches die Handhabung der Komplexität und des Umfangs der existierenden Sicherheitsmuster zum Ziel hat. Ebenso wird eine strukturierte Verwendung der Muster in der Entwurfsphase ermöglicht. Im Folgenden wird die zweidimensionale Klassifizierung vorgestellt,

welche die Muster anhand ihres Abstraktionsniveaus sowie anhand ihrer Verwendung in verschiedene Systemschichten einordnet.

Abstraktionsebenen

Ausgangspunkt für die Klassifikation der Sicherheitsmuster ist zunächst die Festlegung von Abstraktionsebenen, in denen die Muster eingeordnet werden können. Hierbei beschreibt die Einordnung in eine Abstraktionsebene die Nähe eines Sicherheitsmusters zu einer konkreten Sicherheitstechnologie wie zum Beispiel zu einem Sicherheitsstandard oder -produkt. Eine hohe Abstraktionsebene entspricht somit einer technologieunabhängigen Klassifizierung eines Sicherheitsmusters und weist im äußersten Fall eine direkte oder indirekte Beziehung zu technologieunabhängigen Sicherheitsanforderungen auf. Dagegen bezieht sich eine niedrige Abstraktionsebene auf eine oder mehrere Sicherheitstechnologien und somit auf eine konkrete technologische Sicherheitsplattform. Wie in Abbildung 55 dargestellt, ergibt sich durch diese Betrachtungsweise eine geschichtete Klassifikation der Sicherheitsmuster, in welcher Muster in einer hohen Abstraktionsebene bzw. einer oberen Schicht durch Sicherheitsmuster einer niedrigeren Abstraktionsebene bzw. einer unteren Schicht verfeinert werden.

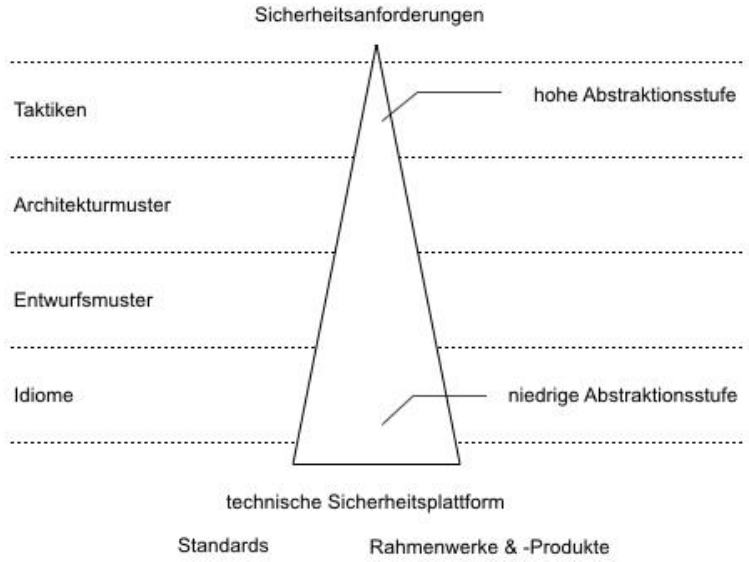

Abbildung 55: Abstraktionsebenen und Kategorien der Klassifikation

Die konkrete Anzahl der Abstraktionsebenen ist dabei an dieser Stelle nicht festgelegt, da sie zum einen abhängig von den jeweils betrachteten Sicherheitsmustern ist. Zum anderen zielt eine effiziente Kategorisierung der Muster darauf ab, die Anzahl der Abstraktionsebenen zu minimieren, so dass relativ schnell eine konkrete Umsetzung der Sicherheitslösung von einer abstrakten Lösung abgeleitet werden kann. Die Anzahl der Abstraktionsebenen ist auch abhängig vom eingesetzten Entwicklungsprozess und den darin vorgesehenen Verfeinerungsstufen der Modelle der fachlichen Funktionalität. Idealerweise sind die Abstraktionsstufen zur Entwicklung der Sicherheitsmaßnahmen mit den zur Entwicklung von fachlicher Funktionalität eingesetzten Abstraktionsebenen abgestimmt. Agile Entwicklungsprozesse, wie etwa Scrum, setzen dabei nur wenige Abstraktionsebenen zwischen abstraktem und implementierungsnahen Entwurf ein, während bei traditionellen Entwicklungsprozessen, wie etwa dem Rational Unified Process, deutlich mehr Zwischenschritte zu finden sind.

Kategorien

Wie in Abbildung 55 dargestellt, wird in dieser Arbeit eine Einteilung von vier generelle Abstraktionsebenen vorgenommen, welche jeweils eine unterschiedliche Anzahl von Unterebene umfassen können. Die Kategorisierung ist dabei nicht exakt, da aufgrund der informellen Beschreibung von Mustern die eindeutige Zuordnung zu einer Kategorie gemäß der folgenden Definitionen nicht immer möglich ist [BH+07:213ff]. Zudem ist auch die Verwendung jeder der aufgezeigten Kategorien nicht verpflichtend. Analog zu der Anzahl der Abstraktionsebenen können abhängig von dem jeweils betrachteten Sicherheitsmusters und des verwendeten Entwicklungsvorgehens Ebenen ausgelassen werden.

Zunächst befinden sich auf der höchsten Abstraktionsstufe Sicherheitslösungen, welche den Übergang von den Schutzzielen und den Sicherheitsanforderungen der Analysephase zu Sicherheitsmaßnahmen in der Entwurfsphase darstellen. Diese werden als Taktiken bezeichnet [FH06] und stellen erste Entwurfsentscheidungen zur Umsetzung eines Qualitätsattributes dar, welche aber noch nicht konkret genug für einen direkten Einsatz sind [BC+12]. Im Sinne der modellgetriebenen Software-Entwicklung (engl. Model Driven Software Development, MDSD [SV+07]) lassen sich Sicherheitsmuster auf dieser Ebene als unabhängig von einer Umsetzung mittels Informationssystemen (engl. computation independent) einordnen. Es bleibt auf dieser Ebene zunächst offen, ob eine Umsetzung mittels technischer Hilfsmittel erfolgt und zum Beispiel eine Zugriffskontrollmaßnahme durch einen menschlichen Akteur (Pförtner), ein physisches Objekt (Tür) oder durch eine Software-Komponente erfolgt.

Mittels Taktiken kann das generelle Vorgehen zur Lösung eines Problems festgelegt werden. Fægri et al. identifizieren drei wesentliche Taktiken zur Umsetzung einer Sicherheitsanforderung: Prävention, Detektion und Wiederherstellung [FH06]. Präventive Maßnahmen reduzieren dabei die Wahrscheinlichkeit bzw. das Risikos von Angriffen und Bedrohungen. Die Detektion von ungewünschten Ereignissen ermöglicht die Analyse von Angriffsmustern und daraufhin die Verbesserung der Sicherheitsmaßnahmen des zu schützenden Systems. Taktiken der Wiederherstellung umfassen alle Maßnahmen zur Reduzierung der Schadensauswirkungen eines Angriffes.

In der darauffolgenden Kategorie werden Taktiken durch Architekturmuster verfeinert. Diese betreffen Entwurfsentscheidungen, welche Einfluss auf die Struktur der gesamten Sicherheitsfunktionaltität eines Software-Systems haben [BR+99] [So07]. Im Rahmen von Sicherheitsmustern muss hierbei zudem zwischen strukturgebenden und funktionsgebenden Mustern unterschieden werden. Die strukturgebenden Architekturkomponenten bestimmen dabei die Komponenten einer Sicherheitsmaßnahme sowie ihre Beziehungen untereinander. Dagegen wird die Funktionalität der Sicherheitsmaßnahme zumeist in Form von Richtlinien, wie zum Beispiel Zugriffskontrollrichtlinien, spezifiziert. In dem vorliegenden Modell werden technische Umsetzungen fokussiert, weshalb auf dieser Abstraktionsebene plattformunabhängige Muster im Sinne der MDSD zu finden sind.

Ein verfeinerter Entwurf der Architekturkomponenten erfolgt durch die Sicherheitsmuster in der Entwurfsmuster-Kategorie. Analog zum fachlichen Entwurf umfasst der Feinentwurf auf struktureller Ebene die innere Strukturierung der Architekturkomponenten durch detailliertere Komponenten. Im Falle der Funktionalität werden hierbei präzisere Angaben zur Umsetzung der Funktionalität gemacht, beispielsweise durch konkretere Angaben zur Umsetzung einer Richtlinie mittels eines bekannten Sicherheitsmodells. Ein Beispiel hierfür ist die Konkretisierung einer rollenbasierten Zugriffskontrollrichtlinie in der Architekturmusterebene durch den Einsatz von hierarchischen Rollen in der Entwurfsmusterebene. Trotz der Konkretisierung sind die in dieser Kategorie eingeordneten Sicherheitsmaßnahmen nach wie vor als technologie- bzw. plattformunabhängig einzuordnen. Jedoch sind diese verfeinerten Maßnahmen darauf ausgelegt, durch Sicherheitsprodukte und -rahmenwerke umgesetzt zu werden.

Die Entwurfsmuster werden in der untersten Abstraktionsschicht durch Idiome weiter verfeinert. Diese stellen Sicherheitsmuster dar, welche inhärent in den verwendeten Sicherheitstechnologien eingebettet sind [BR+99]. Die Idiome abstrahieren zudem von der gesamten Komplexität der verwendeten Sicherheitsplattform, so dass nur die Informationen dargestellt werden, welche zur Umsetzung einer Sicherheitsmaßnahme benötigt werden. Im Sinne der MDSD enthält die Idiom-Kategorie somit plattformspezifische Muster.

Da in dieser Arbeit die Wiederverwendung von bestehenden Sicherheitsinfrastrukturen fokussiert wird, lässt sich die Ebene der Idiome in zwei weitere Ebenen unterteilen, wobei eine Schicht mit abstrakten Idiomen für Sicherheitsstandards oberhalb einer Schicht von technologischen Idiomen für Sicherheitsprodukte angeordnet ist. In einer Ebene für Sicherheitsstandards werden Idiome für die Umsetzung von abstrakteren Sicherheitsmustern mittels dafür vorgesehenen Standards spezifiziert, welche einen normierten Ansatz zur Umsetzung einer Sicherheitsmaßnahme beschreiben. Somit nimmt diese Schicht eine Mediatorrolle zwischen den technologieunabhängigen Sicherheitsmustern und den darunter liegenden technologiespezifischen Mustern ein. Durch diese Standardisierungsschicht wird ein Austausch von Sicherheitstechnologien sowie eine Interoperabilität zwischen verschiedenen Sicherheitstechnologien ermöglicht. Da jedoch zu vielen Sicherheitsbereichen keine standardisierten Lösungen existieren, ist diese Ebene im Falle von existierenden Standards zwar empfehlenswert, ansonsten jedoch lediglich als optional einzustufen.

Des Weiteren zählen bereits existierende Sicherheitsprodukte, -komponenten und -rahmenwerke zur Sicherheitsplattform. Aufgrund der Spezifik der Sicherheitsfunktionalität ist es sinnvoll, bereits erprobte Sicherheitsprodukte für die Umsetzung von Sicherheitsmaßnahmen einzusetzen [EB+07] [DE+09] [DA11]. In größeren Unternehmen besteht zudem eine über die Jahre gewachsene Sicherheitsinfrastruktur, welche für die Umsetzung von Sicherheitslösungen eingesetzt werden kann. Für eine Neuentwicklung sollte die technologische Sicherheitsplattform aus bestehenden Sicherheitsprodukten zusammengesetzt werden.

Systemschichten

Neben den horizontalen Schichten, welche die Muster anhand ihrer Abstraktion in Kategorien einordnet, können zudem mehrere vertikale Schichten unterschieden werden. Diese beschreiben die Ausprägung von Sicherheitsmaßnahmen entsprechend unterschiedlicher Systemschichten [Fe99] in [FP01]. Die in dieser Arbeit vorgenommene Unterteilung ist in Abbildung 56 dargestellt und orientiert sich an der Aufteilung von Fernandez et al. [FP01], welche die Systemschichten Netzwerk, Hardware, Verteilung, System und Anwendung vornehmen. Zusätzlich wird in dieser Arbeit noch eine Ausführungsumgebung berücksichtigt, da Software-Systeme zunehmend nicht mehr direkt auf einem Betriebssystem sondern mit Hilfe von Middleware-Rahmenwerken betrieben werden. Die Netzschicht beinhaltet die Kommunikation zwischen Rechnersystemen, während die Hardwareschicht Sicherheits-

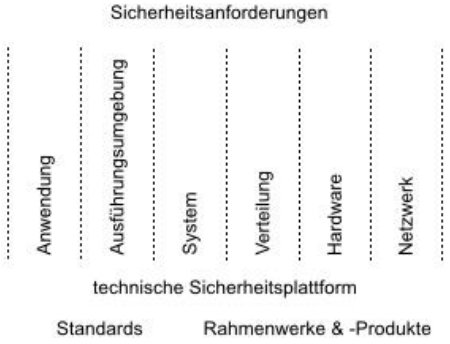

Abbildung 56: Klassifizierung nach Systemschichten

muster für den Schutz von Rechnersysteme an sich betrachtet. Die Systemschicht bezieht sich auf die Systemdienste, welche auf den Rechnersystemen aufbauen, wie zum Beispiel Betriebssysteme, Datenbank-Managementsysteme (DBMS), etc. Die Anwendungsebene betrifft Objekte eines bestimmten auf den Systemdiensten aufsetzenden Software-Systems [FP01].

Diese recht grobe Unterteilung zeigt, welche verschiedenen Systemschichten berücksichtigt werden müssen, um eine mehrschichtige Sicherheit für ein Software-System zu implementieren. Dies wird in dem besprochenen Variabilitätsmodell durch die isolierte Betrachtung der Sicherheitsmuster für die verschiedenen Systemschichten berücksichtigt. Dabei dienen die zuvor besprochenen abstrakten Sicherheitstaktiken jedoch weiterhin als gemeinsame Grundlage für alle Systemschichten. Ist die Verfeinerung der Sicherheitsmuster durch weitere Spezialisierungen weiterhin unabhängig von einer bestimmten Systemschicht, so können diese abstrakten Sicherheitsmuster als generische Lösungen für unterschiedliche Systemschichten dienen. Werden in den sich anschließenden Abstraktionsebenen der Einsatz von Sicherheitsmuster für unterschiedliche Systemebenen identifiziert, so werden diese unabhängig voneinander betrachtet und weiter verfeinert. Zu einer abstrakten Sicherheitslösung wird in diesem Fall für jede betrachtete Systemschicht ein einzelner Musterbaum verwendet.

Die vorgegebene Sicherheitsplattform schränkt zudem die Auswahl von technologieunabhängigen Sicherheitsmustern und deren alternative Ausprägungen im Variabilitätsmodell einer Sicherheitsmaßnahme ein. Aufgrund der vorhandenen Funktionalität der gewählten Sicherheitsplattform werden nicht alle theoretisch möglichen Sicherheitsmuster im Variabilitätsmodell beschrieben. Stattdessen werden die Sicherheitsmuster auf die Möglichkeiten reduziert, welche von der zugrundeliegenden Plattform unterstützt werden. Hierdurch wird das Referenzmodell einer Sicherheitsarchitektur reduziert und somit die Auswahl von Mustern für den Entwickler auf ein übersichtliches Maß gekürzt. Zudem werden nur solche Sicherheitsmaßnahmen ausgewählt, welche mittels der gewählten Sicherheitsplattform auch umgesetzt werden können. Um den Umfang der durch ein Sicherheitsprodukt unterstützten Sicherheitsmuster zu spezifizieren und dazu nötige Idiome festzuhalten, ist es notwendig, eine Analyse der verfügbaren Funktionen durchzuführen [DS+12], was in dieser Arbeit jedoch nicht weiter vertieft wird.

6.1.2 Sicherheitsmustersprachen

Nach der Klassifizierung der Sicherheitsmuster wird in diesem Abschnitt die kategorieübergreifende Gruppierung sowie Anordnung von Sicherheitsmustern erörtert und dadurch die Grundstruktur des Variabilitätsmodells definiert. Hierbei werden die Sicherheitsmuster hinsichtlich ihres gemeinsamen Einsatzes zur Lösung von komplexen Sicherheitsproblemen betrachtet. Aufgrund der in Kapitel 2 gegebenen Definition können in diesem Fall diese Musterkombinationen als Mustersprachen bezeichnet werden [BH+07].

Das Ziel der Anordnung liegt dabei in der Unterstützung von Entwicklern bei der iterativen Verfeinerung der Sicherheitsmuster sowie der Auswahl von Sicherheitsmaßnahmen hinsichtlich eines Sicherheitsproblems, was zu einer Hierarchie innerhalb der Anordnung von Sicherheitsmustern führt. Somit beschäftigt sich dieser Beitrag mit den im Anforderungskatalog in Kapitel 3.1 aufgeführten Anforderungen der Wiederverwendbarkeit und Nachvollziehbarkeit. Angesichts der Klassifizierung werden in dieser Arbeit zwei Arten von Sicherheitsmustersprachen unterschieden, welche die Zusammenhänge innerhalb einer Abstraktionsschicht sowie schichtübergreifende Zusammenhänge betrachten. Durch die Aufteilung werden Entwickler über die Beziehungen zwischen Sicherheitsmustern informiert und erhalten eine Entscheidungsgrundlage für deren Auswahl.

Durch die Abstraktionsebenen entsteht eine Hierarchie zwischen den Sicherheitsmustern, wobei ein Sicherheitsmuster einer höheren Abstraktionsebene einem Sicherheitsmuster einer niederen Ebene in

der Hierarchie vorangestellt ist. Dieser Zusammenhang ist in Abbildung 57 dargestellt. In der Traversierung der Hierarchie ist die wichtigste Information für die iterative Verfeinerung die Auswahl zwischen mehreren Sicherheitsmustern, welche unterschiedliche alternative Möglichkeiten zur Umsetzung eines abstrakten Sicherheitsmusters darstellen. Unter dieser Voraussetzung führt eine Anordnung der Sicherheitsmuster entlang der Abstraktionsstufen zu einer Baumstruktur. Die alternativen Verfeinerungen eines Sicherheitsmusters der Abstraktionsebene N stellen dabei die Kindknoten in der darunter liegenden Abstraktionsebene N+1 dar. Die in die Taktikkategorie eingeordneten Sicherheitsmuster stellen die Wurzeln des Baumes dar, welcher durch die die Taktik umsetzenden Sicherheitsmuster definiert wird. Dagegen werden in den Blättern eines Sicherheits-musterbaumes die Idiome von Sicherheitsstandards bzw. -produkten notiert.

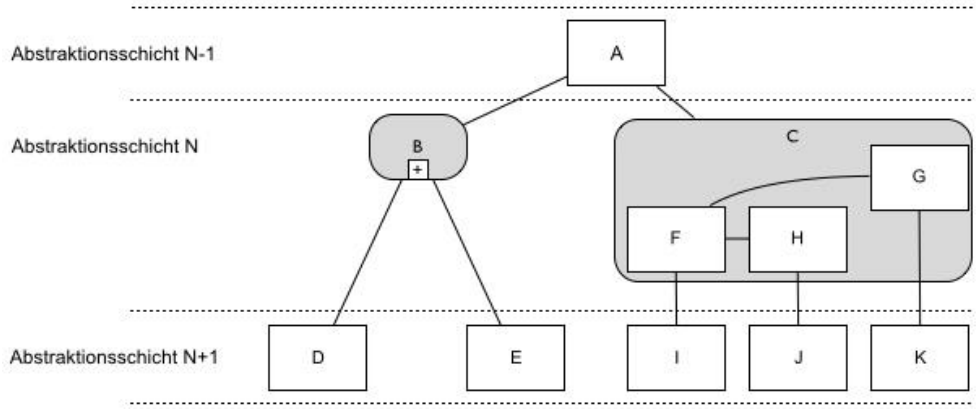

Abbildung 57: Hierarchie von Sicherheitsmustern

Des Weiteren ist die Information über die Umsetzung eines abstrakteren Sicherheitsmusters mittels mehrerer konkreter Sicherheitsmuster relevant. Die Umsetzungen zu einem Sicherheitsmuster weisen zumeist komplexe Beziehungen untereinander auf, welche im allgemeinen Fall zu der Struktur eines ungerichteten Graphen führen. Somit können diese Beziehungen nicht durch die zuvor festgelegte Baumstruktur abgebildet werden. Zur Einhaltung der Baumstruktur wird daher zunächst der Kind-knoten eines Elternknoten als Container aufgefasst, in welchem die spezialisierten Muster und deren Assoziation untereinander spezifiziert werden. Zur besseren Darstellung lassen sich die verfeinerten Muster sowie die Beziehungen ausblenden. Die verfeinerten Sicherheitsmuster stellen selbst wieder Elternknoten für die sie verfeinernden Kindknoten dar. In der kompakten Ansicht werden diese Verfeinerungsbeziehungen zwischen den Mustern in einem Containerknoten und deren Kindknoten eingezeichnet.

Durch den Containeransatz enthält jeder Containerknoten in einem Mustersprachenbaum eine Mustersprache, welche eine Lösung für ein komplexeres Sicherheitsproblem mittels der Verknüpfung von atomaren Sicherheitsmustern beschreibt. Die Muster einer solchen Mustersprache beschreiben dabei die Sicherheitslösung auf einem einheitlichen Abstraktionsniveau. Somit wird keine Spezialisierung der Sicherheitslösung vorgenommen, sondern die Relation zu weiteren Sicherheitsmustern der gleichen Abstraktionsebene spezifiziert. Daher wird für eine solche Mustersprache die Bezeichnung horizontale Mustersprache (HMS, engl. horizontal pattern language, HPL) eingeführt. Im einfachsten Fall besteht eine horizontale Mustersprache aus einem einzigen Muster. Im allgemeinen Fall handelt es sich um eine Graphenstruktur $G_{HMS,N} = \{V_{HMS,N}, E_{HMS,N}\}$, in welcher jeder Knoten $v_N \in V_{HMS,N}$ ein Muster auf dem Abstraktionsniveau N repräsentiert. Die Kanten

$E_{HMS,N} = \{(x,y) \mid x,y \in V_{HMS,N}\}$ stellen die Assoziationen zwischen den Mustern dar und können unterschiedliche Semantik beinhalten.

Die Verfeinerung eines Musters bzw. einer Mustersprache entspricht einem Kindknoten in dem Mustersprachenbaum. In diesem wird das Muster somit auf einem detaillierteren Abstraktionsniveau dargestellt. Die Gesamtheit aller Verfeinerungen eines Musters entlang des Mustersprachenbaumes wird in dieser Arbeit als vertikale Mustersprache (VMS, engl. vertical pattern language, VPL) bezeichnet. Eine vertikale Mustersprache für ein abstraktes Muster p ist somit ein Baum $T_{VMS,p} = \{V_{VMS,p}, E_{VMS,p} \mid p \in G_{HMS,0}\}$. Die einzelnen Knoten $v \in V_{VMS,p}$ beinhalten die möglichen, alternativen Verfeinerungen des Musters. Die Wurzel des Baumes ist somit ein Muster in einer horizontalen Mustersprache auf dem höchsten Abstraktionsniveau, wogegen die Blätter der vertikalen Mustersprache eine Implementierung des Musters bzw. Idiom [BR+99] für eine bestimmte technische Plattform repräsentieren. Im Falle von Sicherheitsmustern besteht eine solche Plattform aus Sicherheitsstandards sowie aus zugehörigen Sicherheitsprodukten, -komponenten und -rahmenwerken.

6.1.3 Beziehungen zwischen Mustern

Die Beziehungen unter den Elementen einer Mustersprache lassen sich entsprechend der horizontalen und vertikalen Mustersprachen in zwei Kategorien aufteilen. Innerhalb einer HMS werden hauptsächlich Beziehungen eingesetzt, welche die Stellung der Muster zueinander innerhalb der gleichen Abstraktionsebene spezifizieren. Zu diesem Zweck werden Nutzungs- und Abhängigkeitsbeziehungen sowie Ausschlussbeziehungen eingesetzt. Hierdurch werden Entwicklern die Abhängigkeiten zu anderen Sicherheitsmustern aufgezeigt, welche bei dem Einsatz eines Sicherheitsmusters zu berücksichtigen sind. Somit können zum Beispiel die entstehenden Entwicklungskosten einer Sicherheitsmaßnahme schnell überblickt werden.

Dagegen wird die Beziehung zwischen zwei Elementen einer VMS aufgrund der unterschiedlichen Abstraktionsebenen durch die Verwendung von Verfeinerungs- und Spezialisierungsbeziehungen beschrieben. Durch die Berücksichtigung von Gemeinsamkeiten und Varianten werden optionale und verpflichtende Beziehungen in beiden Mustersprachentypen eingesetzt. Dies ermöglicht Entwicklern die zielgerichtete Auswahl zwischen unterschiedlichen Sicherheitsmustern bei der Verfeinerung für eine angemessene Umsetzung der Sicherheitsanforderungen des Software-Systems.

Spezialisierungsbeziehung

Die Spezialisierungsbeziehung ist die grundlegende Relation zwischen Mustern verschiedener Abstraktionsstufen in einer VMS. Sie drückt eine Verfeinerung oder Implementierung eines abstrakteren Musters durch ein oder mehrere konkretere Muster aus. Eine Verfeinerung- bzw. Spezialisierungsbeziehung wird verwendet, wenn ein Muster durch zusätzliche Details angereichert wird. Ein Beispiel hierfür ist die Ausprägung des allgemeinen Autorisierungsmusters durch rollenbasierte oder attributbasierte Zugriffskontrollmodelle [FP01] [SC+96] [SF+05:252].

In einem Diagramm wird die Spezialisierungsbeziehung durch eine gestrichelte Linie zwischen dem abstrakten Sicherheitsmuster und den verfeinerten Mustern dargestellt (vgl. Abbildung 59). Dabei wird die Konvention eingeführt, dass das abstraktere Muster oberhalb der konkreteren Muster platziert wird, wodurch die Richtung der Beziehung im Musterbaum verdeutlicht wird. Wird eine andere Darstellung verwendet, so kann ein offener Pfeil eingesetzt werden, welcher in die Richtung der Spezialisierung zeigt und somit die Richtung des Pfeils vom abstrakten zum verfeinerten Muster geht. Ist der Kontext nicht klar, so kann auch der Bezeichner „«realisiert durch»“ (engl. «realizes») bzw. „«implementiert durch»” (engl. «implements») zusätzlich an der Linie platziert werden. Die Assoziation eines abstrakten Musters mit mehreren konkreten Mustern mittels der

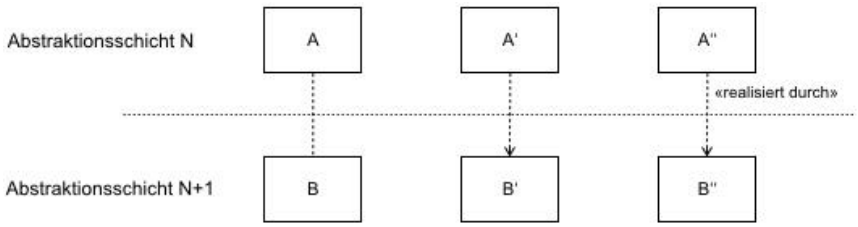

Abbildung 59: Alternative Modellierungsnotationen für Spezialisierungsbeziehungen

Spezialisierungsbeziehung entspricht semantisch einer Und-Beziehung, wodurch alle konkreteren Muster notwendig sind, um das abstraktere Muster umzusetzen.

In der Spezialisierungsbeziehung zwischen Mustern können Variabilität und Gemeinsamkeiten ausgedrückt werden. Dabei stellen variable Spezialisierungen optionale oder alternative Verfeinerungen eines Musters dar. Im ersten Fall wird dabei die Sicherheitsfunktionalität der konkreteren Sicherheitsmuster unter Berücksichtigung von zusätzlichen optionalen Sicherheitsmaßnahmen um ergänzende Sicherheitsfunktionalität erweitert. Diese sind aber für die Umsetzung des abstrakteren Musters nicht zwingend notwendig. Eine solche optionale Verfeinerung wird, wie in Abbildung 58 dargestellt, durch einen nicht ausgefüllten Kreis am Ende der Spezialisierungsbeziehung visualisiert.

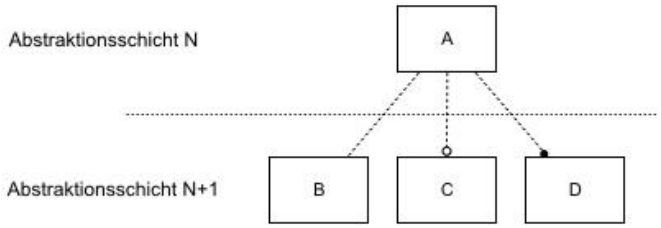

Abbildung 58: Optionale und obligatorische Spezialisierung

Im Falle von alternativen Mustern hingegen stellen zwei oder mehr Muster eine gleichwertige Umsetzung eines abstrakteren Sicherheitsmusters dar. So kann zum Beispiel das abstrakte Autorisierungsmuster mittels der Alternativen rollenbasierte oder attributbasierte Zugriffskontrolle verfeinert werden. Hierbei können zwei unterschiedliche Arten der Auswahl der Alternativen gewählt werden. Zum einen muss zur Spezialisierung aus einer Gruppe von alternativen Sicherheitsmustern eine Wahlmöglichkeit ausgewählt werden. Dies entspricht einer Exklusiv-Oder-Assoziation und wird durch eine Gruppierung der Assoziationen mittels eines nicht ausgefüllten Dreiecks visualisiert (vgl. Abbildung 60). Zum anderen ist die Auswahl und Kombination von zwei oder mehr Alternativen zur Umsetzung möglich. Diese Oder-Beziehung wird mittels einer Gruppierung der Assoziationen innerhalb eines ausgefüllten Dreiecks dargestellt.

Im Gegensatz zu variablen Sicherheitsmustern drücken Gemeinsamkeiten aus, dass Software-Systeme, deren Sicherheitsfunktionalität mittels eines Musterbaumes entwickelt wird, dasselbe Muster zur Implementierung eines Sicherheitsproblems einsetzen. Dies ist insbesondere dann der Fall, wenn eine existierende Sicherheitsinfrastruktur eingesetzt wird und somit auf vorhandene Sicherheitstechnologien und -richtlinien aufgesetzt wird. Das einfachste Vorgehen zur Modellierung von gemeinsamen Sicherheitsfunktionen besteht in der Vernachlässigung von alternativen und optionalen Sicherheitsmustern im Musterbaum. Hierdurch werden die Auswahlmöglichkeiten eingeschränkt bzw. unterbunden, so dass nur die modellierten Sicherheitsmuster verwendet werden können. Sollen

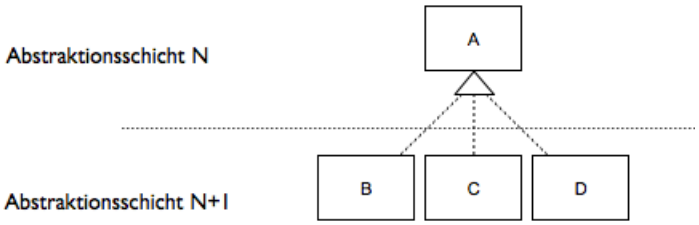

Abbildung 60: Exklusiv-Oder-Spezialisierung

gemeinsame Muster bei der Spezialisierungsbeziehung explizit visualisiert werden, wird dies durch einen ausgefüllten Kreis am Ende der Spezialisierungsbeziehung veranschaulicht (vgl. Abbildung 58).

Nutzungsbeziehungen

Im Gegensatz zu vertikalen Mustersprachen werden in horizontalen Mustersprachen Nutzungsbeziehungen eingesetzt, um die Abhängigkeit von mehreren Mustern der gleichen Abstraktionsebene zu spezifizieren. Ein Muster einer horizontalen Mustersprache kann einerseits einen in sich abgeschlossenen Lösungsansatz für ein Sicherheitsproblem darstellen. Andererseits können für die Umsetzung eines Musters verschiedene weitere Muster hinzugezogen werden, um die Funktionalität des abstrakteren Sicherheitsmusters zu ergänzen. In diesem Fall werden Nutzungsbeziehungen verwendet, um dieses Zusammenspiel zu verdeutlichen. Eine Nutzungsbeziehung wird durch einen durchgezogenen gerichteten Pfeil vom nutzenden zum genutzten Muster dargestellt. Analog zur Spezialisierungsbeziehung lassen sich dabei ebenfalls variable Nutzungsbeziehungen darstellen, welche analog zur Spezialisierungsassoziation modelliert werden.

So kann beispielsweise die Verwendung von einem oder mehreren weiteren Mustern die Fähigkeiten eines Musters erweitern, sofern dies für die Lösung eines gegebenen Sicherheitsproblems hilfreich ist. Ein Beispiel hierfür ist die Erweiterung einer nutzerbestimmten Autorisierung durch eine systembestimmte Autorisierung, um unzulässige Informationsflüsse in einem Software-System zu unterbinden. In diesem Fall ist der Einsatz von weiteren Mustern optional und nicht zwingend erforderlich. Eine solche optionale Musterverwendung wird grafisch anhand einer durchgezogenen Linie dargestellt, welche mittels eines nicht ausgefüllten Kreises abgeschlossen wird. Ist der Kontext nicht klar, so kann optional die Bezeichnung «benutzt optional» verwendet werden. Diese Darstellung wird in Abbildung 61 wiedergegeben.

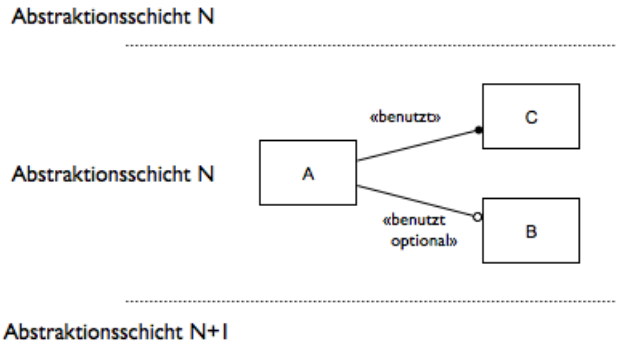

Abbildung 61: Optionale und obligatorische Nutzungsbeziehung

Neben optionalen Mustererweiterungen lassen sich auch Auswahlmöglichkeiten für obligatorisch genutzte Muster modellieren. Analog zur Spezialisierungsbeziehung lassen sich hierbei ebenfalls eine Oder- und eine Exklusiv-Oder-Beziehung unterscheiden. Eine Oder-Beziehung kann eingesetzt

werden, wenn aus einer Menge von Mustern ein oder mehrere Muster zur Umsetzung gewählt werden kann. Dies wird entsprechend mit einer Gruppierung der Assoziationslinien durch ein ausgefülltes Dreieck repräsentiert, wie in Abbildung 62 dargestellt wird. Dagegen wird eine Exklusiv-Oder-Assoziation verwendet, wenn genau eine Alternative aus einer Menge von Sicherheitsmaßnahmen zur Nutzung ausgewählt werden muss. Entsprechend wird die Gruppierung in diesem Fall durch ein nicht ausgefülltes Dreieck begrenzt.

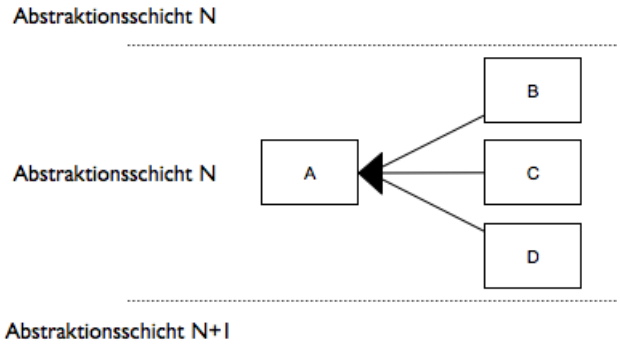

Abbildung 62: Oder-Nutzungsbeziehung

Neben der direkten Nutzung von Sicherheitsmustern zur strukturellen Umsetzung einer Sicherheitsmaßnahme, kann zudem auch die indirekte Verwendung von Sicherheitsmustern modelliert werden. In diesem Fall wird der Einsatz von unabhängigen Sicherheitsmustern, zum Beispiel in einer anderen Sicherheitsmustersprache, vorausgesetzt, um die Vollständigkeit der Sicherheitsmaßnahme zu ermöglichen. Beispielsweise nutzt das abstrakte Muster Zugriffskontrolle die Muster Authentifizierung und Autorisierung nicht direkt. Jedoch werden diese für den Einsatz des Zugriffskontroll-Musters vorausgesetzt, da sonst die Identität des Subjektes nicht bekannt bzw. die Rechtevergabe an das Subjekt ungeklärt ist. In diesem Fall wird die Nutzungsbeziehung zu einer Abhängigkeitsbeziehung, welche durch einen mit der Annotation „«setzt voraus»“ versehenen gestrichelten Pfeil dargestellt wird (vgl. Abbildung 63).

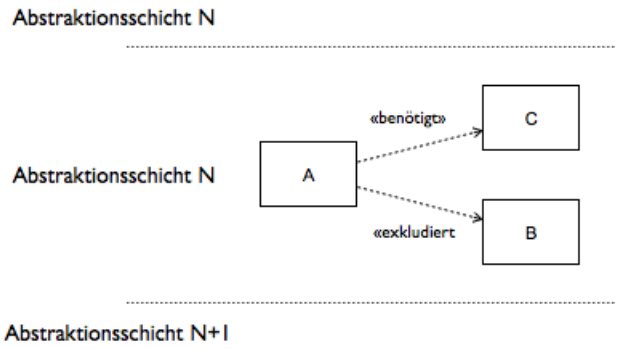

Abbildung 63: Abhängigkeits- und Ausschluss-Beziehung

Eine Exklusion-Beziehung wird eingesetzt, um die Anwendung eines weiteren Musters zu untersagen. Dies wird in Fällen nötig, in welchen der Einsatz zweier oder mehr Muster eine gegenläufige Wirkung auf die Sicherheit eines Systems hat. Auch wenn der Gebrauch eines Sicherheitsmusters die Funktionalität eines anderen Sicherheitsmusters einschränkt bzw. annulliert, wird dies mittels einer Exklusion-Beziehung im Modell spezifiziert. Der Ausschluss von Sicherheitsmustern, welche als alleinige Alternativen bei der Spezialisierung oder bei der Benutzung eines weiteren

Sicherheitsmusters vorgesehen sind, ist durch die Modellierung mittels einer Alternative-Beziehung bereits implizit modelliert.

6.2 Erstellung von Sicherheitsmustersprachen

Unter Einsatz des zuvor spezifizierten Variabilitätsmodells wird in diesem Abschnitt die Erstellung und Erweiterung von horizontalen und vertikalen Mustersprachen für Sicherheitsmaßnahmen erläutert. Die Erstellung von Sicherheitsmustersprachen ist dabei essentielle Voraussetzung für die Entwurfsphase gemäß des in Kapitel 4 beschriebenen sicherheitsbasierten Entwicklungsvorgehens. Während der Erstellung bestehen die wesentlichen Aufgaben in der Klassifizierung von bestehenden Sicherheitsmustern sowie der Festlegung von Beziehungen zwischen den Mustern. Ziel der Aufbereitung ist die Unterstützung der Abbildung von Sicherheitsanforderungen auf konkrete Sicherheitstechnologien mittels schrittweise verfeinerten Entwurfsmodellen. Nach der initialen Erstellung bestehen die weiteren Aufgaben in der Pflege und der damit verbundenen Erweiterung der Sicherheitsmustersprachen, um neue Sicherheitsanforderungen sowie neue Sicherheitstechnologien einfließen zu lassen. Im Folgenden wird eine Übersicht über die wesentlichen Schritte der Erstellung eines Sicherheitsmusterbaumes gegeben. Anschließend werden die Schritte praktisch anhand der Aufbereitung von Sicherheitsmaßnahmen für Authentifizierung, Autorisierung und Zugriffskontrolle demonstriert.

6.2.1 Initiale Erstellung

Zur Erstellung eines Mustersprachenbaumes für eine ausgewählte Sicherheitsmaßnahme wird in dieser Arbeit ein Middle-Out-Ansatz vorgeschlagen. Hierdurch wird gewährleistet, dass eine Abbildung zwischen einer abstrakten Sicherheitsmaßnahme und einer Sicherheitstechnologie existiert. Im Falle eines reinen Top-Down- oder Bottom-Up-Vorgehens kann diese nicht gewährleistet werden, wodurch das in Kapitel 4 vorgestellte Referenzmodell für Sicherheitsarchitekturen nicht erfüllt ist. Somit wird ausgehend von einem abstrakten Sicherheitsmuster eine Menge an möglichen technologischen Umsetzungen ausgewählt. Diese Menge wird in anschließenden Verfeinerungen des abstrakten Sicherheitsmusters eingeschränkt. Zeitgleich wird eine Menge von abstrakten Sicherheitsmustern ausgewählt, für welche ein technologisches Sicherheitsmuster eine mögliche Umsetzung darstellt. Abbildung 64 zeigt eine Übersicht über dieses Vorgehen.

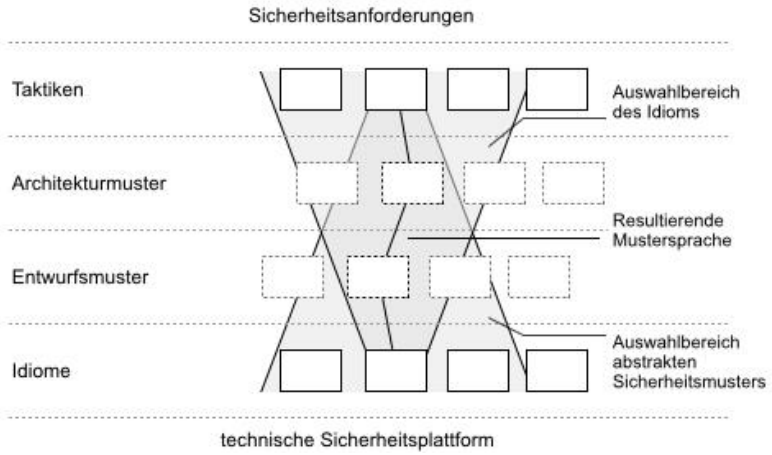

Abbildung 64: Auswahlprozess für die Verknüpfung von Sicherheitsmuster

Spezialisierung von abstrakten Sicherheitsmustern

Im Detail wird ausgehend von der Aufbereitung einer abstrakten Sicherheitsmaßnahme, zunächst eine Menge an Sicherheitsmustern ausgewählt, welche die Sicherheitsmaßnahme beschreiben. Die Auswahl der Sicherheitsmaßnahme wird zum einen aus den Sicherheitsanforderungen, welche im Vorlagenkatalog definiert sind oder für ein Software-System in der Analysephase spezifiziert wurden, sowie zum anderen durch die in Sicherheitsmusterkatalogen definierten Lösungsmöglichkeiten beeinflusst. Hierdurch wird sichergestellt, dass die ausgewählte Sicherheitsmaßnahme in dem Referenzmodell der Sicherheitsarchitektur verankert ist und ein Bedarf zur Umsetzung der Sicherheitsmaßnahme besteht. Die Aktivitäten sind in Abbildung 65 im oberen Teil dargestellt.

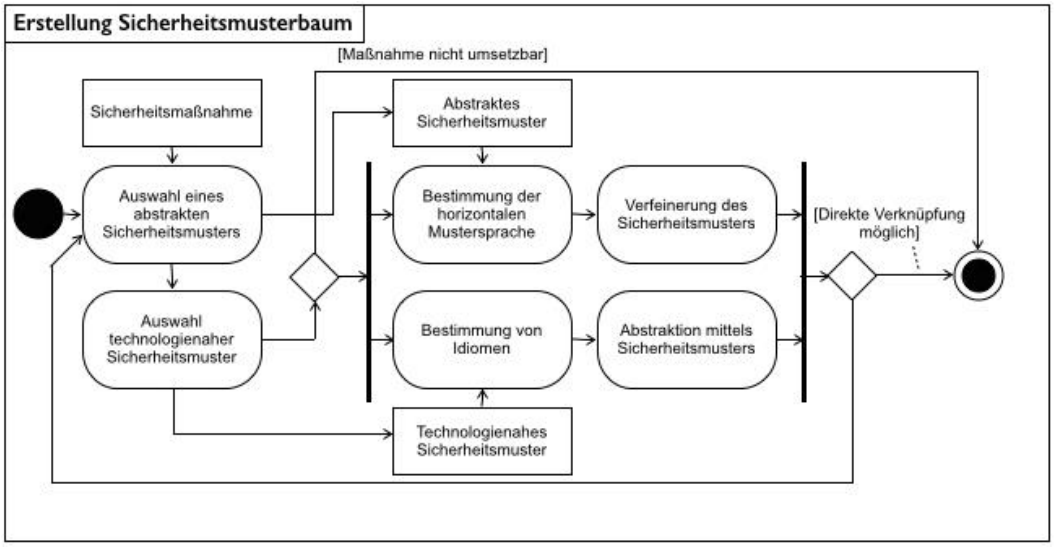

Abbildung 65: Erstellungsprozess für Sicherheitsmusterbäume

Ausgehend von dem Variabilitätsmodell wird dabei ein Sicherheitsmuster ausgewählt, welches zunächst in der obersten Schicht eingeordnet werden kann. Somit kann davon ausgegangen werden, dass durch das Sicherheitsmuster eine grobe Architekturlösung vorgegeben wird oder ein generelles Verfahren zur Lösung des Sicherheitsproblems definiert wird. Dieses Sicherheitsmuster wird zusätzlich einer der drei besprochenen Taktiken zugeordnet und die Verknüpfung zu ein oder mehreren Sicherheitsanforderungen hergestellt, so dass die Überführung aus der Anforderungsanalyse in den Entwurf ermöglicht wird.

Ist eine Sicherheitsinfrastruktur vorhanden, so wird diese hinsichtlich der Umsetzung des abstrakten Sicherheitsmusters durch ein Idiom einer eingesetzten Sicherheitstechnologie untersucht. Ist keine Sicherheitsinfrastruktur vorhanden bzw. ist kein eingesetztes Sicherheitsprodukt der Infrastruktur in der Lage die Maßnahme umzusetzen, so muss eine Entscheidung über die Verwendung der Maßnahme getroffen werden. Eine Möglichkeit besteht in der Auswahl an möglichen externen Produkten, welche die Maßnahme bzw. das Muster umsetzen könnten und somit die Sicherheitsinfrastruktur erweitern. Bei der Auswahl von externen Sicherheitsprodukten wird diese Menge bei weiterer Verfeinerung des Sicherheitsmusters gefiltert. Zur Analyse der Umsetzung des Musters ist Expertenwissen über die eingesetzten Sicherheitsprodukte notwendig, weshalb diese Aktivität von Sicherheitsexperten durchgeführt werden sollte. Dieses Vorgehen ist insbesondere für Maßnahmen erforderlich, welche eine hohe Priorität aufweisen und beispielsweise in vielen Software-Systemen zum Einsatz kommen. Andernfalls wird das Muster nicht weiter verfeinert und eine andere Maßnahme wird zur Umsetzung einer Sicherheitsanforderung ausgewählt.

Anschließend wird das ausgewählte Sicherheitsmuster hinsichtlich der Abhängigkeit zu weiteren Sicherheitsmustern auf dem gleichen Abstraktionsniveau analysiert. Hierdurch wird die zugehörige horizontale Sicherheitsmustersprache gebildet und optionale Assoziationen sowie verpflichtende Abhängigkeiten geklärt. Sind bereits Sicherheitsmustersprachen vorhanden, in welche das ausgewählte Sicherheitsmuster eingeordnet werden kann, so wird der Bezug zu diesen Sicherheitsmustern geklärt. Bei der Ersterstellung eines Musterbaumes kann es jedoch vorkommen, dass das Muster zunächst unabhängig von anderen Mustern bleiben und die horizontale Mustersprache erst in der Aufbereitung von weiteren Sicherheitsmaßnahmen gebildet wird.

Des Weiteren wird das Sicherheitsmuster auf die Umsetzung mittels spezialisierter Sicherheitsmuster untersucht. Das Ziel hierbei ist, die Abbildung des abstrakteren Sicherheitsmusters auf die Sicherheitstechnologie zu konkretisieren, in dem zunächst die Bestandteile des Sicherheitsmusters identifiziert werden. Anschließend werden die Einzelteile des abstrakten Sicherheitsmusters auf spezialisierte Sicherheitsmuster abgebildet. Auf diese Art und Weise wird eine vertikale Mustersprache für die Sicherheitsmaßnahme aufgebaut und somit das abstrakte Sicherheitsmuster auf technologienahe Muster abgebildet.

Abstraktion von Idiomen

Zur Abbildung von technologienahen Mustern bzw. Idiomen auf abstrakte Sicherheitsmuster, wird entsprechend der entgegengesetzte Weg eingeschlagen. Zunächst werden Idiome in der Anwendung der Sicherheitsprodukte identifiziert. Diese beschreiben typische und bewährte Einsatzpraktiken, welche durch den Hersteller vorgegeben werden oder sich durch lange Nutzungszeiten als praktikabel erwiesen haben. Davon ausgehend werden abstrakte Sicherheitsmuster zugeordnet, wodurch die wesentliche Zuteilung der Idiome zu einem Sicherheitsmusterbaum durchgeführt wird. Diese Aktivitäten für diesen Zuordnungsprozess sind in Abbildung 65 im unteren Teil dargestellt.

Anschließend werden innerhalb des Musterbaumes Sicherheitsmuster untersucht, welche auf einem niedrigen Abstraktionsniveau die bewährten Methoden der Sicherheitsprodukte technologieunabhängig beschreiben. Hierbei ist darauf zu achten, dass gegebenenfalls die bewährten Einsatzpraktiken nicht durch Sicherheitsmuster abgebildet werden können, da proprietäre Eigenentwicklungen durchgeführt wurden. In diesem Fall ist zu prüfen, ob die Eigenentwicklungen gerechtfertigt sind, da die zu lösende Sicherheitsproblematik zuvor generell nicht behandelt wurde. Gegebenenfalls eignen sich die Eigenentwicklungen zur Aufbereitung als technologieunabhängiges Sicherheitsmuster. Andernfalls ist die Überarbeitung der bestehenden Sicherheitslösung und der Einsatz von bekannten Sicherheitsmustern zu präferieren.

Durch die abwechselnde Verfeinerung von abstrakten Sicherheitsmustern sowie die Generalisierung von technologienahen Sicherheitsmustern wird eine Annäherung und Verknüpfung der verschiedenen Abstraktionsstufen erreicht. Zur Identifizierung von passenden Sicherheitsmustern sind dabei bestehende Musterkataloge, wie zum Beispiel [SF+05], einzusetzen. Für die Einordnung in passende Abstraktionskategorien lassen sich die zuvor beschriebenen Abstraktionsschichten einsetzen. Bei der Untersuchung der Abhängigkeiten zwischen den Mustern ist ebenfalls die Art der Beziehung zu berücksichtigen. Es muss entschieden werden, ob ein Muster in einer obligatorischen oder optionalen Beziehung zu anderen Mustern steht. Ebenso können Alternativen zu einem existierenden Sicherheitsmuster identifiziert werden. Dieser Fall tritt insbesondere bei weiteren Iterationen der Aufbereitung zu.

6.2.2 Erweiterung von Sicherheitsmusterbäumen

Die Erweiterung eines bestehenden Sicherheitsmusterbaumes wird durch die notwendige Umsetzung einer neuen oder geänderten Sicherheitsanforderung, die Identifikation einer alternativen Lösungs-

möglichkeit für eine Sicherheitsmaßnahme oder durch die Einführung von neuen Sicherheitsprodukten angestoßen. Neben der Erweiterung ist auch eine Reduzierung der Sicherheitsfunktionalität, zum Beispiel durch die Abschaltung eines Sicherheitsproduktes, eine denkbare Weiterentwicklung der Mustersprachen. Da in dieser Arbeit Sicherheitsfunktionalität als eine kritische Komponente in Software-Systemen angesehen wird, wird der Fall der Reduzierung jedoch von der Betrachtung ausgeschlossen.

Wird eine Sicherheitsanforderung nicht ausreichend durch die bestehenden Sicherheitsmaßnahmen abgedeckt, ist die Hinzunahme von weiteren Sicherheitsmaßnahmen notwendig. In diesem Fall werden ähnliche Schritte zur erstmaligen Erstellung des Sicherheitsmusterbaumes für eine Maßnahme durchgeführt. Für die Sicherheitsanforderung wird zunächst eine geeignete Taktik ausgewählt, für welche ein oder mehrere zugehörige abstrakte Sicherheitsmuster zur Umsetzung der Anforderung angewendet werden sollen. Anschließend werden Sicherheitsprodukte der bestehenden Infrastruktur oder externe Produkte hinsichtlich der Umsetzung des Musters analysiert. Ebenso werden die Zusammenhänge zu Sicherheitsmustern der gleichen Abstraktionsebene geklärt, so dass das hinzugefügte Sicherheitsmuster in eine horizontale Mustersprache eingeordnet werden kann. Im Anschluss werden abwechselnd das abstrakte Sicherheitsmuster verfeinert bzw. die technologienahen Muster generalisiert, bis eine direkte Abbildung eines abstrakten Musters auf ein spezialisiertes Sicherheitsmuster möglich ist. Dabei wird bei jedem Vorgang die Anzahl der abstrakten Sicherheits-muster bzw. die Auswahl der Sicherheitsprodukte eingeschränkt.

Im Falle der Identifikation einer alternativen Lösungsmöglichkeit für ein Sicherheitsmuster wird zunächst die Stelle im Musterbaum ausfindig gemacht, an welcher die Alternative einzuordnen ist. Eine solche Alternative kann beispielsweise durch die Analyse der Sicherheitsfunktionalität eines Software-Systems oder durch die Dokumentation eines neuen Sicherheitsmusters in der Fachliteratur eingeführt werden. Ist der Einsatz der Alternative praktikabel, zum Beispiel durch eine effizientere Realisierung einer Sicherheitsanforderung, so wird zunächst geprüft, ob das Sicherheitsmuster mittels der bestehenden Sicherheitsinfrastruktur umgesetzt werden kann oder ob diese durch zusätzliche Produkte erweitert werden muss. Ebenso müssen Abhängigkeiten zu weiteren Sicherheitsmustern in der horizontalen Mustersprache geklärt sowie die Verfeinerung des Musters zu technologienahen Idiomen definiert werden. Diese Aktivitäten sind identisch mit den zuvor besprochenen Schritten zur Aufbereitung der Sicherheitsmuster.

Die Einführung eines neuen Sicherheitsproduktes ist üblicherweise mit dem Einsatz der dadurch bereitgestellten Sicherheitsfunktionalität verbunden. Somit ist es obligatorisch, diese neue Sicherheitsfunktionalität in einem Musterbaum aufzubereiten und mit den bisherigen Musterbäumen abzugleichen. Dies ist insbesondere der Fall, wenn das Sicherheitsprodukt ein oder mehrere ältere Produkte ersetzt. Daher muss für jedes Muster, welches auf einen Pfad in einer vertikalen Mustersprache zu diesen Produkten liegt, auf die Anwendbarkeit im neuen Sicherheitsprodukt überprüft werden. Idealerweise sollte das Produkt so ausgewählt werden, dass eine Rückwärtskompatibilität unterstützt wird. Andernfalls müssen Software-Systeme, welche die älteren Produkte einsetzen, für den Einsatz des neuen Sicherheitsprodukts vorbereitet werden. Durch die Zentralisierung der Sicherheitsfunktionalität in einer Sicherheitsarchitektur sollten sich diese Anpassungen jedoch in Grenzen halten. Des Weiteren ist die Aufbereitung der neuen Funktionalität des eingeführten Sicherheitsproduktes in den existierenden Musterbäumen mittels der bereits beschriebenen Aktivitäten durchzuführen.

6.3 Beispiele für Mustersprachen

Im Folgenden werden die Aktivitäten zur initialen Erstellung und Erweiterung von Sicherheitsmustersprachen anhand der Sicherheitsmaßnahmen für Authentifizierung, Autorisierung und Zugriffskontrolle demonstriert. Die angesprochenen Maßnahmen sind eng miteinander verknüpft und werden in der Praxis gemeinsam eingesetzt. Aufgrund der Absicht der Demonstration ist zu berücksichtigen, dass die besprochenen Resultate bei der Erstellung der Sicherheitsmustersprachen nicht als vollständig zu betrachten sind, sondern vielmehr eine erste Grundlage für weitere Diskussionen und Vervollständigungen darstellen. Ein Grund hierfür ist auch die Vernachlässigung des Abstraktionsniveaus bei der Spezifikation von Mustern in der untersuchten Literatur. Zum Teil wurden daher Muster, welche einen sehr starken Technologiebezug aufwiesen, auf ein höheres Abstraktionsniveau eingeordnet, indem von einer spezifischen Technologie abstrahiert und der grundlegende Lösungsansatz in den Vordergrund gestellt wurde.

Weiter ist zu berücksichtigen, dass nicht für alle existierenden Sicherheitsmaßnahmen bzw. -modelle eine entsprechende Musterbeschreibung existiert. Um die durchgängige Verknüpfung von technologieunabhängigen Mustern mit plattformspezifischen Idiomen zu illustrieren, wird daher im Weiteren die Sicherheitsinfrastruktur des Karlsruher Instituts für Technologie (KIT) als generelle Zielplattform der Sicherheitsmusterbäume betrachtet. Mittels der am KIT eingesetzten Sicherheitsprodukte lässt sich das Middle-Out-Verfahren zur Erstellung der Sicherheitsmusterbäume demonstrieren. Des Weiteren wird die hierdurch bereitgestellte Sicherheitsfunktionalität weiter eingeschränkt, in dem lediglich die Anwendungsschicht als die zu untersuchende Systemschicht festgelegt wird und vor allem Sicherheitsstandards und -rahmenwerke eingeordnet werden, welche in verteilten, Web-basierten Anwendungen zum Einsatz kommen. Hierzu zählt unter anderem der Zugriffskontrollstandard der Extensible Access Control Markup Language [OASIS-XACML-v3.0] sowie das Autorisierungs- und Authentifizierungsrahmenwerk Spring Security [SPRING-SEC] [Mu10]. Dies schränkt somit die Anzahl der untersuchten Sicherheitsmaßnahmen ohne Einschränkung der Allgemeinheit ein und senkt den Umfang der vorgestellten Mustersprachen.

6.3.1 Ausgangspunkt der Mustersprachen

Ausgangspunkt für die Definition von Sicherheitssprachen ist die Spezifikation von abstrakten Sicherheitsmaßnahmen, welche die Wurzeln der von ihnen ausgehenden Mustersprachenbäume darstellen. Zu dieser Spezifikation gehört zunächst die Assoziation zu Sicherheitsanforderungen, welche von den Sicherheitsmaßnahmen umgesetzt werden. Im Falle der hier vorgestellten Sicherheitsmaßnahmen ist der Bezugspunkt zur Analysephase die Sicherheitsanforderung bzw. die zugehörige Vorlage „Zugriffsbeschränkung“ (vgl. Kapitel 5).

Diese Anforderung verlangt, unterschiedlichen Benutzern bzw. Subjekten differenzierten Zugriff auf die durch eine Ressource unterstützen Aktionen, zum Beispiel Lesen und Schreiben, zu ermöglichen. Unter Verwendung der zuvor beschriebenen Taktiken kann diese Anforderung durch die einfache oder kombinierte Anwendung von Sicherheitsmaßnahmen aus unterschiedlichen Kategorien umgesetzt werden. Während detektierende und wiederherstellende Maßnahmen, wie zum Beispiel Auditierung [FH06:304] und Redundanzen, zwar in diesem Falle ebenfalls eine Lösung darstellen, führen sie für diese Sicherheitsanforderung lediglich unterstützende Funktionen aus. Daher wird mit dem Muster Zugriffskontrolle eine präventive Maßnahme gewählt [FH06].

Da das Zugriffskontroll-Muster sehr abstrakt ist, schreibt es lediglich vor, dass die Zugriffe auf Objekte durch Subjekte hinsichtlich ihrer Zulässigkeit kontrolliert werden sollen [FH06:301]. Zugriffskontrolle sollte gewählt werden, wenn den Subjekten nicht im Vornherein bzgl. der

Unterlassung von unzulässigen Zugriffen vertraut wird und die Kontrolle darüber automatisiert erfolgen soll. Auf dem ersten Blick ist bei dieser Formulierung die Trennung zur Sicherheitsanforderung „Zugriffsbeschänkung“ trivial. Dies ist durchaus gewollt, da durch ähnliche Abstraktionsstufen der Übergang von Analysephase zur Entwurfsphase reibungslos möglich wird. Dennoch sei darauf hingewiesen, dass hier eine klare Trennung zwischen einer Anforderungsspezifikation (dem „Was“) und einem Entwurf (dem „Wie“) stattfindet. Hierdurch wird es möglich, dass eine Anforderung durch mehrere Sicherheitsmaßnahmen umgesetzt werden kann. So ließe sich beispielsweise eine abstrakte detektierende Maßnahme „Überwachung“ (engl. monitoring, [FH06:301]) bzgl. Zugriffsbeschränkung so auslegen, dass die Zugriffe von Subjekten auf Objekte überwacht und protokolliert, aber nicht verhindert werden.

Wie in Abbildung 66 dargestellt wird, steht Zugriffskontrolle in Abhängigkeitsbeziehungen zu den Mustern Authentifizierung (engl. Authentication [FH06:304]) und Autorisierung (engl. Authorization [FH06:306]). Die Zugriffskontrolle fokussiert die Überprüfung und Umsetzung einer Zugriffskontrollentscheidung beim Zugriff auf ein Objekt durch ein Subjekt zur Laufzeit eines Systems. Hierzu sind vor allem Architekturerweiterungen durch spezielle Komponenten notwendig, welche Anfragen an die zu schützenden Objekte zur Laufzeit unterbrechen und die Zugriffskontrollentscheidung durchführen sowie durchsetzen. Im Gegensatz zum angesprochenen Überwachungs-Muster, benötigt das Zugriffskontroll-Muster jedoch einen expliziten Regelsatz, durch welchen die zulässigen Zugriffe definiert werden.

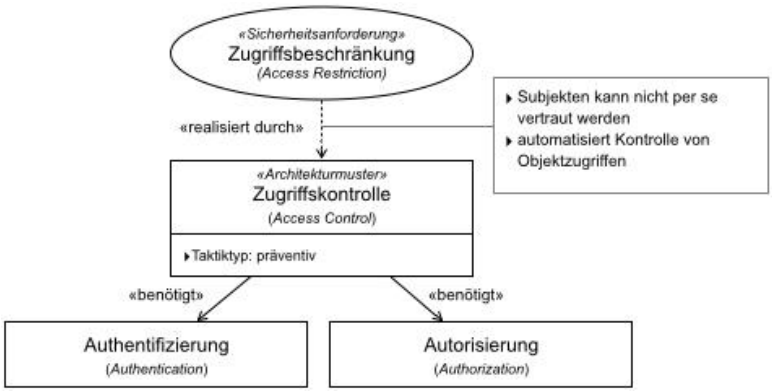

Abbildung 66: Horizontale Mustersprache für Zugriffskontrolle

Als Basis für Zugriffskontrollentscheidungen werden somit Zugriffskontrollrichtlinien (engl. access control policies) benötigt, welche den erlaubten Zugriff von Subjekten auf Objekte definieren und vor der Überprüfungszeit festgelegt wurden. Dieser Regelsatz wird durch das Autorisierung-Muster bereitgestellt [FH06:304] [SF+05:245ff]. Die Spezifikation der Richtlinien entspricht der Zuweisung von Zugriffsrechten auf ein Objekt an ein bestimmtes Subjekt. Daher stehen bei der Autorisierung weniger architekturelle Maßnahmen im Fokus, als vielmehr der Aufbau, die Strukturierung und die Verwaltung der Zugriffskontrollrichtlinien. Die vielfältigen Ansätze hierfür werden im Allgemeinen als Sicherheitsmodelle [Ec09] bzw. Zugriffskontrollmodelle [SS94] bezeichnet. Da vor allem der letztgenannte Begriff aufgrund der in dieser Arbeit vorgenommenen Trennung von Zugriffskontrolle und Autorisierung gegebenenfalls zu Verwirrung führt, wird in dieser Arbeit der wenig übliche Begriff Autorisierungsmodell verwendet. Ebenso wird zum Zwecke der Klarheit statt des Begriffes Zugriffskontrollrichtlinie im Weiteren der Begriff Autorisierungsrichtlinie eingesetzt.

Die Umsetzung von Zugriffskontrollentscheidungen setzt zudem eine Identifizierung und Authentifizierung von einem zur Laufzeit auf ein Objekt zugreifendem Subjekt voraus [Ca04]. Somit

besteht eine Abhängigkeitsbeziehung zwischen Zugriffskontrolle und dem Authentifizierung-Muster [SF+05:187] [FH06:304]. Analog zum Verhältnis von Autorisierung zu Zugriffskontrolle lässt sich eine ähnliche Unterscheidung bei Authentifizierung durchführen. Zum einen werden spezielle Komponenten in der Software-Architektur benötigt, welche die Identifizierung und Authentifizierung von Subjekten zur Laufzeit vornehmen. Anlehnend an die Semantik des Begriffes Zugriffskontrolle, wird in dieser Arbeit für diesen Aspekt der Begriff Zugangskontrolle verwendet.

Zum anderen werden hierzu Authentifizierungsrichtlinien benötigt, welche im Vornherein die Art der Identifizierung und Authentifizierung festlegen sowie Prämissen und Einschränkungen hierfür spezifizieren [BM07]. Im Gegensatz zu der konzeptionellen Trennung von Zugriffskontrolle und Autorisierung hat sich für die Authentifizierung eine solche klare Trennung zwischen Architektur und Richtlinien in der Literatur nicht durchgesetzt. Daher werden in dieser vorgestellten Mustersprache unter dem Authentifizierung-Muster beide Aspekte vereint. In der im Weiteren vorgestellten Verfeinerung des Musters wird diese Unterscheidung wieder aufgegriffen und berücksichtigt.

Zu bemerken ist, dass keine Abhängigkeitsbeziehung zwischen Autorisierung und Authentifizierung besteht. Dies lässt sich ebenfalls damit begründet, dass eine Authentifizierung eines Subjektes zur Laufzeit, d.h. beim konkret stattfindenden Zugriff eines Subjektes auf ein Objekt stattfindet. Dagegen wird die Spezifizierung von Autorisierungsrichtlinien vor Betrieb der Anwendung durchgeführt und ist weitestgehend unabhängig von den konkreten Subjektzugriffen zur Laufzeit.

6.3.2 Authentifizierungssprache

Wie bereits angesprochen umfasst das Authentifizierung-Muster mehrere Aspekte, welche sich sowohl auf die Architektur als auch die Richtlinien dieser Sicherheitsmaßnahme auswirken. Diese verschiedenen Aspekte wirken sich auch auf die Spezifikation der Authentifizierungsmustersprache aus. In Abbildung 67 wird eine erste Spezialisierung des Sicherheitsmusters in dessen wesentlichen Bestandteile durchgeführt. Das Muster „Authentifizierungsnachweis“ (engl. Authentication Credential, Context Holder in [FH06:303]) beschreibt, auf welche Art und Weise sich ein Subjekt identifiziert und den Nachweis für die Assoziation des Subjektes mit dem Identifikator erbringt. Das Muster „Authentifizierungsstärke“ (engl. Authentication Level, [FH06:304]) dagegen legt fest, wie viele Authentifizierungsnachweise erbracht werden müssen. Die konkrete Verwendung und der

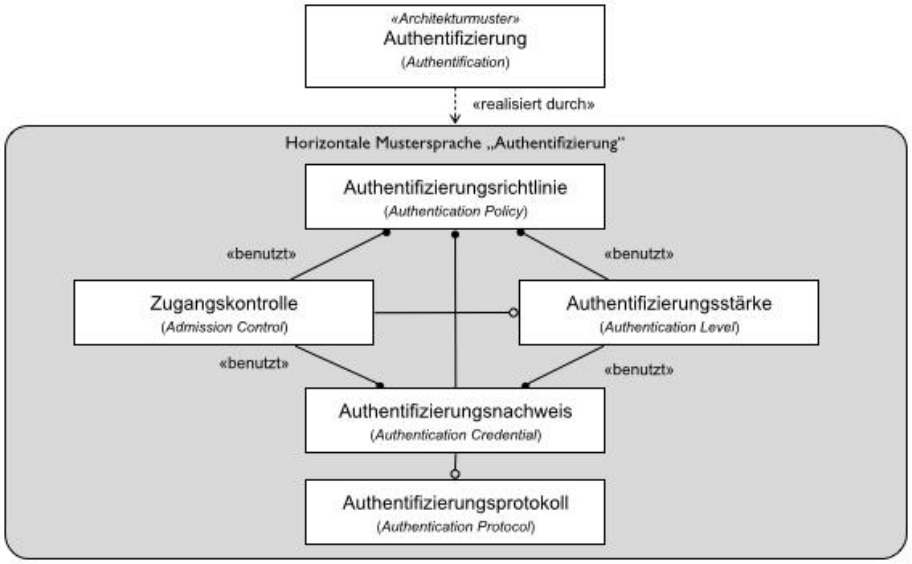

Abbildung 67: Realisierung des Authentifizierung-Muster

spezifische Einsatz von einem oder mehreren Authentifizierungsnachweisen wird durch das Muster „Authentifizierungsrichtlinie“ koordiniert. Die Überprüfung der Authentifizierungsnachweise und -richtlinien zur Laufzeit sowie die Einbettung von dazu nötigen Software-Komponenten in der Software-Architektur wird durch das Muster „Zugangskontrolle“ (engl. Admission Control) bestimmt.

Die korrekte Zuweisung der Identität eines Subjektes ist dabei eine wichtige Voraussetzung für weitere Sicherheitsmaßnahmen [Ec09:439]. Im vorherigen Abschnitt wurde Zugriffskontrolle als abstraktes Sicherheitsmuster vorgestellt, welches Authentifizierung voraussetzt. Ebenso sind Maßnahmen, wie zum Beispiel Auditierung, auf die eindeutige Identität des Subjektes angewiesen. Allerdings weisen alle Authentifizierungsnachweise Fehlerraten bei der Überprüfung der Subjektidentität auf. Eine Verringerung der Fehlerraten führt meist zu einer reduzierten Benutzbarkeit. Somit sind bei der Festlegung und Auswahl einer Form von Authentifizierungsnachweis mehrere Faktoren ausschlaggebend, welche oft konträr zueinander stehen. Da kein perfekter Authentifizierungnachweis existiert, muss eine Auswahl unter Berücksichtigung von Kosten, Bedienbarkeit und Korrektheit der verschiedenen Nachweise durchgeführt werden. In Tabelle 5 sind diese Faktoren zusammengefasst [SF+05:195ff].

Tabelle 5: Kriterien für die Auswahl von Authentifizierungs-Nachweisen

Kategorie	Kriterien
Korrektheit	- **Erkennung von Betrügern:** Es soll erkannt werden, wenn die digitale Identität nicht dem Subjekt zugehörig ist, d.h. sogenannte false positives sollen vermieden werden. - **Erkennung von legitimen Nutzern:** Es soll erkannt werden, wenn die digitale Identität dem Subjekt zugehörig ist, d.h. sogenannte true positives sollen erkannt werden. - **Schutz von Nachweisen:** Die Authentifizierungsnachweise selbst sollen vor z.B. Diebstahl oder Offenlegung geschützt werden.
Bedienbarkeit	- **Berücksichtigung von Nutzercharakteristik:** Die Eigenschaften eines Subjektes, wie z.B. Erfahrung, mobile bzw. feste Örtlichkeit, lokale bzw. entfernte Subjekte müssen berücksichtigt werden. - **Zeit und Aufwand:** Der Aufwand bei der Anwendung des Nachweises soll für das Subjekt minimiert werden. - **Gesundheitsrisiken:** Die schädliche Wirkung (insbesondere bei biometrischen Nachweisen) bei der Anwendung des Nachweises soll für das Subjekt minimiert werden.
Kosten	- **Kosten für Einrichtung:** Die Kosten zur Erstellung, Verteilung sowie Schulung zur Anwendung von Authentifizierungsnachweisen für Subjekte soll minimiert werden. - **Änderungskosten:** Die Kosten für die Änderung bestehender bzw. die Anschaffung neuer Authentifizierungs-Infrastruktur sollen minimiert werden. - **Pflege-, Verwaltungs- und Fixkosten:** Die Kosten zur generellen Instandhaltung und Administration sollen minimiert werden.

Unabhängig von der Art des Nachweises wird zusätzlich eine Form der Übertragung zwischen Subjekt und der authentifizierenden Systemkomponente benötigt. Eine solche Übertragung findet sowohl bei verteilten als auch in lokalen Systemen statt, wie beispielsweise in einer Interprozesskommunikation

in Betriebssystemen oder zwischen den Komponenten eines Software-Systems. Eine direkte Übermittlung über einen unsicheren Kanal führt zu einer erhöhten Wahrscheinlichkeit von Angriffen auf den übertragenen Nachweis, wodurch die Systemsicherheit kompromittiert wird. Daher ist die Übertragung über einen sicheren Kanal oder durch den Einsatz eines sicheren Protokolls abzuwickeln.

Im ersteren Fall wird im Mustersprachenmodell eine optionale Abhängigkeitsbeziehung zwischen dem abstrakten Muster Authentifizierung und dem abstrakten Muster Sicherer Kanal (engl. Secure Channel, [SF+05:434]) modelliert, welche in Abbildung 67 nicht eingezeichnet ist. Im zweiten Fall wird die Übertragung von Authentifizierungsnachweisen mittels des Musters „Authentifizierungsprotokoll“ (engl. Authentication Protocol) durchgeführt. Da in der Mustersprache die Auswahl zwischen der direkten Übertragung eines Nachweises über einen sicheren Kanal als auch die Übertragung mittels eines sicheren Authentifizierungsprotokolls erhalten werden soll, wird ebenfalls eine optionale Benutzt-Beziehung zwischen den Mustern „Authentifizierungsnachweis“ und „Authentifizierungsprotokoll“ modelliert.

Zum Ausgleich der verschiedenen Schwachstellen der Authentifizierungsnachweise werden häufig mehrere Authentifizierungsnachweise von Subjekten verlangt. Die Anwendung des Authentifizierungsstärke-Musters ist insbesondere bei Objekten mit einem hohen Schutzbedarf angebracht. Hierbei sind ebenfalls die Vor- und Nachteile zu beachten, welche bei der Auswahl eines einzelnen Authentifzierungsnachweises berücksichtigt werden müssen. Hinzu kommt, dass die Vor- und Nachteile von verschiedenen Nachweisen kompensiert werden sollten. Somit sind gegebenenfalls nicht alle Nachweisarten beliebig miteinander kombinierbar [SF+05:212]. Eine Kombination von Nachweisen wird meist als starke Authentifizierung (engl. strong authentication,) bzw. gemäß der Anzahl der verwendeten Nachweise als 2-Faktor (engl. 2-factor Authentication, [Ra02]), 3-Faktor (3-factor authentication, [Ra02]) oder Mehrfaktor (engl. Multifactor Authentication, [Ec09:441]) bezeichnet.

Während die Muster Authentifizierungsnachweis und -stärke die Art und Anzahl der verwendeten Nachweise beschreiben, spezifiziert das Muster Authentifizierungsrichtline deren koordinierte Verwendung. Zudem wird die Verwendung von Werkzeugen zur Konfiguration und Wartung von Authentifizierungsnachweisen beschrieben. Die so beschriebenen Richtlinien sind dabei stark abhängig von den verwendeten konkreten Nachweisen. Jedoch lassen sich allgemeine Richtlinien beschreiben, welche auf alle Authentifizierungsnachweise zutreffen. Hierzu zählen zum Beispiel die Beschränkung der Anzahl der Authentifizierungsversuche, die Verzögerung von Folgeversuchen bei fehlgeschlagenen Authentifizierungsversuchen, Aufrufreihenfolge und Zusammenspiel von Authentifizierungsnachweisen [FH06:305f].

Das Zugangskontrolle-Muster beschreibt im Gegensatz zur den bisherigen Mustern die Einbindung von Authentifizierungsfunktionalität in die Software-Architektur einer Anwendung. Somit werden die Komponenten und Dienste, welche für die Umsetzung dieser Funktionalität benötigt werden, sowie die Integration mit den fachlichen Komponenten spezifiziert. In dem Muster werden entsprechend Dienste beschrieben, mittels welchen die Authentifizierungsnachweise ausgewertet und gegebenenfalls auch verwaltet werden können. Daher steht das Muster in Abhängigkeitsbeziehungen zu den restlichen Mustern der Mustersprache, da Details über die verwendeten Nachweisformen benötigt werden, um diese entsprechend zu überprüfen. Ebenso ist hierfür die Einhaltung der Authentifizierungsrichtlinien zu überprüfen.

Authentifizierungsnachweis

Authentifizierungsnachweise werden klassischerweise in drei Kategorien eingeteilt [WM05:332-334] [Sc01:127] [VG02:61-67]. Bei wissensbasierten Authentifizierungsnachweisen (engl. Knowledge Factor bzw. „what-you-know”-Authentication) wird die Identität des Subjektes durch Angabe eines Geheimnisses wie beispielsweise eines Passwortes sichergestellt. Dagegen wird bei einer

besitzbasierten Authentifizierung (engl. Ownership Factor bzw. „what-you-have"-Authentication) die Identität durch Vorzeigen oder Vorlegen eines Objektes, wie zum Beispiel einer Smartcard, auf welcher ein Geheimnis abgelegt ist, vorgenommen. Eine inhärente oder biometrische Authentifizierung (engl. Inherence Factor bzw. „what-you-are"-Authentication) nutzt die biometrischen Eigenschaften eines menschlichen Subjektes, wie zum Beispiel Fingerabdruck, um die Identität des Subjektes zu beweisen. Im Folgenden werden die inhärenten Authentifizierungsfaktoren nicht weiter berücksichtigt, da sie auch am KIT keine unmittelbare Anwendung finden.

Die Anwendung des Musters Authentifizierungsnachweis wählt eine Form der Authentifizierung aus einer der Kategorien aus. Wie Abbildung 68 dargestellt, werden daher im Mustersprachenmodell die Kategorien als Spezialisierung des Authentifizierungs-Musters modelliert, welche mittels einer XOR-Relation verbunden ist. Die Anwendung von mehreren Nachweisen wird durch das Authentifizierungsstärke-Muster beschrieben. Im Wesentlichen handelt es sich dabei um eine Mehrfachanwendung des Authentifizierungs-Musters, so dass dieser Auswahlprozess mehrmals durchlaufen wird. Im Folgenden werden die einzelnen Kategorien weiter verfeinert und beschrieben, wobei die wichtigsten Umsetzungen fokussiert werden. Zunächst werden die klassischen Kategorien der Wissens- und Besitzbasierten sowie der Inhärenten Authentifizierung als Muster beschrieben und weiter verfeinert. Anschließend wird kurz auf die Verwendung von zeitgebundenen Authentifizierungsnachweisen als alternativer Authentifizierungsnachweis eingegangen.

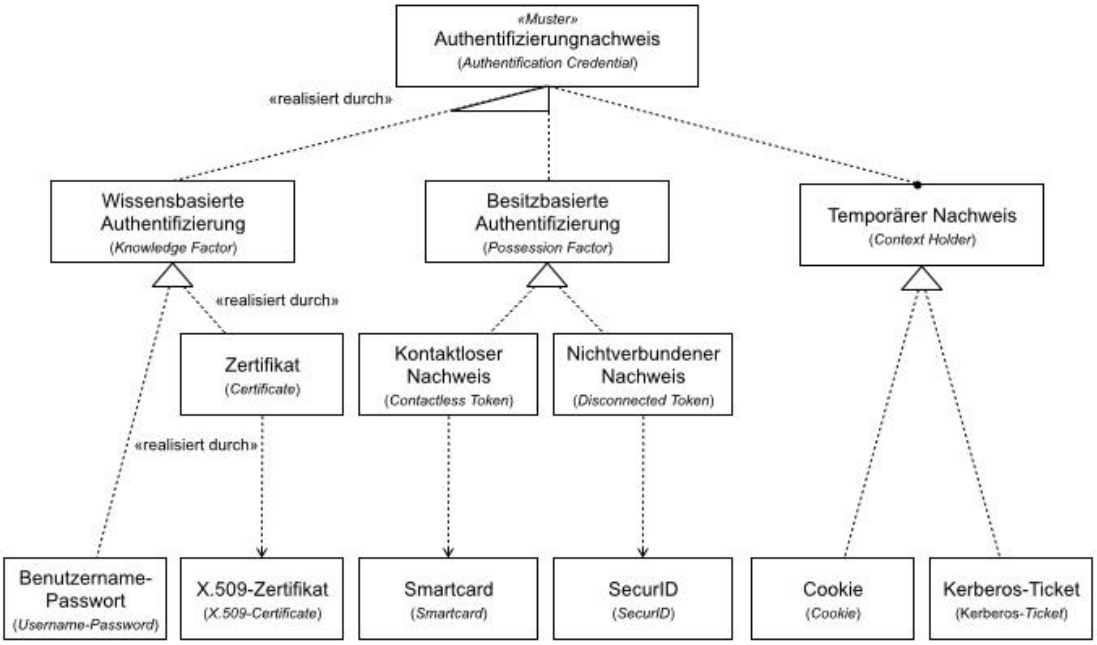

Abbildung 68: Vertikale Mustersprache für Authentifizierungsnachweise

Wissensbasierte Authentifizierung

Die wissensbasierte Authentifizierung gehört zu der verbreitetsten Form der Authentifizierung, da sie durch eine benutzerfreundliche Bedienbarkeit und hohe Kosteneffizienz ausgezeichnet ist. Die Identität des Subjektes wird durch die Preisgabe eines Geheimnisses, über welches jeweils nur das sich authentifizierende Subjekt und die authentisierende Partei, d.h. ein Authentifizierungsdienst, Kenntnis haben. Hierbei ist die Zuverlässigkeit bei der korrekten Erkennung von legitimen und nichtlegitimen Nutzern als mittelmäßig einzustufen. Ebenfalls benötigen wissensbasierte Nachweise einen hohen Schutz, da durch die Weitergabe des Geheimnisses die eindeutige Authentifizierung des Subjektes nicht mehr garantiert werden kann [SF+05:210].

Die Authentifizierung eines Subjektes durch die Angabe eines Benutzernamens und eines zugehörigen Passwortes gehört heutzutage zu den geläufigsten Verfahren zur Authentifizierung in Software-Systemen. Vorzugsweise wird dieser Nachweis von menschlichen Nutzern erbracht, da der Benutzername und das Passwort merkbare Zeichenketten darstellen [SF+05:218]. Hierin liegt allerdings auch der Schwachpunkt dieses Nachweises. Einfache Zeichenketten lassen sich mittels verschiedener Verfahren, wie zum Beispiel Wörterbuchangriffen, sehr einfach nachbauen, da sie häufig auf einem existierenden Wort in einer bestimmten Sprache basieren [Ec09:444f] [SF+05:218]. Daher ist bei Einsatz dieses Musters die Verwendung und Einhaltung von strengen Richtlinien zu berücksichtigen, welche die Komplexität des Passwortes durch bestimmte Regeln erhöhen.

Digitale Zertifikate stellen einen wissensbasierten Authentifizierungsnachweis dar, der sowohl für menschliche Nutzer als auch nichtmenschliche Subjekte einsetzbar ist [FH06:307]. Ein Zertifikat ist eine durch eine dritte, vertrauenswürdige Partei ausgestellte Bescheinigung über die Zuordnung einer digitalen Signatur zu einem Subjekt [Ec09:389]. Digitale Signaturen setzen kryptographische Methoden, wie zum Beispiel Hashfunktionen und asymmetrische Kryptografie ein, um unter anderem die Authentizität von signierten Dokumenten sowie die Identität des Unterzeichnenden zu bestätigen [Ec09:372].

Der X.509-Standard [IETF-X.509] beschreibt den Aufbau sowie den Inhalt eines Zertifikates und gilt heutzutage als eine bewährte Praxis [Ec09:390]. Beispielsweise wird die Server-Authentifizierung und der Aufbau einer verschlüsselten HTTP-Verbindung mittels des SSL/TLS-Protokolls unter Einsatz von X.509-Zertifikaten durchgeführt. Neben der teilweise komplexen initialen Bedienung, liegt der Nachteil von Zertifikaten in der benötigten Infrastruktur zur Verwaltung der Zertifikate (engl. Public Key Infrastructure, PKI [Ec09:395] [Ra02:301] [FH06:307].

Weitere spezifische wissensbasierte Authentifizierungsmethoden werden an dieser Stelle nicht betrachtet, da die gewählte Sicherheitsplattform eine Einschränkung vorgibt. Am KIT werden Authentifizierungsnachweise mittels Benutzername und Passwort vor allem für den Zugang zu Web-basierten Diensten eingesetzt. Hierzu zählen unter anderem das Studierendenportal zur Verwaltung des Studiums für die KIT-Studierenden sowie das interne Mitarbeiterportal. Zudem werden Zertifikate sowohl für Mitarbeiter zur Signierung von E-Mails als auch zur Authentifizierung von Web-Servern ausgestellt. Daher bilden diese Nachweismethoden die vertikale Mustersprache für die wissenbasierte Authentifizierung.

Besitzbasierte Authentifizierung

Im Gegensatz zu wissensbasierten Authentifizierungsnachweisen wird bei besitzbasierten Nachweisen nicht ein Geheimnis zur Authentifizierung eines Subjektes verlangt, sondern das Vorzeigen bzw. die Verwendung eines Hardware-Token. Prinzipiell lassen sich besitzbasierte Nachweise als Weiterentwicklung von wissensbasierten Nachweisen ansehen, da anstelle des Erinnerns des Geheimnisses, dieses auf den Hardware-Token vermerkt ist oder erzeugt wird. Jedoch liegt hier auch ein Schwachpunkt hinsichtlich des Schutzes dieser Art der Authentifizierung, da mit Verlust des Tokens ein Subjekt die Fähigkeit verliert, sich am System zu authentifizieren.

Die Benutzbarkeit ist ähnlich wie bei wissensbasierten Nachweisen gut, vorausgesetzt, die zum Einlesen der Hardware-Token benötigte Infrastruktur ist vorhanden. Dies wiederum führt unter Umständen zu erhöhten Kosten bei der Integration in die existierende Systemlandschaft sowie bei der Erstellung von Hardware-Token. Auch die Verlässlichkeit bei der Erkennung von legitimen und nichtlegitimen Subjekten kann je nach Art des Tokens variieren. Eine erhöhte Verlässlichkeit wird durch die Kombination mit einem zweiten Nachweisfaktor, zum Beispiel einem Passwort erreicht, weshalb besitzbasierte Nachweise üblicherweise in einer Zwei-Faktor-Authentifizierung in Kombination mit einem wissensbasierten Nachweis verwendet werden [Ra02:59]. Durch die physische

Präsenz des Tokens ist diese Form der Authentifizierung auf menschliche Nutzer eingeschränkt [SF+05:211f]. Im Allgemeinen lassen sich verbundene Nachweise, nichtverbundene Nachweise sowie kontaktlose Nachweise unterscheiden. Sie bilden die jeweilige Kategorie der Spezialisierungen im Mustersprachenmodell.

Am KIT lassen sich verschiedene Formen der besitzbasierten Authentifizierung identifizieren, welche aber nicht unmittelbar für den Zugang zu Web-basierten Systemen dienen. Jedoch werden Gebäudezugänge und Bibliotheksausleihen mit der den KIT-Mitarbeitern und -Studierenden ausgestellten KIT-Card geregelt. Ebenso setzen verschiedene kritische Systeme eine Mehr-Faktor-Authentifizierung voraus, welche eine besitzbasierte Authentifizierung beinhalten. Im Folgenden werden daher diese Formen der Vollständigkeit halber erläutert.

Unter dem Muster „Verbundener Nachweis“ werden in der vorgestellten Mustersprache physische Nachweise verallgemeinert. Sie benötigen eine direkte Verbindung zu einem System, um die Authentifizierung durchzuführen. Hierunter fallen zum Beispiel Chipkarten (engl. Smartcards, [Ec09:529] [WM05:333]) und USB-Token, von denen die gespeicherten Geheimnisse mittels spezieller Kartenlesegeräte bzw. durch Anschluss an einen USB-Port ausgelesen werden können. Während im ersten Fall die Anschaffung und Installation von speziellen Lesegeräten zusätzliche Kosten verursacht, sind USB-Verbindungen standardmäßig in Computersystemen vorhanden.

Eine Weiterentwicklung von verbundenen Nachweisen stellen Geräte dar, welche keine direkte Verbindung zur einem Rechnersystem benötigen, das Geheimnis zur Authentifizierung jedoch über eine örtlich begrenzte Funk-Verbindung übertragen. Zu diesen unter dem Muster „Kontaktlose Nachweise“ zusammengefassten Nachweisen zählen zum Beispiel auf Bluetooth-Kommunikation aufbauende Verfahren ebenso wie die neuere Nahfeldkommunikation (engl. Near Field Communication, NFC).

Physische Nachweise, welche nicht mit einem Rechnersystem verbunden werden müssen, sind unter dem Muster „Nichtverbundener Nachweis“ zusammengefasst. In diesem Fall wird periodisch oder auf Anforderung ein Geheimnis mittels physischer Geräte generiert, welches vom Subjekt im System eingegeben werden muss. Es handelt sich bei den Geheimnissen im Allgemeinen um einmalig verwendbare Passwörter, weshalb dieses Muster auch als Spezialisierung des Einmal-Passwort-Musters angesehen werden kann. Durch die Präsenz eines physischen Gerätes, wird das Muster allerdings unter der besitzbasierten Authentifizierung eingeordnet. Beispiel für nichtverbundene Nachweise sind Transaktionsnummern-Generatoren für Aktivitäten im Online-Banking, da sie als Hardware-Ressource dem Subjekt auf Anfrage notwendige Einmal-Passwörter ausstellen. Eine ähnliche Form nimmt auch das SecurID-Authentifizierungssystem der Firma RSA Data Security an [Ec09:452], welches in regelmäßigen Abständen eine Zufallszahl generiert, die in Zusammenhang mit einem Benutzername-Passwort-Nachweis angegeben werden muss [Ra02:59] [Ec09:452].

Temporäre Nachweise

Während die Muster der Nachweiskategorien die Authentifizierung direkt übernehmen, wird das Muster temporärer Nachweis (engl. Context Holder, [FH06]) eingesetzt, um den Austausch von Authentifizierungsnachweisen zu reduzieren. Durch deren wiederholte Übertragung von steigt die Wahrscheinlichkeit, dass sie durch Angriffe abgefangen werden, wodurch die Gefahr von Wiederholungsangriffen steigt. Um dies zu umgehen, beschreibt das Muster „Temporärer Nachweis“, dass dem Subjekt nach einer erfolgreichen Authentifizierung ein zeitlich beschränkter Nachweis ausgestellt wird. Dieser weist das Subjekt bei Folgeanfragen an Objekten als bereits authentifiziert aus.

Im Detail beschreibt das Muster die Protokolle die zur Generierung und zum Austausch des temporären Nachweises verwendet werden. Da Anfragen an Objekte in den meisten Fällen keine Einzelanfragen darstellen, sondern eine Folge von Anfragen eines Subjektes an mehrere Objekte in

einem Software-System ausgetauscht werden, wird im Mustersprachenbaum „Temporärer Nachweis“ als verpflichtendes Muster modelliert. Somit bleibt der ursprüngliche Einsatz dieses Musters als beschränkter Ersatz nach erfolgreicher Authentifizierung mittels eines Authentifizierungsnachweises aus den aufgezählten Kategorien erhalten.

Im Kontext von Web-basierten Software-Systemen dienen beispielsweise Cookies als Container für Formen von temporären Nachweisen, welche für die Zeit einer Sitzung im Browser des Nutzers gespeichert werden. Die am KIT zur Verfügung stehenden Web-Anwendungen setzen diese Technik ein. Im Kontext des Authentfizierungsprotokolls Kerberos dienen sogenannte Tickets als zeitlich begrenzte und sich periodisch ändernde Nachweise für authentifizierte Subjekte [Ec09:494]. Kerberos-Tickets werden beispielsweise im Betriebssystem zur Authentifizierung beim KIT-weiten Active Directory eingesetzt.

Authentifizierungsstärke

Zur Reduzierung der Betrugsversuche bei Authentifizierungsvorgängen kann eine Kombination von Authentifizierungsnachweisen erforderlich sein. Diese Festlegung wird im Mustersprachenbaum durch das Muster „Authentifizierungsstärke“ festgelegt. Die Spezialisierung des Musters ist in Abbildung 69 wiedergegeben. Das grundlegende Verfahren der Bereitstellung eines einzelnen Authentifizierungsnachweises wird durch das Muster „Basis-Authentifizierung“ wiedergegeben. Die Kombination der Authentifizierungsnachweise aus zwei unterschiedlichen Nachweistypen wird durch das 2-Faktor-Authentifizierungs-Muster spezifiziert.

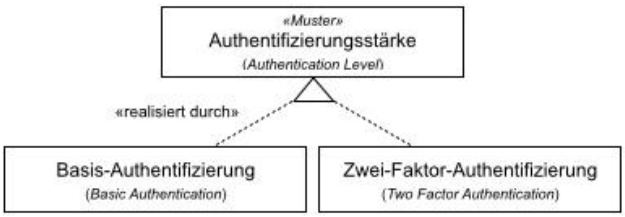

Abbildung 69: Realisierungen des Authentifizierungsstärke-Musters

Sowohl die Basis-Authentifizierung als auch eine 2-Faktor-Authentifizierung wird am KIT eingesetzt. So wird beispielsweise für die Begleichung von Mahngebühren an der KIT-Bibliothek die Eingabe von Benutzername und Passwort sowie die Vorlage der KIT-Karte zur Authentifizierung verlangt. Im Gegensatz dazu wird eine 3-Faktor-Authentifizierung am KIT nicht eingesetzt, weshalb diese Authentifizierungsstärke aus dem Musterbaum ausgeschlossen wird.

Authentifizierungsprotokolle

Nach Festlegung des verwendeten Authentifizierungsnachweises ist es nötig, das Protokoll zur Übermittlung des Nachweises zu spezifizieren. Wird der Nachweis über einen gesicherten Kanal versendet, ist eine direkte Übermittlung möglich. Im Allgemeinen kann diese Voraussetzung nicht eingehalten werden, weshalb Authentifizierungsprotokolle eingesetzt werden, welche die gesicherte Übertragung der Authentifizierungsnachweise auch über einen ungesicherten Kanal ermöglichen. Abbildung 70 zeigt die am KIT unterstützten Authentifizierungsprotokolle.

Die Umsetzung der wissensbasierten Authentfizierung, d.h. der Austausch des gemeinsamen Geheimnisses, wird dabei zumeist durch ein Frage-Antwort-Protokoll (engl. Challenge Response Protocol, [Ec09:461]) durchgeführt. Hierbei werden bei Anfragen auf ein geschütztes Objekt dem anfragenden Subjekt eine oder mehrere Fragen gestellt, die es durch Angabe des Geheimnisses oder anderweitig berechneter Werte zu beantworten hat. Eine einmalige Frage-Anwort-Übermittlung wird

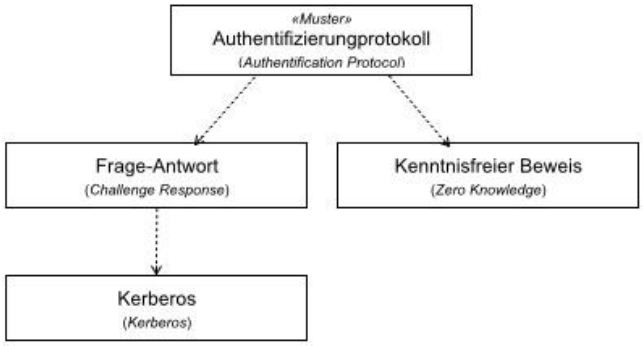

Abbildung 70: Realisierungen des Authentifizierungsprotokoll-Musters

zum Beispiel bei Einsatz eines Benutzernamens-Passwort-Nachweises verwendet. Eine mehrfache Interaktion liegt zum Beispiel bei Einmal-Passwörten und bei einer Mehrfaktor-Authentifizierung vor.

Zur Absicherung der Protokolle werden zusätzlich symmetrische bzw. asymmetrische Verfahren eingesetzt, mittels welcher die gestellten Fragen und insbesondere Passwörter verschlüsselt übertragen werden [Ec09:462-465]. Durch den Austausch von Informationen sind Frage-Antwort-Verfahren verwundbar gegenüber Lausch- und Wiederholungsattacken [Ec09:463]. Ein bekanntes Beispiel für ein Frage-Anwort-Protokoll in verteilten Systemen ist das Kerberos-Protokoll [Ec09:495], welches wie besprochen am KIT eingesetzt wird und daher im Mustersprachenbaum als Spezialisierung des Musters „Authentifizierungsprotokoll“ definiert wird.

Der Vollständigkeit halber wird eine Alternative zu Frage-Anwort-Protokollen in Abbildung 69 dargestellt. Hierunter fallen Authentifizierungsverfahren, welche einen kenntnisfreien Beweis (engl. Zero Knowledge Protocol, [Ec09:465ff]) nutzen, wie beispielsweise das Fiat-Shamir-Verfahren [FS87] [Ec09]. Im Allgemeinen sind Zero-Knowledge-Protokolle beim Einsatz von besitzbasierten Authentifizierungsnachweisen wie zum Beispiel Smartcards relevant.

Zugangskontrolle

Während die bisherigen Sicherheitsmuster die Konfiguration der Authentifizierung in Software-Systemen behandeln, setzt sich das Muster der Zugangskontrolle mit den Architekturkomponenten auseinander, welche die für die Authentifizierung notwendigen Daten verarbeiten.

Eine grundlegende Voraussetzung für einen geschützten Zugang zu einem Software-System besteht in der Reduzierung von Zugangsmöglichkeiten und der zentralen Verwaltung der Zugangskontrolle. Das Muster „Zentraler Zugangspunkt“ wird zu diesem Zweck eingesetzt, um komplexe Software-Systeme vor ungeregeltem Zugang durch beliebige Eintrittspunkte zu schützen und dem Benutzer einen klar definierten Zugangspunkt zum System bereitzustellen. Ein zentraler Zugangspunkt ist notwendig, um einen klaren Punkt zur Durchführung der Authentifizierung in der Architektur des System zu definieren und somit redundante Authentifizierungsfunktionalität zu vermeiden. Aufgrund der weitestgehend abstrakten Beschreibung des Musters in [SF+05:279ff] stellt der zentrale Zugangspunkt eine erste Realisierung des Zugangskontrolle-Musters dar. In Abbildung 71 wird die Umsetzung des Zugangskontrolle-Musters dargestellt.

Um eine flexible Authentifizierungsstrategie zu ermöglichen, wird das Zentraler-Zugangspunkt-Muster mit dem Muster „Kontrollpunkt“ kombiniert, mittels welchem die eigentliche Prüfung der Authentifizierungsinformation weitergeleitet wird. Das Kontrollpunkt-Muster lässt sich für die Überprüfung von Zugriffskontrollrichtlinien einsetzten und wird daher in Abschnitt 6.3.3 genauer beleuchtet.

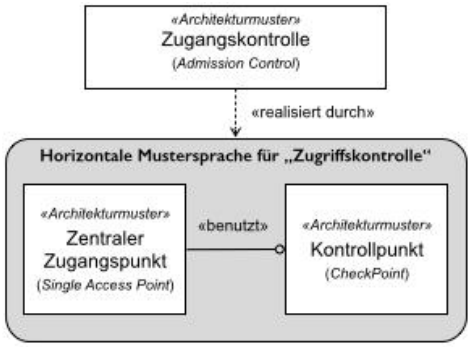

Abbildung 71: Realisierung des Zugangskontrolle-Musters

Wie in Abbildung 72 dargestellt, wird eine mögliche Realisierung des zentralen Zugangspunktes durch das Authentifizierer-Muster [SF+05:323ff] bereitgestellt, welches die Verifikation der durch die Benutzer vorgelegten Authentifizierungsnachweise beschreibt. Durch den Authentifizierer werden die auf einer abstrakteren Ebene definierten Authentifizierungsprotokolle und -nachweise eingesetzt, um die Authentifizierung von Subjekten durchzuführen. Eine Variation des Authentifizierers ist das Muster „Single Sign-On“, durch welches die erfolgreiche Authentifizierung auf mehrere Software-Systeme ausgeweitet wird.

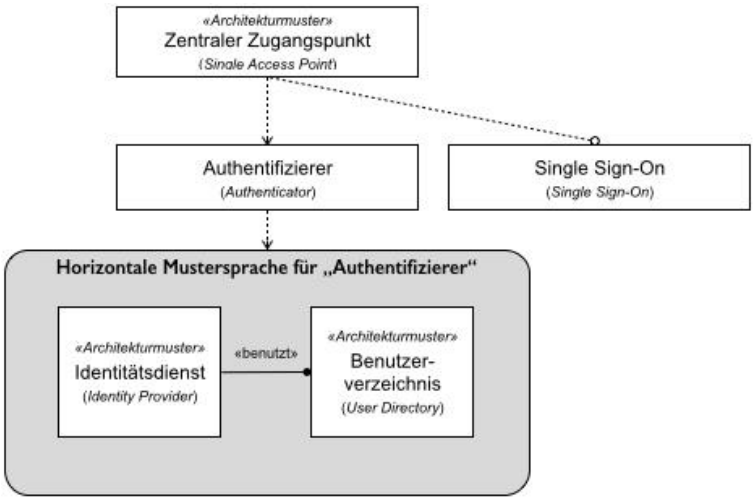

Abbildung 72: Realisierung des Musters „Zentraler Zugangspunkt“

Zur Umsetzung des Authentifizierers lassen sich weitere Entwurfsmuster einsetzten, mittels welchen dessen Funktionalität auf verschiedene Komponenten verteilt wird. So lässt sich zunächst ein Identitätsdienst (engl. Identity Provider [DF+07]) einsetzen, um eine zentrale Schnittstelle zur Kontrolle von Authentifizierungsnachweisen und der Ausstellung von temporären Nachweisen bereitzustellen. Der Identitätsdienst nutzt hierzu das Muster „Benutzerverzeichnis“, um die an ihn übermittelten Nachweise mit den im Benutzerverzeichnis abgelegten Daten zu vergleichen.

Am KIT wird die Funktionalität des Identitätsdienstes sowie des Benutzerverzeichnisses durch eine zentral bereitgestellte Instanz des Microsoft Active Directory (AD) umgesetzt. Ebenso wird eine Implementierung der Alternative Single Sign-On durch ein Shibboleth-Authentifizierungsdienst realisiert, wodurch die Benutzernachweise für das KIT auch in externen Web-Diensten und -anwendungen zur Authentifizierung eingesetzt werden können.

6.3.3 Autorisierungssprache

Ausgehend von dem abstrakten Autorisierungsmuster, lässt sich eine erste Spezialisierung hinsichtlich der Art der Zugriffskontrolle vornehmen. Klassischerweise wurden lediglich system- und nutzerbestimmte Autorisierungsmodelle unterschieden. In betrieblichen Software-Systemen werden des Weiteren rollenbasierte Richtlinien verwendet. Relativ neu sind Richtlinien, welche auf die verfügbaren Metadaten von Subjekt, Objekt und der Ausführungsumgebung eingehen. In Abbildung 73 ist die Einordnung von relevanten Autorisierungsmodellen zu Demonstrationszwecken dargestellt.

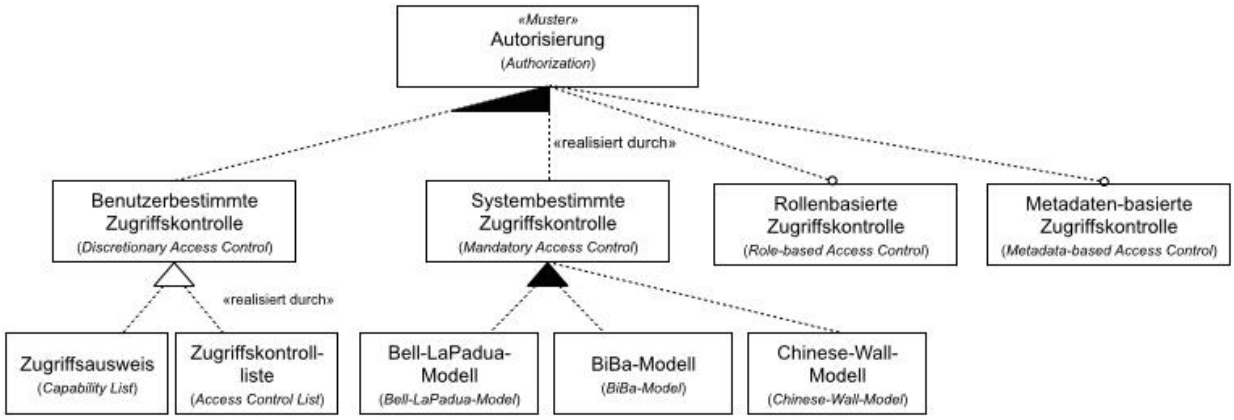

Abbildung 73: Vertikale Mustersprache für das Autorisierung-Muster

Benutzerbestimmte und systembestimmte Autorisierungsmodelle

Eine benutzerbestimmte Autorisierung (engl. discretionary access control, DAC) ermöglicht es Subjekten, die Zugriffsrechte auf Objekte selbst zu bestimmen und weiterzugeben [WM05:136] [Ec09:212]. Hierbei wird davon ausgegangen, dass einem Objekt ein Subjekt als Ressourcenbesitzer zugeteilt wird. Dieser hat entsprechend alle Zugriffsrechte auf das Objekt. Des Weiteren kann der Ressourcenbesitzer die möglichen Zugriffsrechte an andere Subjekte weitergeben. Eine Erweiterung von DAC-Modellen sieht es vor, dass Ressourcenbesitzer auch die Eigentumsrechte an andere Subjekte weitergeben. Ein bekanntes Beispiel für diese Art der Zugriffskontrolle sind Dateien eines Betriebssystems, welche einem Benutzer gehören. Andere Benutzer erhalten die Möglichkeit Dateien zu lesen oder zu schreiben, indem der Besitzer ihnen die entsprechende Freigabe erteilt. Eine typische Umsetzung von benutzerbestimmten Zugriffskontrolle sind Zugriffskontrolllisten (engl. access control list, ACL) sowie Zugriffskontrollausweise (engl. capability list, [Ec09:633ff]).

Dem DAC-Modellen entgegen stehen systembestimmte Autorisierungsmodelle (engl. mandatory access control, MAC). Bei diesen Modellen werden die Zugriffsrechte durch eine zentrale administrative Instanz vorgegeben und durch das System umgesetzt [WM05:135] [Ec09:672]. Ein Benutzer bzw. ein Subjekt hat somit keine Möglichkeit, die Autorisierungsrichtlinien zu verändern. DAC- und MAC-Modelle können auch simultan eingesetzt werden. In solchen Fällen werden die DAC-Richtlinien gegebenenfalls durch die MAC-Richtlinien dominiert, d.h. eine nach den DAC-Richtlinien erlaubte Operation an einer Ressource wird durch eine den gleichen Zugriff einschränkende MAC-Richtlinie unterbunden. Ein Beispiel hierfür ist das Open-Source Betriebssystem SELinux [Ec09:675].

Systembestimmte Autorisierung wird eng mit einer mehrschichtigen Autorisierung (engl. multi-level security, MLS) in Verbindung gebracht. In dieser Form werden Subjekte und Objekte mit Sicherheitsklassifikationen versehen. Mittels der Objektklassifikation wird die Art der sensitiven

Daten festgelegt, welche in einem Objekt gespeichert werden können. Diese Sensitivitätsklassen sind dabei partiell geordnet, zum Beispiel streng geheim, geheim, vertraulich und öffentlich. Subjekte werden ebenfalls in diese Klassen eingeordnet, wodurch ihre Sicherheitsklassifikation (engl. clearance) bestimmt wird. Die systembestimmten Regeln (MAC-Richtlinien) regeln den Zugriff und den Informationsfluss zwischen diesen Sensitivitätsklassen [Ec09:672] [WM05:135] [SF+05:253].

Das bekannteste Modell für die systembestimmte Autorisierung ist das Modell von Bell und LaPadula [BL73]. Zahlreiche Varianten dieses Modells wurden in der Literatur vorgestellt. In [Sa93] wird eine essentielle Basis des Modells vorgestellt, wodurch es zu Abweichungen zu der originalen Formulierung des Modells kommt. Für die Einordnung in die gewählte Abstraktionsstufe im Mustersprachenbaum ist es jedoch geeignet, da ausgehend davon, die verschiedenen Spezialisierungen des Modells als Varianten in den Mustersprachenbaum eingeordnet werden können. Ziel des Modells ist es, das Schutzziel der Diskretion umzusetzen [Ec09:266] [Sa93] [SF+05:254].

Während das Bell-LaPadula-Modell das Schutzziel der Diskretion umsetzt, zielt das Biba-Model [Bi77] [SF+05:254] auf die Erhaltung des Schutzzieles der Integrität ab. Das wesentliche Konzept hinter dem Biba-Modell ist die Vermeidung von Informationsflüssen von Objekten mit niedrigeren Integritätsanforderungen in Objekte mit hohen Integritätsanforderungen [Sa93]. Wie in [Sa93] besprochen, lassen sich das Bell-LaPadula-Modell und das Biba-Modell in Situationen, in welchen sowohl Integritäts- als auch Diskretionsanforderungen erfüllt werden müssen, gemeinsam verwenden. Somit wird die Spezialisierung durch eine Oder-Beziehung modelliert, wodurch die gemeinsame Anwendung der beiden Modelle ermöglicht wird.

Während das Bell-LaPadula- und das Biba-Modell neben einem kommerziellen Einsatz häufig in militärischen Projekten eingesetzt werden [SF+05:255], ist das Chinese-Wall-Modell von Brewer und Nash [BN89] [Sa93] speziell auf den kommerziellen Sektor zugeschnitten. Ziel des Modells ist es, einen Informationsfluss zu verhindern, welcher zu einem Interessenkonflikt für Subjekte führen kann [Sa93]. In [Sa92] wird das Chinese-Wall-Modell als Spezialisierung des Multi-Level-Security-Musters bzw. des MAC-Musters vorgestellt.

Rollenbasierte Autorisierungsmodelle

Die bisher vorgestellten Varianten von Autorisierungsmodellen stellen die zu schützenden Objekte in den Vordergrund. Bei einer größeren Menge an zu schützenden Objekten bzw. einer zunehmenden Anzahl an Subjekten steigt die Komplexität der Verwaltung der Zugriffsrichtlinien. Deshalb werden vor allem im geschäftlichen Umfeld die von Personen zu erledigenden Aufgaben in den Fokus des Zugriffsschutzes gestellt. Die zur Durchführung von bestimmten Aufgaben notwendigen Zugriffsrechte werden hierzu von den jeweiligen Objekten entkoppelt und in Rollen gesammelt. Rollen repräsentieren somit ein typisches Aufgabenfeld, welches durch einzelne Subjekte durchgeführt wird. Einem Subjekt werden Zugriffsrechte auf Objekte durch die Zuweisung zu ein oder mehreren Rollen erteilt. Voraussetzung ist hierbei, dass nur eine geringe Fluktuation der Aufgaben und somit der in den Rollen zusammengefassten Zugriffsrechte vorliegt.

Es existieren zahlreiche Varianten des allgemeinen Modells für rollenbasierte Zugriffskontrolle. In [SC+96] stellen Sandhu et al. eine in Abbildung 74 dargestellte Familie von vier konzeptuellen Referenzmodellen für die rollenbasierte Autorisierungsmodelle vor. Diese stellen die grundlegendsten Formen für weitere Spezialisierungen von RBAC-Modellen dar. Das dort vorgestellte Grundmodell RBAC-0 entspricht bereits dem vorgestellten abstrakten Muster für RBAC. Es besteht aus einer oder mehreren Rollen, Subjekten, Zugriffsrechten sowie Sitzungen. Ein Zugriffsrecht wird dabei abstrakt, als die Erlaubnis auf ein oder mehreren Objekten mit einer bestimmten Aktion zugreifen zu dürfen, definiert. Somit wird der Abstraktionsgrad der Objekte nicht direkt festgelegt, sondern durch den Kontext des Systems bestimmt [SC+96]. Einer Rolle können mehrere Zugriffsrechte zugewiesen

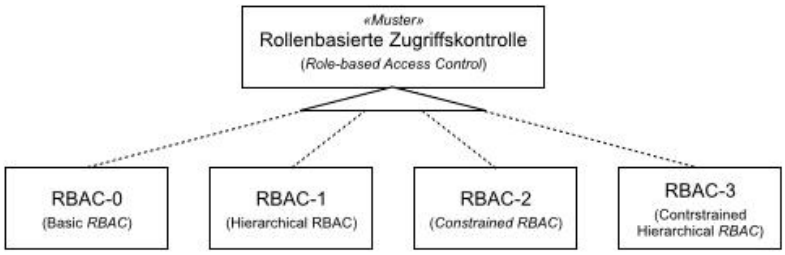

Abbildung 74: Realisierungen des Musters rollenbasierte Zugriffskontrolle

werden. Ebenso können Zugriffsrechte mehreren Rollen angehören. Die gleiche Multiplizität besteht auch für die Beziehung zwischen Rollen und Subjekten. Ein Subjekt aktiviert seine Zugehörigkeit zu ein oder mehreren Rollen in Sitzungen. Im letzteren Fall akkumulieren sich die Rechte des Subjektes aus allen aktiven Rollen. Auch können mehrere Sitzungen gleichzeitig aktiv sein.

Das RBAC-1-Modell erweitert das Basismodell um Rollenhierarchien. Hierdurch sollen zum einen hierarchische Geschäftsrollen abgebildet werden und zum anderen die Verwaltung von Zugriffsrechten weiter vereinfacht werden. Die Rollen in einer Hierarchie sind partiell geordnet und stehen somit in einer reflexiven, transitiven und antisymmetrischen Beziehung zueinander [SC+96]. Somit erbt eine Rolle alle Berechtigungen von sich selbst, sowie von allen Vorgängerrollen in der Hierarchie. Die Antisymmetrie verhindert die Vererbung von Berechtigungen zweier Rollen voneinander, wodurch Redundanzen entstehen können. Ebenso sind Mehrfachvererbungen möglich, mit deren Hilfe eine Rolle die Berechtigungen von zwei oder mehr Vorgängerrollen erbt. Wird einem Subjekt eine Rolle zugewiesen, so kann es eine Sitzung mit einer beliebigen Kombination von Vorgängerrollen der zugewiesenen Rolle durchführen [SC+96].

Eine weitere Variante des RBAC-Modells ist der Einsatz von Restriktionen bezüglich der Zuweisung von Subjekten zu Rollen bzw. der Zuweisung von Zugriffsrechten zu Rollen. Mit diesem RBAC-2-Modell soll verhindert werden, dass Subjekte gleichzeitig zu sich gegenseitig ausschließenden Rollen zugewiesen werden, da dies gegen die Prinzipien der Trennung von Zuständigkeiten sowie der minimalen Rechte verstößt und gegebenenfalls zu Missbrauch im System führen kann [SC+96]. Die Inhalte der Einschränkungen sind abhängig von der Umgebung, in welcher die Rollen eingeführt werden. Generelle Einschränkungen beinhaltet die Separation von sich gegenseitig ausschließenden Rollen. Einem Subjekt können hierdurch nicht zwei oder mehr Rollen zugewiesen werden, wenn diese Rollen sich gegenseitig ausschließen. Analog gilt diese Einschränkung für Zugriffsrechte. Zu weiteren Einschränkungen zählt die Restriktion der Anzahl der Subjekte, denen eine Rolle zugewiesen werden kann sowie die Zuweisung von Rollen nur unter der Voraussetzung, dass einem Subjekt bereits bestimmte Rollen zugewiesen wurden. Eine Prämisse für die Einhaltung der Einschränkungen ist unter anderem, dass Subjekte nicht mehrere digitale Identitäten beanspruchen können [SC+96].

Das RBAC-1- und das RBAC-2-Modell schließen sich nicht im Vornherein aus. Das RBAC-1-Modell lässt sich sowohl um Einschränkungen erweitern, wie in das RBAC-2-Modell Rollenhierarchien eingeführt werden können. Die Kombination dieser beiden Modell in ein RBAC-3-Modell [SC+96] erweitert die Möglichkeiten zur Spezifikation der Zuweisung von Zugriffsrechten und Rollen, wodurch jedoch auch die Komplexität des Modells und somit der Verwaltungsaufwand steigt.

Die Komposition führt auch zu konzeptionellen Fragestellungen hinsichtlich der Ausprägung des Modells, da sich das RBAC-1- und das RBAC-2-Modell stellenweise widersprechen. So muss entschieden werden, inwiefern die Einschränkungen des RBAC-2-Modells im RBAC-3-Modell auf die Rollenhierarchien erweitert werden. Weiterhing müssen die Einschränkungen, welche im RBAC-2-Modell die Zuweisung zu Rollen beschreiben, auf die Existenz von Rollenhierarchien angepasst

werden. Somit verhindern zum Beispiel Einschränkungen, welche zwei Rollen als gegenseitig ausschließend spezifizieren, die Mehrfachvererbung von diesen beiden Rollen. Ebenso wird die Anzahl der zugewiesenen Rollen nicht mehr ausschließlich durch Einschränkungen geregelt, sondern auch durch die Rollenhierarchien. Aus diesen Gründen ist das RBAC-3-Modell als eigenständige spezialisierte Alternative zu den restlichen RBAC-Modellen dargestellt. Eine Modellierung des RBAC-3-Musters mittels einer Oder-Spezialisierungsbeziehung der RBAC-1- und RBAC-2-Modelle würde diese Anforderungen nicht ausreichend ausdrücken.

Im Kontext des KIT wird im Folgenden davon ausgegangen, dass Zugriffsrichtlinien für Web-Anwendungen mittels des RBAC-1-Musters umgesetzt werden. Diese Annahme kann insofern begründet werden, das verschiedene Rollen am KIT existieren, deren Verwaltung handhabbar durchgeführt werden muss. Aufgrund der Anzahl an jährlich hinzukommenden und abgehenden Studierenden sowie der neu eingestellten bzw. aus dem Universitätsbetrieb ausscheidenden wissenschaftlichen Mitarbeitern ermöglicht die Zentralisierung der Zugriffsrechte durch das Rollenkonzept eine effiziente Rechteverwaltung.

Standardisierte rollenbasierte Zugriffsrichtlinien

Für die Spezifizierung von Zugriffskontrollrichtlinien lässt sich die von der OASIS standardisierte Extensible Access Control Markup Language (XACML [OASIS-XACML-v3.0]) verwenden. XACML stellt sowohl eine Autorisierungssprache als auch ein konzeptionelles Architekturmuster für die Umsetzung von Zugriffskontrolle dar. Die Autorisierungssprache stellt flexible Möglichkeiten zur Beschreibung von Regeln und Bedingungen für die Spezifikation der Richtlinien bereit. Insbesondere können hierfür beliebige Attribute von Subjekten und Objekten eingesetzt werden. Hierdurch ist XACML insbesondere als Implementierung des MBAC-Musters geeignet. Aufgrund der flexiblen Struktur der Autorisierungssprache lässt sich diese auch für die Spezifikation von rollenbasierten Zugriffskontrollrichtlinien einsetzen.

Im Folgenden wird hierzu das XACML-RBAC-Profil [OASIS-XACML-RBAC] für das RBAC-0- und das RBAC-1-Muster verwendet, um Idiome für die Implementierung dieser Muster in XACML zu beschreiben. Für die Umsetzung des RBAC-2- und des RBAC-3-Musters ist es nötig, Einschränkungen auf die Rollenzuweisung zu Subjekten zu deklarieren. Mittels dieses XACML-RBAC-Profils kann zwar die Rechtezuweisung zu Rollen implementiert werden. Es ist jedoch nicht vorgesehen, die Rollendefinition und die Rollenzuweisung zu Subjekten mittels XACML-Richtlinien auszudrücken. Dies wird durch eine externe Komponente, welche als Role Enablement Authority bezeichnet wird, durchgeführt.

Um rollenbasierte Autorisiererungsrichtlinien in XACML umzusetzen, sind die wesentlichen Elemente des RBAC-0-Musters, d.h. Subjekte, Rollen, Objekte, Operationen und Berechtigungen, in XACML abzubilden. Hierzu werden in dem RBAC-Profildokument von XACML bereits erste Vorgaben definiert [OASIS-XACML-RBAC]. So werden zunächst unterschiedliche Arten von Subjekten in XACML durch entsprechende XACML-Subjektdeklarationen in den Richtlinien deklariert. Zur Unterscheidung der Subjektart und zur Beschreibung des Subjektes wird das XACML-Attribut „Category“ mit entsprechenden Werten belegt. So können zum Beispiel der Nutzer, der Rechner sowie die Anwendung oder der Quellcode, welche an der Erstellung der zu überprüfenden Anfrage beteiligt sind, als Subjekte deklariert werden. Die zugehörigen XACML-Idiome sind in Abbildung 75 dargestellt.

Objekte und Operationen werden mittels XACML-Ressourcen- bzw. Aktions-Deklarationen beschrieben. Hierzu werden die generischen XACML-Sprachstrukturen verwendet, welche im XACML-Basisstandard beschrieben werden [OASIS-XACML-v3.0]. Objekt- und Aktions-deklarationen werden im RBAC-Profil hauptsächlich zur Spezifikation von Berechtigungen eingesetzt.

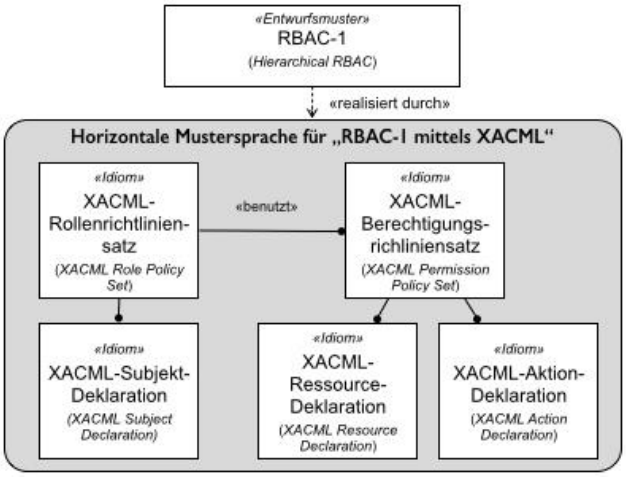

Abbildung 75: Horizontale Mustersprache für rollenbasierte Zugrifffskontrolle in XACML

Die Berechtigungen in XACML werden dabei in zwei separaten XACML-Richtlinien spezifiziert. Mittels eines sogenannten Berechtigungsrichtliniensatzes (engl. permission policy set, PPS) werden zunächst die Objekte und die darauf erlaubten Aktionen beschrieben. Anschließend wird in einem Rollenrichtliniensatz (engl. role policy set, RPS) die Zuweisung der Berechtigungen zu einer Rolle vorgenommen, in dem die für die jeweilige Rolle vorgesehenen Berechtigungsrichtliniensätze referenziert werden. Dieses Vorgehen modularisiert die Berechtigungen und Berechtigungszuweisungen und ermöglicht es, Berechtigungen für ein Objekt an mehrere Rollen zu vergeben bzw. mehrere Berechtigungen an eine Rolle zu vergeben. Die Richtliniensätze müssen dabei so abgelegt werden, dass nur die Rollenrichtliniensätze als Ausgangspunkt bei der Auswertung der Autorisierungsrichtlinien verwendet werden.

Für die Umsetzung der vom RBAC-1-Muster vorgesehenen Rollenhierarchien ist die Modularisierung von Berechtigungszuweisung zu Rollen in Rollenrichtliniensätze und Berechtigungsrichtliniensätze bereits eine große Unterstützung. Diese Unterteilung wurde bewusst gewählt, um eine Evolution der Richtlinien von einem RBAC-0- zu einem RBAC-1-Modell ohne große Änderungen an den Richtlinien vornehmen zu können [OASIS-XACML-RBAC]. Eine Vererbung von Berechtigungen von einer Juniorrolle an eine Seniorrolle wird mittels des XACML-RBAC-Profils deklariert, indem die für die Juniorrolle vorgesehenen Berechtigungsrichtliniensätze auch in dem Rollenrichtliniensatz der Seniorrolle referenziert werden. Analog kann auch eine Mehrfachvererbung umgesetzt werden, indem die Berechtigungsrichtliniensätze aller Juniorrollen referenziert werden.

Metadatenbasierte Zugriffskontrolle

Eine weitere Form einer möglichen Autorisierungsstrategie ergibt sich aus der Verwendung von Metadaten, welche über Subjekte und Objekte zur Verfügung stehen. Diese Metadaten-basierten Autorisierungsmodelle (engl. metadata-based access control, MBAC [PF+04]) finden bei einer steigenden Anzahl von zu schützenden Objekten sowie in Situationen Anwendung, in welchen der Inhalt eines Objektes und die Umgebung des Subjektes eine Rolle in der Zugriffsentscheidung spielen. Unter diesen Umständen degenerieren rollenbasierte Ansätze im schlechtesten Fall zu DAC-Ansätzen, da die Komplexität zu einer deutlichen Erhöhung von Rollen mit jeweils nur relativ kleinen Zugriffsrechten führt [YT+05]. Im Folgenden wird aufgrund der aktiven Behandlung des Themas in der Fachliteratur die Einordnung dieser Ansätze in einen Musterbaum vorgenommen. Abbildung 76 zeigt dabei eine Auswahl an betrachteten Ansätzen.

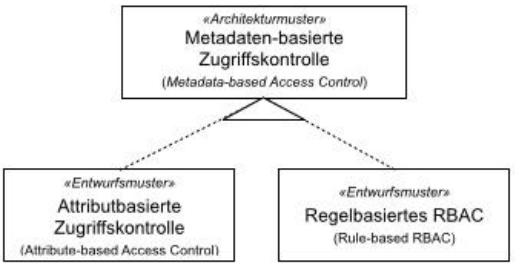

Abbildung 76: Realisierungen des Musters Metadatenbasierte Zugriffskontrolle

Als Abgrenzung zu RBAC werden in MBAC-Modellen nicht mehr Zugriffsrechte anhand von Aufgabenbereichen gebündelt und Subjekten zugewiesen, sondern anhand der Eigenschaften von Subjekten und Objekten bestimmt. Hierzu ist es nötig, dass die Eigenschaften von Subjekten, Objekten und der Umgebung statisch und gegebenenfalls auch zur Laufzeit dynamisch erfasst werden. Eigenschaften werden als Name-Wert-Paare spezifiziert, wobei der Eigenschaftsname zur Spezifikation der Autorisierungsrichtlinien verwendet und bei der anschließenden Auswertung mit dem derzeitigen Wert ersetzt wird. Die einstufige Indirektion von RBAC wird bei MBAC um eine zweite Indirektionsstufe erweitert, indem sowohl Deskriptoren für Subjekt- als auch für Objekteigenschaften für die Autorisierung verwendet werden. In diesen Deskriptoren werden mittels der Eigenschaften Bedingungen spezifiziert, welche auf ein oder mehrere Subjekte bzw. Objekte zutreffen und wodurch implizit Gruppen von Subjekten und Objekten beschrieben werden [PF+04].

Im Bereich von Web-Services wurde in [YT+05] ein konkretes attributbasiertes Autorisierungsmodell (engl. Attribute-based Access Control, ABAC) vorgestellt. Hierbei werden die Zugriffsrechte ebenfalls von den zu schützenden Objekten bzw. den zugreifenden Subjekten entkoppelt und in dynamische Zugriffskontrollrichtlinien zusammengefasst. Diese Richtlinien nutzen dabei Regeln, welche Subjekt- und Objektinformationen in Form von Attributen beinhalten. Somit ist nicht nur die Identität eines Subjektes bzw. eines Objektes für die Zugriffskontrollentscheidung ausschlaggebend, sondern alle vorhandenen Informationen über ein Subjekt oder ein Objekt können mit in die Spezifikation der Zugriffskontrollrichtlinien einfließen.

Al-Kathani und Sandhu nutzen die Flexibilität eines MBAC-Ansatzes in Kombination mit einem rollenbasierten Zugriffskontrollmodell, um anhand von Subjekteigenschaften eine regelbasierte Zuweisung von Rollen zu den Subjekten vorzunehmen (engl. Rule-based RBAC, RB-RBAC) [AS02]. Hierzu werden die Eigenschaften von Subjekten definiert und anschließend analog zu den Deskriptoren im allgemeinen MBAC-Muster in Bedingungen verwendet. Diese Bedingungen korrelieren mit ein oder mehreren Rollen, welche zugewiesen werden, sobald die Bedingungen erfüllt sind. Da die Bedingungen zur Laufzeit ausgewertet werden, können Rollenzuweisungen aufgehoben werden, sobald die notwendigen Bedingungen nicht mehr erfüllt sind.

Dies stellt somit zunächst eine Einschränkung des allgemeinen MBAC-Musters dar, da lediglich die Subjekteigenschaften verwendet werden. Betrachtet man die Zuweisung von Zugriffsrechten auf Objekte zu Rollen als Bedingung, welche auf statischen Objekteigenschaften beruht, lässt sich die Einordnung dieses Musters als Spezialisierung des MBAC-Musters rechtfertigen. Gleichzeitig ist es auch eine Weiterentwicklung des zuvor besprochenen allgemeinen RBAC-Musters, welches ein Argument für die Einordnung als Spezialisierung in der vertikalen Mustersprache von RBAC ist. Da jedoch das allgemeine RBAC-Muster nicht die dynamische Zuweisung von Rollen anhand von Attributeigenschaften berücksichtigt, wurde im vorliegenden Fall die Einordnung als MBAC-Musterspezialisierung vorgesehen.

6.3.4 Zugriffskontrollsprache

Mittels der Autorisierungsmustersprache werden Entwickler bei der Auswahl und Umsetzung eines passenden Modells für die Spezifikation von Autorisierungsrichtlinien gemäß des Kontextes der zu entwickelnden Anwendung unterstützt. Das Ergebnis ist hierbei ein Regelsatz, welcher für jeden Nutzer eines Software-Systems die ihm zustehenden Rechte bezüglich der auf Ressourcen ausführbaren Aktionen spezifiziert. Die Einhaltung dieser Richtlinien sollte nicht den Subjekten selbst überlassen werden, da diesen im Allgemeinen nicht vertraut werden kann. Daher wird komplementär zur Autorisierung die vorbeugende Taktik der Zugriffskontrolle eingesetzt. Das Ziel der Zugriffskontrolle ist im Gegensatz zur Autorisierung die Umsetzung der Autorisierungsrichtlinien zur Laufzeit eines Systems. Hierzu sind in erster Linie Architekturkomponenten notwendig, welche die Richtlinien auswerten und den Schutz der Ressourcen im Software-System durchführen.

Dabei ist zu beachten, dass die Komponenten zunächst weitestgehend unabhängig von dem gewählten Autorisierungsmodell sind. Es wird davon ausgegangen, dass eine generische Implementierung der Komponenten lediglich die Auswertung der Richtlinien durchführt. Dennoch existiert eine Abhängigkeit zu dem letztendlich eingesetzten Autorisierungsmodell und der Zugriffskontrollkomponente, welche diese Richtlinien auswertet und umsetzt. Diese Abhängigkeit wird jedoch in dem Mustersprachenbaum auf die Assoziation zwischen den abstrakten Mustern Zugriffskontrolle und Autorisierung beschränkt und wird nicht bis auf eine technologiespezifische Ebene mitgeführt. Die Semantik der Assoziation auf abstrakter Ebene ist somit nicht nur beschränkt auf die Aussage, dass Zugriffskontrolle eine Autorisierung voraussetzt. Im erweiterten Sinne umfasst diese Aussage alle Abstraktionsebenen, sodass eine auch technologiespezifische Implementierung von Zugriffskontrolle die Details der technologiespezifischen Implementierung von Autorisierung berücksichtigen sollte.

Architektur und Komponenten

Ausgehend von dem abstrakten Muster bzw. der vorbeugenden Taktik der Zugriffskontrolle, können zwei Architekturmuster identifiziert werden, welche auf einer sehr grobgranularen Architekturebene die Kapselung von Sicherheitsfunktionalität in einem Software-System behandeln. Zum einen wurde bereits mit dem Muster „Zentraler Zugangspunkt“ (engl. Single Access Point, [SF+05:279]) ein architekturelles Authentifizierungsmuster vorgestellt, mit welchem externe Benutzer identifiziert und authentifiziert und die Interaktionsmöglichkeiten mit einem Software-System eingeschränkt werden können. Im Bereich der Zugriffskontrolle lässt sich zum anderen das Muster „Kontrollpunkt“ (engl. Check Point [SF+05:287]) komplementär dazu einsetzen, die vom zentralen Zugriffspunkt durchgeführten Sicherheitsüberprüfungen flexibel und erweiterbar zu gestalten. Der kombinierte Einsatz dieser Sicherheitsmuster ermöglicht den Schutz eines Software-Systems vor unautorisiertem Zugriff. In Abbildung 77 sind die möglichen Bestandteile des Kontrollpunkt-Musters in einer horizontalen Mustersprache wiedergegeben.

Zentrales Merkmal des abstrakten Kontrollpunkt-Musters ist die Zentralisierung von Sicherheitsfunktionalität wie zum Beispiel Zugriffskontrollfunktionalität in einem Software-System. Mittels des Checkpoint-Musters soll eine erweiterbare bzw. änderbare Umsetzung einer geschäftlichen Sicherheitsrichtlinie ermöglicht werden. Hierzu werden mittels des Kontrollpunkts verschiedene Sicherheitskontrollen umgesetzt, zu welchen auch die vom Muster „Zentraler Zugangspunkt“ vorgenommene Identifizierung und Authentifizierung gehört. Die Struktur des Kontrollpunktes sieht vor, dass eine einheitliche Schnittstelle bereitgestellt wird, welche von der Komplexität dieser Sicherheitskontrollen abstrahiert. In diesem Sinne entspricht die Schnittstelle des Kontrollpunktes der Schnittstelle im Strategie-Entwurfsmuster [GH+94]. Konkrete Implementierungen der Kontrollpunkt-Schnittstelle stellen unterschiedliche Strategien für die Umsetzung einer Sicherheitsrichtlinie bereit. So ist es

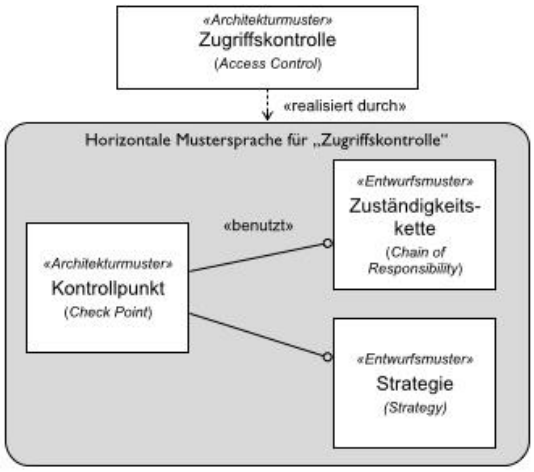

Abbildung 77: Realisierung der Zugriffskontrolle mittels des Kontrollpunkt-Musters

denkbar unterschiedliche Implementierungen von Zugriffskontrolle, zum Beispiel durch Einsatz verschiedener Autorisierungsmodelle, je nach Bedarf einzusetzen.

Optional lässt sich das Kontrollpunkt-Muster durch den Einsatz des Musters Zuständigkeitskette (engl. Chain of Responsibility [GH+94] [BR+99]) erweitern [SF+05:292ff]. Somit wird es möglich, unterschiedliche Sicherheitsüberprüfungen je nach Kontext in unterschiedlicher Kombination gemeinsam zu benutzen. Hierzu wird die Sicherheitsüberprüfung an jeden konkreten Bearbeiter in der Zuständigkeitskette weitergereicht, bis eine erfolgreiche Abarbeitung bzw. endgültige Ablehnung der Anfrage an den Kontrollpunkt erfolgt. Jeder konkrete Bearbeiter implementiert somit eine Sicherheitsüberprüfung bzw. unterschiedliche Formen einer Sicherheitsüberprüfung. Auf die Zugriffskontrolle bezogen, ist es somit möglich, unterschiedliche Autorisierungsmodelle zu unterstützen. Beispielsweise ließe sich eine Zuständigkeitsliste mit einem Zugriffskontrolllisten-Bearbeiter und einem Bell-LaPadula-Bearbeiter implementieren. Ersterer überprüft die Berechtigungen für den Benutzer anhand einer vorliegenden Zugriffskontrollliste während Letzterer die Berechtigungen anhand einer systemweiten Klassifizierung überprüft. Auf diese Art und Weise könnte eine kombinierte DAC-MAC-Autorisierungsrichtlinie implementiert werden, in welcher die Zugriffskontrollkomponenten lose gekoppelt sind. Auch die Umstellung auf eine andere Autorisierungsform, wie zum Beispiel einer rollenbasierten Autorisierung ließe sich durch den Austausch der Bearbeiter in der Zuständigkeitsliste des Kontrollpunktes umsetzen.

Während das Kontrollpunkt-Muster die Zentralisierung und den flexiblen Austausch von Sicherheitsüberprüfungen fokussiert, wird die Einbindung des Kontrollpunktes in andere fachliche Komponenten eines Software-Systems vernachlässigt. Die Spezialisierung des Kontrollpunkt-Musters mittels des Musters Referenzmonitor (engl. Reference Monitor, [SF+05:256]) behandelt diesen Punkt in der Hinsicht, dass jede Anfrage an eine geschützte Ressource zunächst abgefangen und die Autorisierung des Subjektes überprüft wird. Dementsprechend wird im Gegensatz zur der relativ freien Auswahl an durchzuführenden Sicherheitsprüfungen im Kontrollpunkt-Muster lediglich die Zugriffskontrolle beim Referenzmonitor als Prüfungsmethode vorgesehen. In Abbildung 78 wird die Beziehung zwischen Kontrollpunkt-und Referenzmonitor-Muster dargestellt.

Das Muster schreibt weiter vor, dass das Abfangen von Anfragen auf alle Objekte in einem Software-System auszuweiten ist, welche durch Subjekte angefordert werden können. Somit lässt sich das Referenzmonitor-Muster nicht nur auf die Interaktionen zwischen einem menschlichen Benutzer und einem Software-System anwenden, sondern es ermöglicht auch die autorisierten Zugriffe zwischen den Komponenten eines Software-Systems selbst. Dies ist insbesondere vorteilhaft, falls

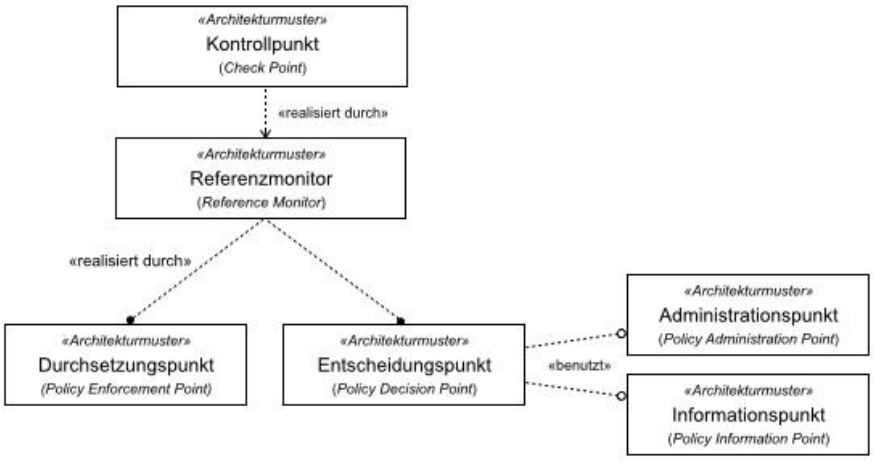

Abbildung 78: Spezialisierung des Kontrollpunkt-Musters

Komponenten von Dritten eingesetzt werden und deren Zugriffsmöglichkeiten im System begrenzt werden sollen. Da Anfragen an beliebige Objekte bzw. Ressourcen abgefangen werden sollen, stellt ähnlich wie beim Kontrollpunkt-Muster das Referenzmonitor-Muster zunächst nur eine Schnittstelle und gegebenenfalls eine abstrakte Implementierung bereit. Aufgrund der Eigenschaften von Software-Ressourcen, wie zum Beispiel einer Web-Seite, Methodenaufrufe oder Dateisystemzugriff, sind konkrete Implementierungen an die jeweiligen Ressourcen anzupassen, um die Anfragen abfangen zu können.

Typischerweise muss bei der Implementierung eines Referenzmonitors lediglich die Logik zum Abfangen von Anfragen an Objekte implementiert werden, wogegen die Logik zur Auswertung von Autorisierungsrichtlinien unabhängig von spezifischen Objekten ist. Die Separation dieser beiden Funktionen in eigenständige Komponenten wird in den Mustern „Durchsetzungspunkt“ (engl. Policy Enforcement Point, PEP) und „Entscheidungspunkt“ (engl. Policy Decision Point, PDP), wie in Abbildung 78 dargestellt, berücksichtigt [ES+06]. In dieser Aufgabenteilung ist der PEP für das Abfangen von Anfragen an ein Objekt sowie für die Umsetzung der Zugriffskontrollentscheidung zuständig. Zudem kapselt die PEP-Komponente die Kommunikationslogik, welche zur Interaktion mit dem PDP benötigt wird. Die PEPs sollten als leichtgewichtige Komponenten implementiert werden, welche die Anfrage effizient an den PDP weiterleiten und bei einem autorisierten Zugriff die Anfrage von der fachlichen Komponente weitergeben.

Der PDP ist somit für die Auswertung von Autorisierungsrichtlinien zuständig und stellt die Zugriffskontrollentscheidung über eine Schnittstelle zur Verfügung. Zudem werden die Richtlinien von der PDP-Komponente verwaltet. Diese Separation der Komponenten ermöglicht die Auslagerung und zentrale Verwaltung von Autorisierungsregeln sowie der dazugehörigen Auswertungslogik in den PDP. Somit kann der PDP auch als eine physisch vom Software-System entfernte Komponente umgesetzt werden. Die Kommunikation zwischen PEP und PDP basiert in diesem Fall auf dem Client-Server-Prinzip. Eine weitere Verfeinerung des allgemeinen Referenzmonitor-Musters ist die Separation der Auswertungslogik für Richtlinien von der Logik zum Abfangen der Anforderung von der der Umsetzung der Zugriffskontrollentscheidung an einer Ressource.

Das PEP/PDP-Muster wird auch von dem XACML-Standard aufgegriffen und weiter verfeinert. Insbesondere wird die konzeptionelle Architektur des PDP detailliert beschrieben. Hierzu werden zwei weitere Komponenten eingeführt, welche Aufgaben übernehmen, die im abstrakten PEP-Muster noch durch den PEP selbst durchgeführt werden mussten. Zum einen wird die Verwaltung der Richtlinien in eine eigenständige Komponente ausgelagert. Diese Komponente zur Richtlinienadministration (engl.

Policy Administration Point, PAP) übernimmt somit die Ablage von XACML-Richtlinien und bietet dem PDP eine Schnittstelle an, mit Hilfe derer der PDP für eine Zugriffskontrollentscheidung notwendige XACML-Richtlinienabfragen kann [OASIS-XACML-v3.0]. Zum anderen wird das PEP/PDP-Muster um eine Informationskomponente (engl. Policy Information Point, PIP) erweitert, welche aus verschiedenen Datenquellen dem PDP auf Anfrage Information bereitstellt, welche dieser für die Auswertung der XACML-Richtlinien benötigt. Da der XACML-Standard die Auswertung von Attributen von Subjekten, Objekten sowie der Umgebung der Anfrage berücksichtigt, greift der PIP auf entsprechende Datenquellen zu, welche diese Attributdaten beinhalten.

Feinentwurf der Zugriffskontrollkomponenten

Während der XACML-Standard die Struktur des PDP verfeinert, werden keine spezifischen Details über den PEP beschrieben. Grund hierfür ist die schon besprochene Abhängigkeit des PEP zur jeweiligen Art des zu schützenden Objektes, wodurch eine standardisierte Beschreibung der PEP-Struktur unpraktikabel wird. Trotzdem haben sich generische Entwurfsmuster zur Umsetzung eines PEP als nützlich erwiesen. Wie schon für das allgemeine Referenzmonitor-Muster beschrieben [SF+05:258], lässt sich das Entwurfsmuster „Interceptor“ [GH+94] für das Abfangen von Anfragen an zu schützende Objekte einsetzen.

Ebenso können auch andere Entwurfsmuster für diesen Zweck eingesetzt werden. Insbesondere eigenen sich Strukturmuster wie zum Beispiel „Stellvertreter“ (engl. Proxy [GH+94]), „Fassade“ (engl. Façade [GH+94]), „Kompositum“ (engl. Composite [GH+94]) und „Dekorierer“ (engl. Decorator [GH+94]) dazu, die PEP-Logik von der fachlichen Logik strukturell zu trennen und Anfragen an Objekte abzufangen. Momentan lässt sich eine Einteilung von PEP-Strukturen in eine verhaltensbasierte sowie eine strukturbasierte Implementierung vornehmen. Zum ersteren zählt unter anderem der Einsatz des Interceptor-Musters während zum letzteren die genannten Strukturmuster gehören. In Abbildung 79 sind mögliche Entwurfsmuster zur Implementierung von PEPs im Kontext von Java-basierten Web-Anwendungen aufgezeigt.

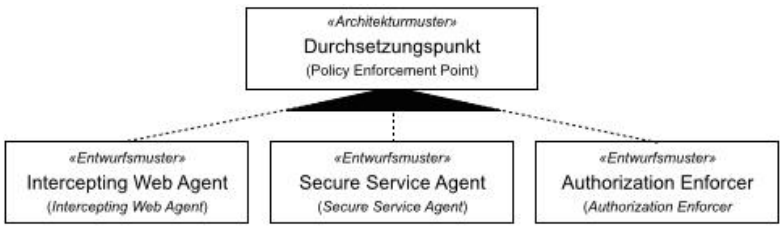

Abbildung 79: Implementierungsoptionen für das Durchsetzungspunkt-Muster

Das Muster „Intercepting Web Agent“ (IWA [CN+05:606ff]) stellt eine Implementierung eines PEPs für existierende Web-Anwendungen dar, für welche eine Authentifizierung bzw. Zugriffskontrolle nachgerüstet werden soll, da die Anwendung nicht geändert werden soll bzw. kann. Hierzu wird der IWA in die Umgebung der Web-Anwendung, wie zum Beispiel dem Web-Server, installiert und so konfiguriert, dass die eingehenden Anfragen an die Web-Anwendung abgefangen und an einen PEP weitergeleitet werden. Der Vorteil des IWA liegt in der klaren Trennung der Sicherheitsfunktionalität von der Anwendungslogik.

Auf der Ebene einzelner Komponenten bietet das Muster „Authorization Enforcer“ [CN+05:548ff] eine Möglichkeit zur Überprüfung von Zugriffsrechten. Hierzu werden die betreffenden Komponenten zur Entwicklungszeit direkt verändert und um zusätzliche Befehle ergänzt, so dass eine Zugriffskontrolle zur Laufzeit durchgeführt werden kann.

Ein Secure Service Agent (SSA [ES+06]) bietet eine hybride Lösung zwischen IWA und Authorization Enforcer an. Ein SSA ist ebenfalls zur Absicherung von Komponenten eines Software-Systems vorgesehen. Hierzu wird der SSA selbst als eine Komponente mit einer klaren Schnittstelle realisiert, wodurch diese bei Bedarf von den fachlichen Komponenten aufgerufen werden kann, um eine Zugriffskontrollprüfung anzustoßen.

6.4 Resümee

Ausgehend von dem im Kapitel 4 beschriebenen sicherheitsbasierten Entwicklungsprozess wurde in diesem Kapitel die Aufbereitung von existierendem Sicherheitswissen für den Entwurf von Sicherheitsmaßnahmen in Form von Sicherheitsmustern thematisiert. Sicherheitsmuster wurden dabei als geeignetes Mittel zur Dokumentation von bewährten Lösungsverfahren ausgewählt, um eine Integration in einen fachlichen Entwicklungsprozess zu ermöglichen. Der kombinierte Einsatz von Sicherheitsmustern stellt jedoch eine Herausforderung dar, da die Abhängigkeiten zwischen den Mustern bisher nicht eindeutig spezifiziert wurden.

Mittels des vorgestellten Variabilitätsmodells wurde ein Rahmen für die Kategorisierung und eine Verknüpfung der Sicherheitsmuster geschaffen, so dass deren Einsatz zur Modellierung von Sicherheitsmaßnahmen die strukturierte Umsetzung von Sicherheitsanforderungen ermöglicht und die Ableitung auf die bestehende Sicherheitstechnologie unterstützt. Das Variabilitätsmodell definiert hierzu zunächst verschiedene Abstraktionsstufen, durch welche die Nähe der Sicherheitsmuster zu abstrakten Sicherheitsanforderungen bzw. konkreten Sicherheitstechnologien ausgedrückt werden kann. Durch abstrakte Muster wird zudem eine Modularisierung der einzelnen Sicherheitsbereiche durchgeführt, wodurch eine unabhängige Kategorisierung und Weiterentwicklung der Beziehungen zwischen Sicherheitsmustern ermöglicht wird.

Des Weiteren wurden unterschiedliche Beziehungstypen zwischen den Sicherheitsmustern identifiziert, mittels welchen Abhängigkeiten zwischen Mustern der gleichen Abstraktionsstufe sowie Spezialisierungsbeziehungen zu Mustern einer tieferen Abstraktionsstufe modelliert werden können. Hieraus entstehen horizontale sowie vertikale Mustersprachen, welche den gemeinsamen Einsatz von Sicherheitsmustern sowie deren Spezialisierung beschreiben. Durch die vertikalen Mustersprachen werden Musterbäume gebildet, welche die Ableitung von abstrakten Sicherheitsmaßnahmen auf technologienahe Muster bzw. Idiome ermöglichen. Des Weiteren wurden Beziehungstypen definiert, welche die Spezifikation von alternativen Lösungsmöglichkeiten erlauben, was eine Auswahl zwischen möglichen Maßnahmen gestattet.

Die betrachteten Sicherheitsmuster werden in dem Variabilitätsmodell durch die Berücksichtigung einer bestehenden Sicherheitsinfrastruktur eingeschränkt, wodurch das in Kapitel 4 vorgestellte Referenzmodell erfüllt wird. Hierdurch werden Sicherheitsmuster berücksichtigt, welche eine Umsetzung in den vorhandenen Sicherheitsprodukten der Infrastruktur finden. Somit lässt sich beim Einsatz der Sicherheitsmuster ein entwickeltes Software-System in diese Infrastruktur einbetten. Die Aufbereitung von Sicherheitsmustern unter Berücksichtigung einer existierenden Sicherheitsinfrastruktur wurde anhand von Sicherheitsmaßnahmen für Authentifizierung, Autorisierung und Zugriffskontrolle für das Karlsruher Institut für Technologie (KIT) durchgeführt. Hierbei wurden bekannte Sicherheitsmuster unter Einsatz des Variabilitätsmodells kategorisiert und in Beziehung gesetzt, um eine Ableitung von Sicherheitsanforderungen auf Sicherheitstechnologien zu demonstrieren.

7 Tragfähigkeitsnachweise

Die in dieser Arbeit entwickelten Konzepte für ein sicherheitsbasiertes Entwicklungsvorgehen werden im folgenden Kapitel anhand von zwei Software-Systemen zur Entwicklung von deren Sicherheitsfunktionalität angewendet. Das Ziel der Demonstration ist dabei der Nachweis der Anwendbarkeit und des effektiven Nutzens der erbrachten Beiträge anhand von realen Anwendungsszenarien. Die dazu untersuchten Software-Systeme stammen aus unterschiedlichen fachlichen Domänen und beschreiben somit unterschiedliche Interessens- und Wissensgebiete [SV+07]. Hierdurch wird zudem der Aspekt der Wiederverwendbarkeit der Ergebnisse auf unterschiedliche Domänen aufgezeigt.

Durch die praktische Anwendung der vorgestellten Aktivitäten und Artefakte auf die konkreten Szenarien wird nachgewiesen, dass die vorgestellten Beiträge die Entwicklung von sicheren Software-Systemen unterstützen, indem praktikable Werkzeuge für Entwickler bereitgestellt werden. Durch die zu jedem Zeitpunkt eines Entwicklungsprozesses klar definierten Aktivitäten sowie die einzusetzenden Entwicklungsartefakte definiert sich die Praktikabilität des Entwicklungsvorgehens. Hierdurch wird eine durchgängige und systematische Entwicklung von Sicherheitsfunktionalität ausgehend von den spezifizierten Sicherheitsanforderungen und modellierten Sicherheitsmaßnahmen erreicht.

Die Erfüllung der weiteren in Kapitel 3.1 aufgestellten Anforderungen Wiederverwendung, Integrationsfähigkeit und Nachvollziehbarkeit beschreibt die Effektivität der erbrachten Beiträge. Hierzu werden die durchgeführten Aktivitäten und die Anwendung sowie der Einsatz von Entwicklungsartefakten nach jeder Phase analysiert und das resultierende Ergebnis bewertet. Die Bewertung findet dabei unabhängig von den zuvor eingesetzten Kriterien zur Erstellung der Beiträge statt. Hierdurch wird verhindert, dass die für die Erstellung der Beiträge dieser Arbeit eingesetzten Mittel und Anforderungen selbst wieder zur Einschätzung der Ergebnisse eingesetzt werden.

Als Beispiel wurden für die Anwendungsszenarien Software-Systeme zu Demonstrationszwecken ausgewählt, welche an der Forschungsgruppe Cooperation und Management (C&M) von Prof. Dr. Sebastian Abeck am Karlsruher Institut für Technologie (KIT) entwickelt wurden. Bei diesen Systemen handelt es sich zum einen um ein geographisches Umweltinformationssystem (UIS) zur Unterstützung von Mitarbeitern und Studierenden des KIT auf dem Campusgelände. Zum anderen wurde ein auf dem Paradigma des Internets-der-Dinge basierendes Software-System betrachtet, welches zur autonomen Raumverwaltung entwickelt wurde.

Im Abschluss dieses Kapitels wird die Umsetzung der Beiträge anhand deren praktischer Anwendung analysiert. Dieser Abgleich zwischen den erfüllten und den gestellten Anforderungen ermöglicht die Ermittlung der Effektivität und Praktikabilität der Beiträge. Ebenso wird ein Vergleich mit den in Kapitel 3 beschriebenen alternativen Ansätzen auf Basis der Erfahrungen durchgeführt. Der durch die Beiträge der vorliegenden Arbeit erzielte Mehrwert wird hierdurch betont.

7.1 KITCampusGuide – Sicherheit in dienstorientierten Umweltinformationssystemen

Geographische Informationen spielen eine immer größer werdende Rolle in Informationssystemen. Insbesondere erfreuen sich kartenbasierte Anwendungen großer Beliebtheit. Kommerzielle Anwendungen, wie zum Beispiel Google Maps [GOOGLE-MAPS], Microsoft Bing Maps [BING-MAPS] oder Yahoo Maps [YAHOO-MAPS], sowie quelloffene und gemeinschaftsbasierte Projekte, wie zum Beispiel Openstreetmap [OSM], stellen vielfältige ortsbezogene Informationen bereit und lassen sich durch Programmierschnittstellen (engl. application programming interface, API) in

beliebige Anwendungen integrieren. Durch die zunehmende Verbreitung von mobilen Endgeräten treten bei der Nutzung von diesen Software-Systemen Fragestellungen über den Schutz der Privatsphäre auf.

Im Rahmen dieser Arbeit wurde eine solches geographisches Software-System, der KITCampusGuide (KCG), hinsichtlich ihrer Sicherheitsfunktionalität untersucht. Das Ziel dieses Software-Systems ist es, für die deutschlandweit verteilten Standorte des KIT eine kartenbasierte Ansicht zu liefern, auf welcher Suchanfragen für bestimmte Standorte, sogenannten Points of Interest (POI), angezeigt und gegebenenfalls auch Routen zu diesen gesuchten POIs angezeigt werden können.

Im Folgenden werden die für das weitere Verständnis notwendigen Zielsetzungen und Funktionalitäten des KITCampusGuides beschrieben. Anschließend wird die Entwicklung der Sicherheitsfunktionalität anhand des beschriebenen Entwicklungsvorgehens und dem Einsatz der Entwicklungsartefakte demonstriert. Für die Entwicklung des KITCampusGuide wurde ein traditioneller Software-Entwicklungsprozess eingesetzt, welcher dem Wasserfall-Modell folgt. Somit wird ebenfalls die Anpassung der Aktivitäten des Entwicklungsvorgehens auf dieses Prozessmodell vorgeführt.

7.1.1 Analyse der Sicherheitsanforderungen

Zunächst wird die in Kapitel 4 beschriebene Analyse von Sicherheitsanforderungen gemäß den dort vorgestellten Aktivitäten des Entwicklungsvorgehens für das KITCampusGuide-System durchgeführt. Hierbei werden die in Kapitel 5 vorgestellten Vorlagen für Sicherheitsanforderungen eingesetzt, um deren Einsatz zur effizienten Spezifikation der Anforderungen zu demonstrieren. Wie in Kapitel 4 beschrieben, ist die Analyse von Sicherheitsanforderungen eng mit der Erhebung von funktionalen Anforderungen eines Software-Systems verknüpft. Daher wird zunächst ein kurzer Überblick über die relevante Funktionalität des KITCampusGuide gegeben, welche in der Anforderungserhebung betrachtet wird.

Überblick KITCampusGuide

Die generelle Zielsetzung des KITCampusGuides ist die Unterstützung von täglichen Aktivitäten von Studierenden, Mitarbeitern und Gästen des KIT durch die Bereitstellung von Informationen über wichtige Lokalitäten (engl. Points of Interests, POI) auf den geografisch verteilten Campus des KIT mittels einer kartenbasierten Darstellung. Zu diesen POIs zählen in erster Linie Hörsäle und Seminarräume, in welchen Vorlesungen, Übungen, etc. stattfinden. Zur Nutzung von Sprechzeiten sind des Weiteren die Position der Büros von Dozenten und Mitarbeitern auf dem Campus relevant. Neben der Suche nach POIs und der Darstellung in einer Karte soll vor allem eine Unterstützung bei der Orientierung und Navigation auf dem Gelände durch den KITCampusGuide ermöglicht werden.

Abbildung 80 zeigt eine Übersicht der Funktionalität des KITCampusGuide in Form eines UML-Anwendungsfalldiagramms. Die Suche nach POIs sowie deren Anzeige auf einer Karte ist dabei allen Benutzern des KITCampusGuides möglich. Dagegen ist es nur bestimmten Personen gestattet, POI-Daten in das System einzutragen und diese zu modifizieren. Diese Benutzer werden im Diagramm als Akteur „POI-Verwalter“ dargestellt. Ebenso soll den Benutzern eine Navigation zu bestimmten POIs ermöglicht werden, indem eine Route auf der Karte dargestellt wird und diese bei den Bewegungen des Nutzers aktualisiert wird. Hierzu ist die regelmäßige Bestimmung der aktuellen Position eines Benutzers vorgesehen, um eine Navigationsunterstützung umzusetzen.

Identifizierung von Ressourcen

Gemäß den in Kapitel 4.3 vorgestellten Aktivitäten für die Analyse von Sicherheitsanforderungen werden zunächst die schützenswerten Ressourcen identifiziert. Im Falle des KITCampusGuide sind vor allem die Informationen zu POIs sowie die Koordinaten des Standortes von Benutzern zu

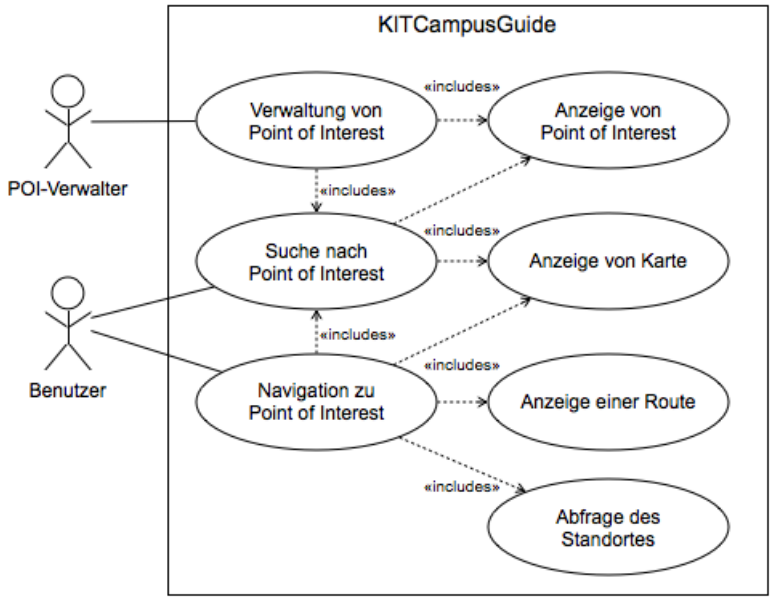

Abbildung 80: Untersuchte Anwendungsfälle des KITCampusGuide

berücksichtigen. Die POI-Daten repräsentieren die öffentlich zugänglichen Informationen, welche von Benutzern gesucht werden können und auf der Karte dargestellt werden. Ausgehend von dem Ziel des KITCampusGuide, die Unterstützung der Benutzer durch Bereitstellung der POI-Daten, kann davon ausgegangen werden, dass sich die Benutzer auf die Korrektheit der POI-Daten verlassen. Aus Sicht der Benutzer stellen die POI-Daten somit wertvolle Ressourcen dar, da aus ihnen ein persönlicher Nutzen gezogen wird.

Durch die POI-Daten des KITCampusGuide wird die aktuelle Situation am KIT, zum Beispiel die Belegungspläne von Hörsälen, wiedergegeben. Durch diese direkte Widerspiegelung stellt die Korrektheit und Aktualität der POI-Daten einen Wert für die POI-Verwalter dar, da diese für die Verwaltung und Bereitstellung der POI-Daten zuständig sind. Der Wert ist hierbei mit der Reputation des öffentlichen Bildes des KITs verknüpft.

Des Weiteren wird der aktuelle Standort eines Benutzers bei der Navigationsunterstützung verarbeitet und kann daher als Ressource des Software-Systems angesehen werden. Um einen Benutzer zu einem vorgegebenen Ziel zu leiten, ist es nötig, dass der KITCampusGuide in regelmäßigen Abständen die Position eines Benutzers mit der vorberechneten Route vergleicht. Der Abruf der Position erfolgt mit Hilfe von Empfängern für das Global Positioning System (GPS), welche zum Beispiel in einem modernen Mobiltelefon vorhanden sind, oder über eine Ortung mittels WLAN. Für einen Benutzer stellt dessen derzeitiger Aufenthaltsort eine wertvolle Information dar, da hierdurch eine persönliche Information des Nutzers an das Software-System weitergegeben wird, um einen persönlichen Mehrwert zu schaffen. Daher sollte diese Information nicht durch das Software-System bzw. durch Angreifer missbraucht werden.

Ermittlung des Schutzbedarfes

Da die identifizierten Ressourcen einen Wert für die beteiligten Personen darstellen, wird als nächster Schritt in der Anforderungserhebung deren Schutzbedarf festgelegt. Hierzu wird die Einteilung von Schutzbedarfskategorien, wie sie im Grundschutzhandbuch des BSI festgelegt ist, verwendet [Ec09:177ff]. Zur Bestimmung der zugehörigen Kategorie werden Schadensszenarien auf ihre Anwendbarkeit in dem konkret vorliegenden Fall des KITCampusGuides überprüft.

Für die Informationen zu POIs lässt sich zum einen das Schadensszenario der Beeinträchtigung der Aufgabenerfüllung anwenden. Hierunter fallen zum Beispiel Szenarien, in welchen Termine durch die Anzeige falscher POI-Informationen durch das KITCampusGuide-System versäumt werden oder sich verzögern. Zum anderen kann das Schadensszenario mit negativen Auswirkungen angewendet werden, da durch fehlerhafte POI-Daten der KITCampusGuide als fehlerhaft aufgefasst und nicht mehr von den Benutzern eingesetzt wird. Im Allgemeinen kann hierunter die gesellschaftliche Stellung des KIT bei den Benutzern leiden und somit Ansehensverluste eintreten. Da jedoch dieser Vertrauensverlust als geringfügig und als zu verkraften angesehen sowie die gegebenenfalls falsche oder nicht verfügbare Darstellung von POI-Information für den Benutzer toleriert werden kann, ist eine Einordnung der POI-Daten gemäß dem Grundschutzhandbuch in die Schutzbedarfskategorie niedrig bis mittel vorzusehen.

Im Falle der Erfassung der GPS-Daten des Benutzers ist ein Schadensszenario bezüglich der Beeinträchtigung des informationellen Selbstbestimmungsrechts vorstellbar. Dies ist gemäß dem Szenario der Fall, wenn die Daten ohne Einwilligung erhoben oder unbefugt weitergegeben wurden sowie wenn die Daten zu einem anderen Zweck als bei der Einwilligung zugestimmt wurde, verwendet werden. Die Auswirkungen in diesem Fall halten sich jedoch in Grenzen, da die Erfassung der GPS-Daten auf dem Campus beispielsweise nur geringfügige Effekte auf die gesellschaftliche Stellung oder die wirtschaftlichen Verhältnisse des Benutzers haben. Ebenso ist der Vertrauensverlust gegenüber der Institution des KIT als gemäßigt anzusehen. In diesem Fall setzt das Grundschutzhandbuch einen Schutzbedarf der Kategorie niedrig bis mittel fest, welcher daher für die GPS-Daten angewendet wird.

Festlegung von Schutzzielen

Nach der Identifizierung von Ressourcen und deren Schutzbedarf besteht der nächste Schritt in der Festlegung von Schutzzielen. Der Schutzbedarf wird dabei durch konkrete Forderungen ausgedrückt, welche zur Erhaltung des Wertes der Ressourcen beitragen sollen. Hierdurch wird zum einen eine weitere Filterung der Vorlagen für Sicherheitsanforderungen durchgeführt, welche im nachfolgenden Schritt zu einer Menge von zu berücksichtigenden Bedrohungen und Angriffen führt. Zum anderen wurde durch die Festlegung eines niedrig bis mittleren Schutzbedarfes bereits eine Menge an Vorlagen gefiltert, deren Sicherheitsanforderungen für mittlere bis hohe Schutzbedürfnisse bestimmt sind. Hierdurch lassen sich die verbleibenden Vorlagen nutzen, in dem die in den Vorlagen hinterlegten Schutzziele auf die Anwendbarkeit für die identifizierten Ressourcen des KITCampusGuide überprüft werden.

Im Falle der POI-Daten lassen sich aus den verbleibenden Vorlagen und der oben angeführten Beschreibung der Funktionalität die Schutzziele Integrität und Verfügbarkeit festlegen. Die Integrität der Daten soll gewährleistet werden, da sonst bei einer Änderung der POI-Daten durch unbefugte Personen oder Systeme deren Korrektheit nicht mehr gewährleistet werden kann. Da es sich um öffentliche Daten handelt, sollen diese für die Benutzer des KITCampusGuide ungehindert verfügbar sein. Die Öffentlichkeit der Daten und die Tatsache, dass keine personenbezogenen Daten ohne Einwilligung vom System bereitgestellt werden, schließt weitere Schutzziele, wie zum Beispiel die Vertraulichkeit der Daten, aus.

Im Gegensatz hierzu werden bei den vom KITCampusGuide erhobenen Positionsdaten der Benutzer die Vertraulichkeit und die Integrität der Daten als notwendig betrachtet. Die Integrität der Daten ist dabei relevant, um eine korrekte Funktionsweise des Software-Systems, zum Beispiel bei der Navigationsunterstützung, zu ermöglichen. Zudem soll so dem Missbrauch des Systems durch vorgetäuschte Positionsdaten vorgebeugt werden. Da die erhobenen Positionsdaten den aktuellen Aufenthaltsort von Benutzern wiedergeben, sollen diese Daten nur vom System zur Ausübung der

angeforderten Funktionalität eingesetzt werden. Somit wird eine vertrauliche Behandlung der Daten gefordert, um beispielsweise eine Verfolgung von bestimmten Benutzern durch unbefugte Personen zu verhindern. Das Schutzziel der Verfügbarkeit spielt bei den Positionsdaten eine untergeordnete Rolle, da bei deren Nichtverfügbarkeit der Funktionsumfang des KITCampusGuides in einem tolerablen Rahmen eingeschränkt ist. Abbildung 81 zeigt das Resultat aus der Schutzbedarfsanalyse und der Festlegung der Schutzziele für die schützenswerten Ressourcen im KITCampusGuide.

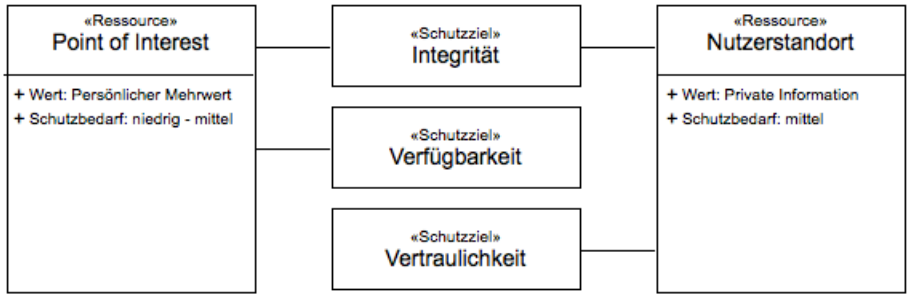

Abbildung 81: Schutzziele und Schutzbedarf der Ressourcen im KITCampusGuide

Bedrohungsanalyse

Durch die erneute Filterung der Vorlagen kann die Untersuchung von Bedrohungen und Angriffen zunächst eingeschränkt und somit effizienter durchgeführt werden. Somit werden lediglich die in den verbleibenden Vorlagen referenzierten Angriffe auf das KITCampusGuide-Szenario angewendet. Hierdurch wird das Problem reduziert, eine vollständige Erfassung aller möglichen Bedrohungen für das Software-System durchzuführen. Dies hat zum Nachteil, dass hierfür im Allgemeinen kein Abschlusskriterium zur Verfügung steht und dies zudem zu einer sehr breiten Palette an Schutzmaßnahmen führt, welche gegebenenfalls für die Problemstellung überdimensioniert sind. Da die Untersuchung von Angriffen jedoch ein kritischer Schritt in der Anforderungsanalyse ist, können nach dieser eingeschränkten Analyse nach Bedarf weitere im Katalog abgelegte Vorlagen hinsichtlich der für den KITCampusGuide zutreffenden Angriffe überprüft werden. Dieser Schritt ist vor allem dann relevant, wenn die Aktivitäten zur Anforderungsanalyse in einer anderen Reihenfolge als in Kapitel 4.3 beschrieben ausgeführt werden.

Ausgehend von den in Kapitel 4 vorgestellten Bedrohungskategorien lassen sich für die POI-Daten des KITCampusGuide die Verfälschung bzw. Sabotage der POI-Daten und die Beeinträchtigung der Dienstgüte der Bereitstellung der POI-Daten als mögliche Bedrohungen identifizieren. Eine Sabotage oder eine Verfälschung von POI-Daten zielt dabei darauf ab, zum einen die öffentliche Reputation des KITs zu schädigen oder zum anderen die Benutzer des KITCampusGuides in ihrer Ausführung von Aktivitäten zu stören. In beiden Fällen wird dabei das Schutzziel der Integrität und der Verfügbarkeit der POI-Daten verletzt. Ein konkreter Angriff zur Sabotage könnte hierbei die Verfälschung der POI-Daten an der Datenquelle vorsehen, in dem ein unbefugter Zugriff erlangt und die Manipulation der POI-Daten durchgeführt wird. Ebenso kann eine Verfälschung von POI-Daten für einen bestimmten Benutzer durch das Abfangen und die Manipulation von übertragenden Daten durchgeführt werden.

Da der KITCampusGuide von verschiedenen Benutzern verwendet wird, sollte auch ein menschlicher Irrtum bei der Bedienung des Software-Systems mit berücksichtigt werden. Dies ist insbesondere der Fall, wenn POI-Verwalter neue Daten eingeben bzw. bestehende Daten aktualisieren. Hierbei können Fehleingaben zu verwirrenden Resultaten für die Benutzer des KITCampusGuide führen. Ebenso sollte eine Fehlbedienung durch Benutzer nicht das gesamte System blockieren, da hierdurch die Verfügbarkeit eingeschränkt wird.

Für die Positionsdaten der Benutzer sind vor allem Angriffe der Bedrohungskategorie Diebstahl relevant. Eine unbefugte Erhebung sowie die Entwendung bzw. unrechtmäßige Weitergabe dieser Daten ohne Zustimmung der Nutzer stellt eine Verletzung der Schutzziele der Vertraulichkeit sowie der Integrität dar und ist zudem auch eine gesetzlich illegale Aktivität. Dabei birgt vor allem die Möglichkeit zur Überwachung der Bewegungen von einzelnen Personen die Gefahr der Verletzung der Privatsphäre. Ein konkreter Angriff ist dabei unter anderem das Abhören der Übertragung der Positionsdaten des Benutzers an das KITCampusGuide-System.

Eine weitere Möglichkeit eines Angriffs ist die Vortäuschung von falschen Positionsdaten gegenüber dem KITCampusGuide. Insbesondere in Kombination mit dem Diebstahl von Zugangsinformationen von anderen Benutzern kann dies gegebenenfalls zu einer Schädigung des persönlichen Ansehens von Benutzern führen. In diesem Fall ist die Integrität der Positionsdaten eines Benutzers verletzt.

Formulierung von Sicherheitsanforderungen

Aus den analysierten Werten für die Ressourcen lassen sich aus dem Vorlagenkatalog die zugehörigen Sicherheitsanforderungen für die POI-Daten sowie die Positionsdaten ableiten. Diese sind in Tabelle 6 zusammengefasst. Die Sicherheitsanforderungen stellen konkrete, jedoch technologieunabhängige Voraussetzungen für die Nutzung der Funktionalität des KITCampusGuides dar, so dass die spezifizierten Schutzziele eingehalten sowie die analysierten Angriffe abgewendet bzw. deren Auswirkungen vermindert werden können. Die Sicherheitsanforderungen werden in Abbildung 82 in Form von Sicherheitsanwendungsfällen dargestellt und in Relation zu den zugehörigen Anwendungs- sowie Missbrauchsfällen gesetzt. Somit wird das Anwendungsfalldiagramm für den KITCampusGuide komplettiert und die Anforderungsanalyse abgeschlossen.

Tabelle 6: Auszug aus den Sicherheitsanforderungen für den KITCampusGuide

Anwendungsfall	Sicherheitsanforderung
Verwaltung von Point of Interest	Ein Point of Interest soll zur Erhaltung der Integrität vor einer Sabotage durch einen unbefugten Einzeltäter geschützt werden.
Anzeige von Point of Interest	Ein Point of Interest soll bei der Anzeige zur Erhaltung der Integrität vor einer Manipulation durch einen Einzeltäter geschützt werden.
Navigation zu Point of Interest	Der Standort eines Benutzers soll zur Erhaltung der Integrität vor der Manipulation durch einen Einzeltäter geschützt werden.
Abfrage des Standortes	Der Standort eines Benutzers soll zur Erhaltung der Vertraulichkeit vor einem Abhören durch einen Einzeltäter geschützt werden.

Zur Wahrung der Schutzbedürfnisse der POI-Daten vor den Angriffen der Sabotage soll die Ausführung der Anwendungsfälle der Eintragung von neuen POI-Daten sowie die Aktualisierung von bestehenden POI-Daten ausschließlich durch befugte POI-Verwalter durchgeführt werden können. Als Folge dieser Einschränkung sollen unbefugte Personen daran gehindert werden, die Datenquellen der POIs so zu verändern, dass den Benutzern fehlerhafte Daten dargestellt werden. Ebenso sollen die POI-Daten bei der Übertragung an den Benutzer vor Veränderung geschützt werden, um einer Manipulation der Daten während der Übertragung vorzubeugen. Des Weiteren soll hierdurch die Verfügbarkeit der POI-Daten durch das KITCampusGuide-System gewährleistet werden.

Geeignete Schutzmaßnahmen sind ebenfalls für die Erhebung und die temporäre Speicherung der GPS-Daten auf Seiten des KITCampusGuide-Systems vorzusehen. Durch diese Einschränkungen an der Funktionalität der Navigationsunterstützung durch das System wird die Einsicht oder die Manipulation der Daten durch unbefugte Personen unterbunden. Für die Übertragung der

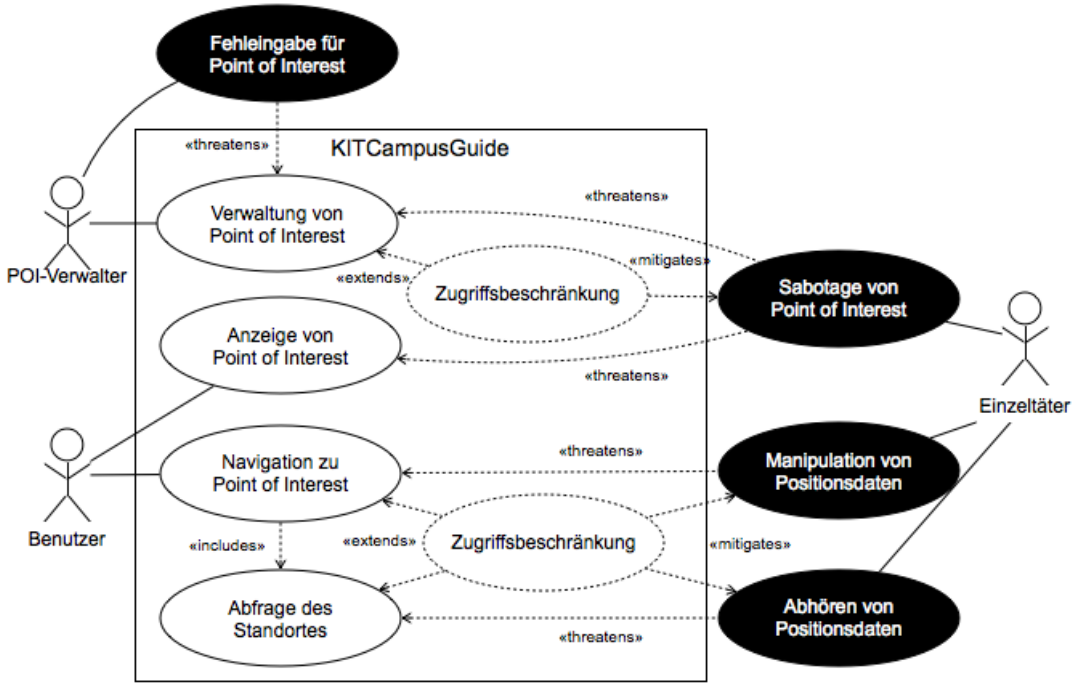

Abbildung 82: Ergebnisse der Sicherheitsanforderungsanalyse des KITCampusGuides

Positionsdaten von Benutzern sollen ebenfalls Maßnahmen getroffen werden, so dass diese Daten vor dem Einsehen durch Dritte geschützt sind. Zur Vorbeugung eines Angriffs durch die Vortäuschung von Positionsdaten von Benutzern durch einen Angreifer, wird ebenfalls eine Einschränkung der Anwendungsfälle der Navigationsunterstützung vorgenommen. Hierzu soll diese Funktionalität lediglich Benutzern zur Verfügung gestellt werden, welche dem System bekannt sind.

Resümee

Durch den Einsatz der Vorlagen konnten in der Analysephase des KITCampusGuide die Sicherheitsanforderungen effizient abgeleitet werden. Hierzu boten die Vorlagen einen Rahmen, innerhalb welchem eine strukturierte Analyse mittels eines Abgleiches zwischen den abstrakt in den Vorlagen spezifizierten Werten für Schutzbedürfnisse, Schutzziele, Angriffe sowie Anforderungen und der im KITCampusGuide konkret vorhandenen Sicherheitssituation durchgeführt werden konnte. Die Ergebnisse der Sicherheitsanforderungsanalyse fließen in die nachfolgende Entwurfsphase ein und bieten hierbei den Vorteil, bereits erste abstrakte Maßnahmen für die spezifizierten Anforderungen aufzuzeigen.

7.1.2 Entwurf der Sicherheitsarchitektur und -maßnahmen

Nach der Erhebung und Spezifikation der Sicherheitsanforderungen in der Analysephase werden in der sich anschließenden Entwurfsphase Sicherheitsmaßnahmen modelliert, welche diese Anforderungen umsetzen. Gemäß dem sicherheitsbasierten Entwicklungsvorgehen werden hierzu die Sicherheitsanforderungen auf abstrakte Sicherheitstaktiken abgebildet, welche anschließend iterativ verfeinert werden, um eine Sicherheitsarchitektur und die technologische Umsetzung der Sicherheits-funktionalität zu realisieren. Bei der Ableitung der Sicherheitsmaßnahmen wird auf die in Kapitel 6 beschriebenen Sicherheitsmuster für Authentifizierung, Autorisierung und Zugriffskontrolle zurückgegriffen. Somit werden anderweitige Sicherheitsmaßnahmen an dieser Stellen nicht untersucht.

Auswahl von Taktiken

Für die POI-Daten wurden in der Analysephase Vorlagen ausgewählt, welche zum einen die Integrität der Daten bei Sabotageangriffen und menschlichen Irrtum schützen und zum anderen die Verfügbarkeit der Daten erfordern. Im Fall der Positionsdaten von Benutzern des KITCampusGuides sollen sowohl die Schutzziele der Integrität als auch der Vertraulichkeit bei Eintritt von Lauschangriffen, unbefugten Zugriffen sowie der Manipulation der Daten gewährleistet werden. Um diese Sicherheitsanforderungen umzusetzen, werden die mit den Vorlagen verknüpften Sicherheitstaktiken analysiert und gemäß dem Schutzbedarf der Ressourcen ausgewählt.

Wie in Kapitel 6 beschrieben, existieren drei unterschiedliche Taktiken zur Umsetzung von Sicherheitsanforderungen. Für die Umsetzung der Sicherheitsanforderungen für POI-Daten sind sowohl eine Taktik der Wiederherstellung als auch der Vorbeugung geeignet. Eine Möglichkeit für den Einsatz einer wiederherstellenden Maßnahme ist dabei das Zurücksetzen von POI-Daten nach einem Sabotageangriff auf die zuvor gültige Version. Ebenso können die Schadensauswirkungen bei einer durch menschlichen Irrtum vorgenommenen Fehleingabe durch eine effiziente Wiederherstellungsmaßnahme vermindert werden. Eine Wiederherstellung bedarf dabei der zusätzlichen Sicherheitsmaßnahme einer Sicherung der POI-Daten.

Als vorbeugende Maßnahme lässt sich eine Zugriffskontrollmaßnahme implementieren, wodurch zum einen die unbefugten Benutzer des KITCampusGuides vor der Eintragung und Veränderung von POI-Daten gehindert werden und zum anderen Angreifern der Zugriff auf die POI-Daten erschwert wird. Eine Zugriffskontrolle setzt gemäß der horizontalen Sicherheitsmustersprache für Zugriffskontrolle in Kapitel 6 die vorhergehende Authentifizierung von Benutzern sowie deren Autorisierung voraus. Diese zusätzlichen Maßnahmen müssen daher bei der weiteren Verfeinerung der Zugriffskontrolle berücksichtigt werden.

Im Falle der Positionsdaten sollten vorwiegend vorbeugende und detektierende Maßnahmen berücksichtigt werden, da wiederherstellende Maßnahmen nicht den gewünschten Schutzeffekt gegenüber den analysierten Angriffen bieten. Dem Abhören von Positionsdaten sollte dagegen aktiv vorgebeugt und mögliche Angriffe rechtzeitig erkannt werden, so dass entsprechende Gegenmaßnahmen eingeleitet werden können. Einem aktiven Mitlauschen der Positionsdaten sollte beispielsweise durch die Übermittlung der Daten durch Einsatz des Sicherheitsmusters „Sicherer Kanal“ [SF+05:434] vorgebeugt werden, welcher die gesendeten Daten verschlüsselt. Ebenso sollte auch bei den Positionsdaten eine Zugriffskontrolle eingeführt werden, so dass diese nicht von unbefugten Personen eingesehen oder verändert werden können.

Im weiteren Verlauf der Modellierung der Sicherheitsfunktionalität des KITCampusGuides wird der Fokus auf die Umsetzung der Zugriffskontrollmaßnahmen sowie der zugehörigen Authentifizierung und Autorisierung gelegt. Hierfür werden die in Kapitel 6 vorgestellten Sicherheitsmustersprachen für diese Maßnahmen eingesetzt. Die Verfeinerung der weiteren Maßnahmen anderer Taktiken erfolgt dabei analog. Ziel der Verfeinerung ist die Demonstration des Einsatzes der Sicherheitsmuster für die strukturierte und effiziente Modellierung von Sicherheitsmaßnahmen.

Auswahl von Architekturkomponenten

Das in Kapitel 6 vorgestellte Variabilitätsmodell für Zugriffskontrolle behandelt die Architekturkomponenten, mit deren Hilfe die durch eine Autorisation festgelegten Zugriffsregeln im Software-System überprüft und umgesetzt werden. Hierdurch wird eine Integration von fachlicher Architektur und Sicherheitsarchitektur benötigt, deren Komponenten jedoch lose gekoppelt sein sollen. In Abbildung 83 ist die fachliche Architektur des KITCampusGuide in Kombination mit den ausgewählten Zugriffskontrollkomponenten und -diensten dargestellt.

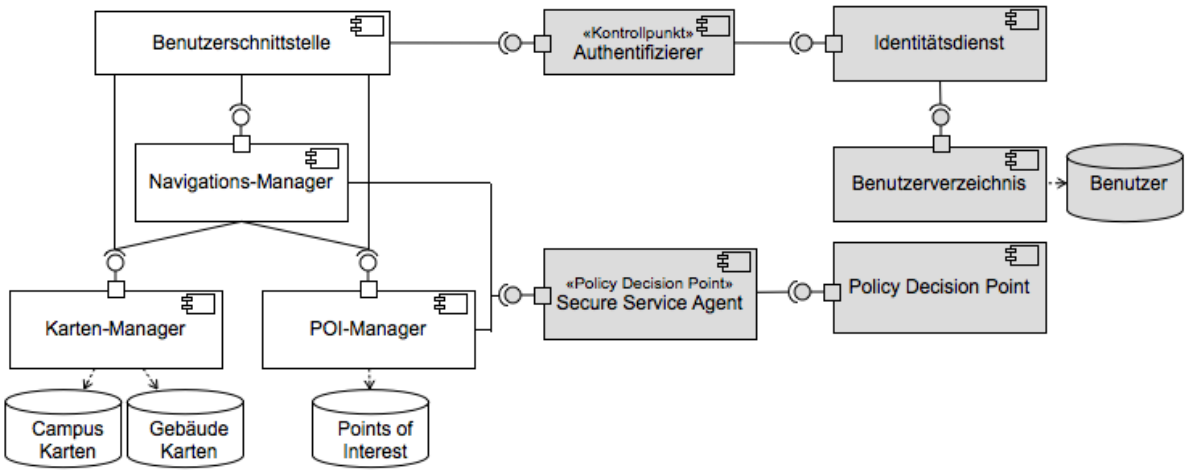

Abbildung 83: Fachfunktionalität und Sicherheitskomponenten des KITCampusGuides

Der KITCampusGuide wird in eine klassische 3-Schichten-Architektur mit einer graphischen Benutzerschnittstelle (GUI), einer Funktions- und einer Datenhaltungsschicht aufgeteilt. In der Funktionsschicht wird die Suche nach POIs sowie deren Verwaltung durch die Komponente POI-Manager übernommen. Die Funktionalität zur Bereitstellung von Kartendaten wird durch die Karten-Manager-Komponente angeboten. Der Navigationsmanager nutzt die Funktionalität des Karten-Managers und des POI-Managers zur Berechnung und Darstellung einer Route zwischen zwei oder mehr POIs. Die Komponenten speichern zusätzliche relevante Daten ab, welche als schützenswerte Ressourcen identifiziert wurden. Lediglich der Navigations-Manager berechnet die Routeninformation aus den POI-und Kartendaten, so dass diese Komponente keinen eigenen persistenten Datensatz benötigt.

Zunächst wird durch das Variabilitätsmodell die Zentralisierung der Zugriffskontrollfunktionalität durch Einsatz des Kontrollpunkt-Musters verlangt, da hierdurch eine Trennung von Zuständigkeiten und eine Modularisierung der sicherheitsrelevanten Funktionalität erreicht wird. Ebenfalls wird hierdurch im Allgemeinen der flexible Austausch von Sicherheitsüberprüfungen ermöglicht. Das Kontrollpunkt-Muster wird durch das Referenzmonitor-Muster spezialisiert, welches die Überprüfung von allen Anfragen an eine geschützte Ressource auf vorhandene Berechtigungen erfordert. Im Falle der POI-Daten werden hiermit Anfragen an die POI-Manager-Komponente sowie die Navigations-Manager-Komponente des KITCampusGuide kontrolliert.

Als Konkretisierung des Referenzmonitors wird das Policy-Enforcement-Point-Muster (PEP) und das Policy-Decision-Point-Muster (PDP) eingesetzt, um das Abfangen einer Anfrage sowie die Umsetzung einer Zugriffskontrollentscheidung von der Berechnung der Entscheidung zu entkoppeln. Der PEP schützt dabei die Dienste, in dem die Anfragen abgefangen werden und an den PDP zur Auswertung weitergeleitet werden. Insbesondere wird dabei das Secure-Service-Agent-Entwurfsmuster eingesetzt, da zum einen die Komponenten gleichartige Ressourcen darstellen und somit nur eine Form von PEP benötigt wird. Zum anderen wird die Umsetzung der Zugriffskontrollentscheidung an den Komponenten direkt vorgenommen, so dass eine Portierung der Komponenten in eine andere Ausführungsumgebung ermöglicht wird.

Für die Umsetzung der Authentifizierungsmaßnahme, welche eine Voraussetzung für die Zugriffskontrolle ist, sind weitere spezialisierte Maßnahmen notwendig, welche den internen Aufbau der Authentifizierung klären. Aus architektureller Sicht sind dabei zunächst die Komponenten für die Überprüfung der Identität relevant. Der in Kapitel 6 vorgestellte Maßnahmenbaum zur Umsetzung von Authentifizierungsmaßnahmen setzt hierbei zunächst einen zentralen Zugangspunkt voraus, mittels welchem der Zugang zum Software-System und den verarbeitenden Ressourcen, wie etwa die POI-

Daten und die Positionsdaten, geregelt wird. Eine Spezialisierung des Zugangspunktes stellt das Authentifizierer-Muster dar, welches analog zum PEP die Zugangsüberprüfung durchsetzt. Als zusätzliche Komponenten werden ein Benutzerverzeichnis und ein Identitätsdienst benötigt. Im Benutzerverzeichnis werden die Authentifizierungsinformationen von Benutzern sicher abgelegt. Der Identitätsdienst bietet analog zum PDP eine zentrale Funktionalität zur Überprüfung von Authentifizierungsnachweisen und wird vom Authentifizierer genutzt.

Auswahl von Sicherheitsrichtlinien

Die Umsetzung der Zugriffskontrolle erfordert neben der Authentifizierung auch die Umsetzung einer Autorisierungsmaßnahme, welche die notwendigen Regeln bereitstellt, auf deren Basis die Zugriffskontrolle operieren kann. Ebenso benötigt das Authentifizierungsmuster neben den architekturellen Komponenten weitere Instruktionen, auf welche Art und Weise die Authentifizierung erfolgen soll. Im Folgenden werden diese Sicherheitsrichtlinien mittels der zugehörigen Musterbäume für den KITCampusGuide definiert.

Für die Überprüfung der Zugriffsrechte müssen Benutzer des KITCampusGuides, gemäß der Spezialisierung des Authentifizierungsmusters, Nachweise erbringen, mittels welcher die Identität der Benutzer eindeutig geklärt ist. Wie in Kapitel 6 dargestellt, sind mehrere Arten von Nachweise möglich, welche auch in Kombination eingesetzt werden können. Im vorliegenden Szenario gibt die existierende Sicherheitsinfrastruktur bereits eine Methode des Authentifizierungsnachweises vor. Da alle Studierenden sowie Mitarbeiter des KIT ein allgemeines Benutzerkonto vom Rechenzentrum des KIT, dem Steinbuch Centre for Computing (SCC), erhalten, soll dieses auch für den KITCampusGuide eingesetzt werden.

Hierbei handelt es sich um einen wissensbasierten Authentifizierungsnachweis, welcher durch eine Kombination aus Benutzername und Passwort umgesetzt wird. Somit wird die Authentifizierungsstärke der zu erbringenden Nachweise auf eine Ein-Faktor-Authentifizierung beschränkt. Durch das SCC sind zudem Richtlinien vorgegeben, welche die Stärke der Passwörter festlegen, wodurch leicht zu erratende oder berechnende Zeichenkombinationen unterbunden werden sollen. In Kombination mit den architekturellen Komponenten konnte das Authentifizierungsmuster somit erfolgreich eingesetzt werden.

Die Autorisierung regelt die Zugriffsberechtigungen für die Benutzer des KITCampusGuides. Wie in der Analyse der Sicherheitsanforderungen besprochen, sollen lediglich spezifische KIT-Mitarbeiter, welche mit der Verwaltung von POI-Daten beauftragt sind, die Möglichkeit haben, diese zu bearbeiten. Studenten und Gäste dagegen sollen lediglich eine Berechtigung zum Lesen von POI-Daten erhalten. In diesem Kontext spiegelt die Vergabe von Rechten an Rollen die Berechtigungsanforderungen des KITCampusGuides am geeignetsten wider. Für den Zugriff auf den Navigations-Manager, welcher unter anderem die aktuelle Position eines Benutzers abfragt, soll lediglich der jeweilige Benutzer berechtigt sein, seine eigenen Positionsdaten zu lesen. Ein schreibender Zugriff ist nicht zugelassen, um einer Manipulation von Positionsdaten vorzubeugen.

Somit wird der Ansatz einer rollenbasierten Zugriffskontrolle gewählt, in welcher die allgemeinen Rollen „ROLLE_BENUTZER“ und „ROLLE_POI-VERWALTER“ definiert werden. Einem Benutzer ohne Modifikationsrechte von POI-Daten wird dabei die Rolle „ROLLE_BENUTZER“ zugewiesen, welche lediglich Leserechte auf die POI-Daten besitzt. Dagegen erhalten zur Verwaltung der POIs berechtigte KIT-Mitarbeiter die Rolle „ROLLE-POI-VERWALTER“ zugewiesen, welche die entsprechenden Berechtigungen kapselt. Ebenso wird „ROLLE_BENUTZER“ ein Leserecht auf dem Navigations-Manager zugewiesen. Diese Zuweisungen sind in Abbildung 84 dargestellt.

Zur besseren Verwaltung der Rechtezuweisungen wird eine einfache zweistufige Rollenhierarchie zwischen „ROLLE_BENUTZER“ und „ROLLE_POI-VERWALTER“ eingesetzt. Hierbei erbt die

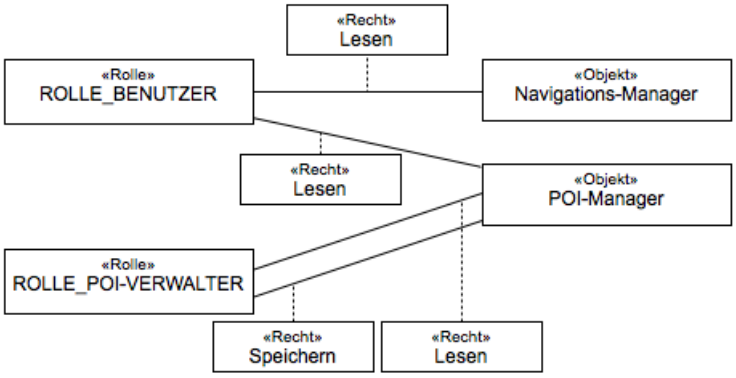

Abbildung 84: Rollen und Rechtezuweisung für den KITCampusGuide

Verwalterrolle das Leserecht der Benutzerrolle und erhält zusätzlich die Schreibrechte für POIs. Obwohl durch die geringe Anzahl der Rollen eine RBAC-0-Modell ohne Hierarchie ebenfalls eingesetzt werden kann, ist die Realisierung einer Rollenhierarchie später zur Verwaltung der XACML-Richtlinien sinnvoll.

Feinentwurf

Als Implementierungsplattform des KITCampusGuides wurde das Java Enterprise Edition (JEE) Rahmenwerk ausgewählt. Hierdurch spielt insbesondere die Servlet-Technologie für die serverseitig angebotene Funktionalität des KITCampusGuides eine Rolle. Die Benutzerschnittstelle wurde dagegen mit der auf der Servlet-Technologie basierenden JavaServer-Pages-Technologie umgesetzt (JSP). Das JEE-Rahmenwerk bestimmt die Ausführungsumgebung der Anwendung, indem ein Web-Container benötigt wird, um die serverseitigen Komponenten auszuführen. Wie in Abbildung 85 dargestellt, wird dagegen die Benutzerschnittstelle in einem Web-Browser beim Client ausgeführt.

Um eine Trennung der Zuständigkeiten zwischen den fachlichen Komponenten und der Sicherheitsfunktionalität zu erreichen, werden des Weiteren die Sicherheitskomponenten auf getrennten Ausführungsumgebungen bereitgestellt. Zum einen dient dies der Stabilität sowie dem Schutz des gesamten Systems und ermöglicht die Wiederverwendung der Sicherheitsfunktionalität. Zum anderen ist diese Trennung bereits durch die verwendete Sicherheitsinfrastruktur am KIT vorgegeben.

Zur Interaktion zwischen fachlicher Funktionalität und den externen Sicherheitsdiensten, werden leichtgewichtige Komponenten in die fachliche Ausführungsumgebung integriert. So wird beispielsweise der Secure Service Agent als Realisierung eines Policy Enforcement Points in den JEE-Web-Container bereitgestellt, so dass die fachlichen Komponenten die Zugriffskontrollanfrage an den SSA auslagern können. Der Secure Service Agent kapselt dabei die Logik zur Kommunikation mit dem zugehörigen Policy Decision Point und implementiert die Funktionalität zur Umsetzung der Zugriffskontrollentscheidung. Die Zugriffskontrollrichtlinien für den KITCampusGuide werden vom Policy Decision Point überprüft. Am KIT wird kein Sicherheitsprodukt eingesetzt, mittels welchem die feingranulare Autorisierung des KITCampusGuides überprüft werden kann. Daher wird diese Sicherheitsfunktionalität mittels eines Sicherheitsproduktes bzw. -rahmenwerks eigenständig realisiert.

Auf der Benutzerschnittstelle wird eine separate Komponente zur Authentifizierung von Subjekten eingesetzt, welche die Anfragen von Benutzern abfängt und zur Prüfung an den Identitätsdienst weiterleitet. Der hierzu nötige Identitätsdienst wird dabei separat von dem Policy Decision Point betrieben und vom SCC angeboten, welcher auch das Benutzerverzeichnis betreibt.

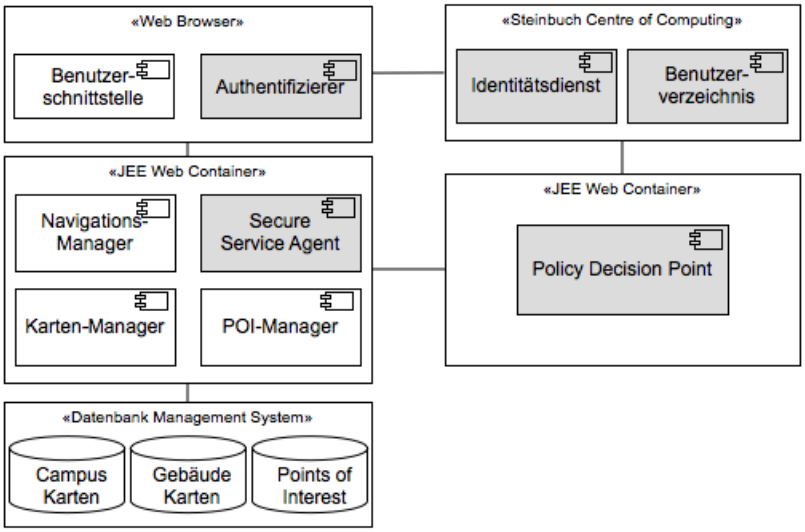

Abbildung 85: Feinentwurf des KITCampusGuides

Technologieabbildung

Zur Umsetzung der Sicherheitskomponenten wird eine Sicherheitsplattform eingesetzt, welche ausgehend von den vom KIT bereitgestellten Authentifizierungsdiensten das Sicherheitsrahmenwerk Spring Security nutzt, um feingranulare Zugriffskontrolle in dem KITCampusGuide umzusetzen.

Die Authentifizierung der Benutzer wird für den KITCampusGuide durch den vom SCC bereitgestellten Identitätsdienst durchgeführt, welcher auf der Shibboleth-Plattform [SHIBBOLETH], eine Erweiterung des SAML-Standards zur Authentifizierung (Security Assertion Markup Language [OASIS-SAML]), basiert. Durch die Shibboleth-Realisierung der Authentifizierung wird zudem eine Single-Sign-On-Lösung bereitgestellt, welche die mehrmalige Authentifizierung gegenüber verschiedenen Software-Systemen am KIT überflüssig macht. Hierzu werden entsprechende temporäre Nachweise bei einer erfolgreichen Authentifizierung den Benutzern ausgestellt. Der Identitätsdienst stützt sich dabei auf das am KIT eingesetzte proprietäre Benutzerverzeichnis Microsoft Active Directory [MS-AD], in welchem die Daten bezüglich der KIT-Studierenden, Mitarbeiter und Partner abgelegt sind. Dieses implementiert dabei eine proprietäre Erweiterung des LDAP-Standards (Lightweight Directory Access Protocol [IETF-LDAP]).

Für die Integration der Authentifizierungsfunktionalität in die Benutzerschnittstelle wurde das Spring-Security-Rahmenwerk eingesetzt, welches bereits vorgefertigte GUI-Komponenten zur Umsetzung des Authentifizierer-Musters in einem Web-basierten Software-System bereitstellt. Im Falle des KITCampusGuides handelt es sich zum einen um spezielle JavaServer-Pages, welche ein Anmelde-formular bereitstellen und die Kommunikation mit dem Identitätsdienst durchführen. Zum anderen werden Filterklassen zu Verfügung gestellt, welche die Anfrage an geschützte Web-Ressourcen abfangen und an die Anmeldeseite weiterleiten. Somit stellt diese Kombination die Realisierung des Authentifizierer-Musters dar.

Ebenso wurde die Funktionalität des Secure Service Agents zur Realisierung eines Policy Enforcement Points mittels der vom Spring-Security-Rahmenwerk zur Verfügung gestellten Mittel umgesetzt. Hierbei handelt es sich um die Umsetzung des Musters Steuerungsumkehr (engl. Inversion of Control [GH+94]), in welchem die zu schützenden Methoden einer Servlet-Klasse annotiert und dadurch zur Laufzeit vor der Ausführung der Methoden eine Zugriffskontrollüberprüfung angestoßen wird. Hierdurch wird eine Trennung zwischen der fachlichen und sicherheitsbezogenen Funktionalität auf Quellcodeebene ermöglicht.

Die Implementierung der Policy-Decision-Point-Funktionalität wird ebenfalls nicht durch die am KIT zur Verfügung stehenden Sicherheitsprodukte bereitgestellt. Daher wird hierfür eine Lösung gewählt, welche auf den XACML-Standard zur Beschreibung von Zugriffskontrollrichtlinien aufbaut. Zur Auswertung der standardisierten Richtlinien wird zunächst das Sicherheitsprodukt Axiomatics Policy Server [AXIOMATICS] eingesetzt. Das Spring-Security-Rahmenwerk wird anschließend erweitert, um die Zugriffskontrollanfrage an den Policy Server über die bereitgestellten Schnittstellen weiterzuleiten.

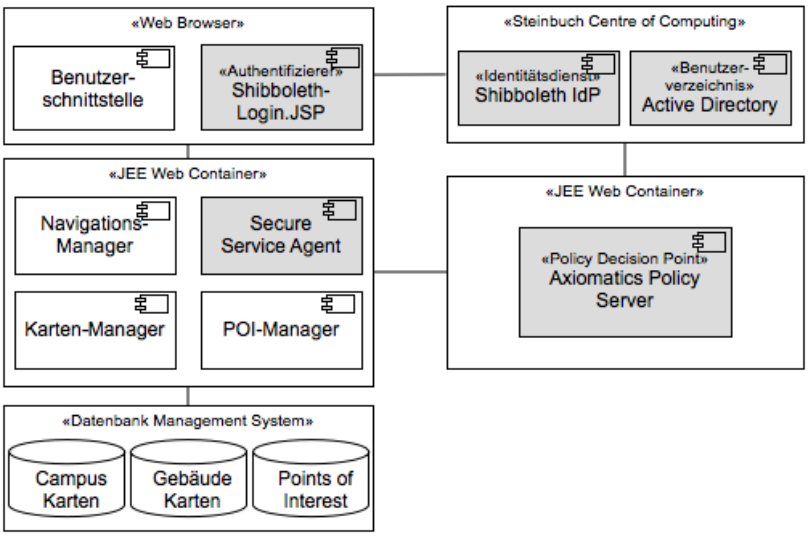

Abbildung 86: Technologien zur Umsetzung des KITCampusGuides

Für die Umsetzung der zuvor definierten Zugriffskontrollrichtlinien wird das RBAC-Profil von XACML angewendet, welches Idiome zur Definition von Richtlinien gemäß des RBAC-0-Modells vorgibt. Hierzu wird je ein Berechtigungssatz zur Spezifikation der erlaubten Aktionen für den POI-Manager sowie den Navigations-Manager definiert. Anschließend werden Rollenberechtigungssätze definiert, welche die Rollen mit den zugehörigen Berechtigungssätzen in Verbindung bringen.

Resümee

Durch die Anwendung der Sicherheitsmustersprachen wurde ausgehend von den abstrakten Sicherheitsmaßnahmen eine kontinuierliche Verfeinerung der Sicherheitsfunktionalität des KIT-CampusGuide-Systems vorgenommen. Hierdurch wurde die strukturierte Entwicklung der Sicherheitsmaßnahmen unterstützt. Bei der Ableitung der Sicherheitsmaßnahmen wurde durch die Sicherheitsmustersprachen eine Hilfestellung bei der Auswahl von verschiedenen Lösungen zur Realisierung der Sicherheitsanforderungen des Systems gegeben. Zudem wurden bei der Verfeinerung die verschiedenen Sicherheitsdienste der KIT-Sicherheitsinfrastruktur berücksichtigt, so dass das resultierende Software-System in die Infrastruktur integriert werden konnte.

7.1.3 Bewertung und Ausblick

Durch Einsatz des in Kapitel 4 beschriebenen sicherheitsbezogenen Entwicklungsvorgehens konnte in diesem Tragfähigkeitsnachweis die systematische und strukturierte Entwicklung der sicherheitsbezogenen Funktionalität für Authentifizierung, Autorisierung und Zugriffskontrolle für das KIT-CampusGuide-System demonstriert werden. Die Anwendung der in Kapitel 4 und Kapitel 5 beschriebenen aufbereiteten sicherheitsbezogenen Entwicklungsartefakte gestattete eine durchgängige

Betrachtung der Sicherheitsfunktionalität entlang der Anforderungsanalyse und der Entwurfsphase des Entwicklungsprozesses.

Durch den Einsatz der Sicherheitsanforderungsvorlagen wurde eine zielgerichtete Analyse der Sicherheitsanforderungen durchgeführt. Hierzu ermöglichen die Vorlagen die Strukturierung der Analyseaktivitäten und stellen zusätzlich wiederverwendbare Entwicklungsartefakte bereit. Durch die Verknüpfung von wesentlichen Artefakten in den Vorlagen, wie zum Beispiel Schutzbedarf, Schutzziel, Bedrohung und Sicherheitsanforderung, wurde die Analyse von Sicherheitsanforderungen für den KITCampusGuide unterstützt. Zudem wird durch die Vorlagen ein Rahmen geschaffen, in welchem bestehende Ansätze zur sicherheitsbasierten Entwicklung integriert und zielgerichtet eingesetzt werden können. Im Falle der Analyse wurde dies durch den Einsatz von Missbrauchs- und Sicherheitsanwendungsfällen veranschaulicht.

Durch die mittels des Variabilitätsmodells spezifizierten Sicherheitsmustersprachen konnten begründete Entscheidungen über den Einsatz von Sicherheitsmaßnahmen gemäß der vorliegenden Sicherheitsprobleme für den KITCampusGuide getroffen werden. Dabei wurden Entscheidungen teilweise bereits durch die existierende Sicherheitsinfrastruktur des KIT vorgegeben, wodurch die Modellierung und Entwicklung der Sicherheitsmaßnahmen effizient durchgeführt werden konnte.

Die entwickelte Sicherheitsfunktionalität zeichnet sich durch bewährte Methoden aus. So wird der Kern der Sicherheitsdienste zentral durch den KIT-eigenen IT-Dienstleister bereitgestellt und gepflegt. Hierdurch wird die Wartung und die Evolution der Sicherheitsfunktionalität durch Sicherheitsexperten gefördert. Zusätzliche Sicherheitsfunktionalität, wie zum Beispiel die Überprüfung von Autorisierungsrichtlinien durch einen Policy Decision Point, wurden im Falle des Axiomatics Policy Servers durch neue Sicherheitsprodukte realisiert.

Die Absicherung der Ressourcen und fachlichen Komponenten wird durch leichtgewichtige Sicherheitskomponenten durchgeführt, welche lose gekoppelt sind. Dies ermöglicht zum einen eine Trennung der Zuständigkeiten zwischen Sicherheitsexperten und fachlichen Software-Entwicklern. Zum anderen wird auch der Austausch von Sicherheitsfunktionalität durch diese Modularisierung unterstützt. Die durchgängige Verfeinerung der Sicherheitsmaßnahmen durch die Mustersprachen ermöglicht es somit, die modellierte Sicherheitsfunktionalität auf konkrete Technologien abzubilden, welche durch die Sicherheitsinfrastruktur vorgegeben sind. Dies erleichtert die Implementierung der Sicherheitsfunktionalität, da die eingesetzten Technologien und Produkte durch langjährige Nutzungsdauer bereits erprobt und getestet sind.

Die Sicherheitsdienste konnten dabei durch den Einsatz von Standards zur Beschreibung der Schnittstelle bzw. zur Beschreibung der sicherheitsbezogenen Daten realisiert werden. Hierdurch wird die Unabhängigkeit von einem konkreten Sicherheitsprodukt gewährleistet. Des Weiteren wird eine Interoperabilität zwischen verschiedenen Systemen ermöglicht. Beispielsweise wird im Falle des Shibboleth-Identitätsdienstes ein Single-Sign-On-Verfahren umgesetzt, wodurch eine mehrmalige Authentifzierung umgangen wird.

Die betrachtete Funktionalität des KITCampusGuides stellt lediglich einen überschaubaren Ausschnitt des gesamten Funktionsumfangs dar, um den Einsatz der vorgestellten Beiträge dieser Arbeit zu demonstrieren. Zukünftig wird das KITCampusGuide-System um zusätzliche Funktionen erweitert, um den am KIT beteiligten Personen weitere Dienste zur Unterstützung bei alltäglichen Aktivitäten auf dem Campusgelände anzubieten. Hierzu gehören unter anderem eine Plattform zur Erstellung und Verwaltung von Lerngruppen und eine Diskussionsplattform. Diese Funktionen werden dabei durch die Integration von bestehenden Software-Systemen am KIT ermöglicht.

Durch diese Erweiterungen wird eine Überarbeitung der Sicherheitsfunktionalität unumgänglich. Hierbei bietet das bereits aufbereitete Sicherheitswissen einen Leitfaden zur Umsetzung von

zusätzlichen Sicherheitsmaßnahmen, sofern sich die bestehende Sicherheitsfunktionalität als nicht ausreichend herausstellt. Es ist jedoch abzusehen, dass die wesentlichen Kerndienste der Sicherheitsarchitektur erhalten bleiben und genutzt werden können. Dagegen sind Anpassungen bei den Autorisierungsrichtlinien und den leichtgewichtigen Sicherheitskomponenten zur Kopplung und Integration der Fachfunktionalität vorzunehmen. Durch die im Referenzmodell vorliegenden Konventionen zum Einsatz der Richtlinien wird hierbei ebenfalls eine Hilfestellung bei deren Formulierung bereitgestellt.

7.2 SmartMeetings – Sicherheit in verteilten, autonomen Systemen

Das Ziel des EU-Projektes OpenIoT ist es, eine quelloffene Middleware zur Umsetzung des Paradigmas des Internets-der-Dinge (IoT) in großflächig verteilten Software-Systemen zu entwickeln. Das IoT-Paradigma sieht dabei die Ausstattung von Entitäten der realen oder virtuellen Welt mit Rechen- oder Speicherleistung vor, so dass diese mittels Internettechnologien vernetzt und in bestehende sowie zukünftige Internet-basierte Dienste eingebunden werden können. Die physischen Gegenstände sind dabei durch Sensoren, Aktoren und eingebettete Systeme ausgestattet, durch welche jeder Gegenstand einzeln oder in einer autonom vernetzten Kooperation mit weiteren Gegenständen unterschiedliche Hilfsdienste anderen Gegenständen und Software-Systemen bereitstellen kann. Das Paradigma erweitert dabei den derzeitigen Ansatz des Cloud Computings zur Bereitstellung von Infrastrukturen, bei welchem Ausführungsplattformen (engl. Platform as a Service, PaaS) und Software-Systeme (engl. Software as a Service, SaaS) als Internet- und Web-basierte Dienste bereitgestellt werden. Die Ergänzungen umfassen die Bereitstellung von sensorbasierten Infrastruktur- und Anwendungsdiensten wie beispielsweise Beobachtung und Messung (engl. Sensing as a Service), Ortbestimmung (engl. Location as a Service) und Begleitung (engl. Traceability as a Service).

Die im Rahmen des OpenIoT-Projektes entwickelte Middleware soll dazu genutzt werden, um ein Software-System mit dem Namen „SmartMeetings“ zur Suche und Reservierung von freien Arbeitsplätzen für Studierende des KIT zu entwickeln. Die den Studierenden zur Verfügung stehenden Arbeitsplätze sind auf dem gesamten Campusgelände verteilt und stehen meist nicht in genügender Anzahl zur Verfügung. Um eine bessere Auslastung von freien Arbeitsplätzen zu gewährleisten und Studenten in ihrem Lernprozess zu unterstützen, soll es möglich sein, freie Arbeitsplätze durch eine Web-basierte Oberfläche zu suchen und zu reservieren. Hierbei soll das IoT-Paradigma eingesetzt und Arbeitsplätze mit Sensorik ausgestattet werden. Das Ziel dieses Ansatzes ist es, die Arbeitsplätze dezentral zu verwalten, indem diese autonom über ihre aktuelle Belegung und Ausstattung Auskunft geben können. Mittels des OpenIoT-Rahmenwerkes soll die Arbeitsplatzverwaltung als durch ein Sensornetzwerk realisierter Dienst zur Verfügung gestellt werden.

Die Arbeitsplatzsuche des SmartMeetings-System soll als zusätzliche Funktionalität in die bereits bestehende KITCampusGuide-Anwendung integriert werden. Hierdurch lässt sich die Funktionalität des KITCampusGuides zur Anzeige des Arbeitsplatzes auf der Campuskarte und Navigation zum Arbeitsplatz nutzen. Im Folgenden soll anhand dieses Szenarios veranschaulicht werden, inwieweit der in dieser Arbeit vorgestellte Ansatz auf die iterative Erweiterung einer Anwendung und auf die Sicherheitsfragestellungen eines IoT- und Cloud-basierten Ansatzes angewendet werden können. Dabei ist zu berücksichtigten, dass die Datenverarbeitung des OpenIoT-Rahmenwerks auf semantischen Technologien beruht, für welche noch keine allgemein anerkannten Sicherheitsmaßnahmen entwickelt wurden. Daher wird der Fokus der sicherheitsbasierten Entwicklung auf die Realisierung von Schutzmaßnahmen auf der Anwendungs- bzw. Dienstebene gelegt.

7.2.1 Analyse der Sicherheitsanfoderungen

Nachfolgend werden die Sicherheitsanforderungen des SmartMeetings-Systems unter Einsatz des in Kapitel 4 vorgestellten Entwicklungsvorgehens und der in Kapitel 5 entwickelten Sicherheitsanforderungsvorlagen analysiert. Hierbei wird im Gegensatz zur Entwicklung des KITCampusGuides ein agiles Vorgehen eingesetzt, um die Anforderungen zu spezifizieren.

Überblick über das SmartMeetings-System

Wie bereits dargestellt, soll es KIT-Studierenden mittels der SmartMeetings-Anwendung ermöglicht werden, einen Arbeitsplatz auf dem KIT-Campus zu finden und zu reservieren. Hierbei sollen sowohl Reservierungen von Arbeitsplätzen für einzelne Personen als auch Räume für Lerngruppen mit mehreren Personen unterstützt werden. Um eine entsprechende Reservierung vorzunehmen, soll zunächst eine Suche sowie eine Filterung von Arbeitsplätzen nach den Bedürfnissen der Studierenden durch das SmartMeetings-System unterstützt werden. Für Lerngruppen soll zudem die Möglichkeit bestehen, eine Benachrichtigung über die Reservierung an die Mitglieder der Gruppe zu versenden. Zur Bestätigung der Reservierung soll des Weiteren eine Interaktion zwischen dem Benutzer und dem Arbeitsplatz mittels einer Sensorabfrage durchgeführt werden. Wird diese Reservierungsbestätigung nicht durchgeführt, so wird die Reservierung storniert.

In dem in Abbildung 87 dargestellten UML-Anwendungsfalldiagramm wird eine Übersicht über die geforderte Funktionalität des SmartMeetings-Systems gezeigt. Die Suche nach und das Filtern von Arbeitsplätzen steht den Benutzern des Systems zu. Dabei kann jedoch zwischen Studierenden des KIT, welche in allen Räumen auf dem KIT-Gelände Arbeitsplätze reservieren dürfen, und Bibliotheksmitgliedern, welche nicht anderweitig am KIT beschäftigt sind und für welche nur Plätze und Räume in der KIT-Bibliothek reservierbar sind, unterschieden werden. Im Falle einer Reservierung für Lerngruppen sollen die Reservierungsdaten nur von Mitgliedern der Gruppe eingesehen werden. Für andere Benutzer soll der Raum lediglich als gebucht angezeigt werden, so dass weitere Details über die Reservierung nicht verfügbar sind.

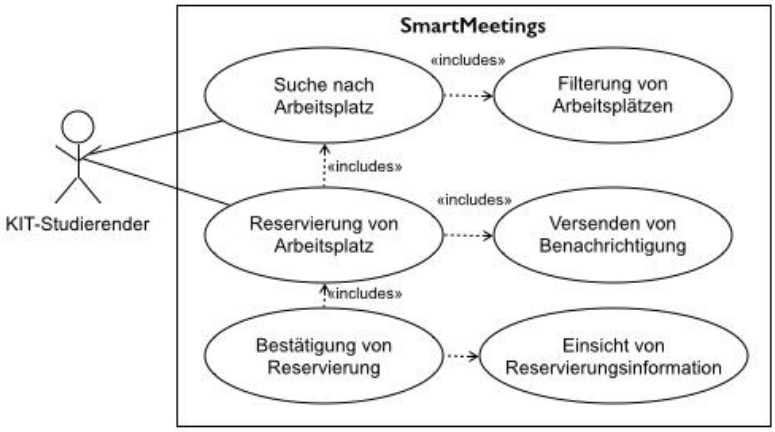

Abbildung 87: Anwendungsfälle für das SmartMeetings-System

Für die in der agilen Entwicklung des SmartMeetings-Systems eingesetzte Scrum-Methode resultieren die Anwendungsfälle in User Stories, welche den Inhalt des Product Backlog darstellen und die weitere Entwicklung des Systems bestimmen. Ein Auszug der User Stories ist in Tabelle 7 dargestellt. In einem Anwendungsfall können dabei mehrere User Stories enthalten sein.

Tabelle 7: Abbildung von Anwendungsfällen auf User Stories

Anwendungsfall	Zugehörige User Stories
Suche nach Arbeitsplatz	- Als KIT-Studierender möchte ich eine Übersicht über freie Arbeitsplätze auf dem Campus erhalten, um meine Lernaktvitiäten planen zu können. - Als KIT-Studierender möchte ich nach bestimmten Arbeitsplätzen suchen können, um deren Status (frei/belegt) einsehen zu können.
Filterung von Arbeitsplätzen	- Als KIT-Studierender möchte ich nach Eigenschaften der Arbeitsplätze filtern können, um über deren Eignung für meine Ansprüche entscheiden zu können.
Reservierung von Arbeitsplatz	- Als KIT-Studierender möchte ich einen Arbeitsplatz reservieren können, um an ihm für eine festgelegt Zeit arbeiten zu können. - Als KIT-Studierender möchte ich eine Übersicht über meine Reservierung von Arbeitsplätzen erhalten, um meine Termine planen zu können.
Einsicht von Reservierungsinformation	- Als KIT-Studierender möchte ich die Details meiner Reservierungen ändern können, um flexibel auf Terminänderungen reagieren zu können.
Versenden von Benachrichtigung	- Als KIT-Studierender möchte ich die Informationen über eine Arbeitsplatzreservierung an meine Lerngruppe versenden können, um meine Termine planen zu können

Identifizierung von Ressourcen

Im Kontext des SmartMeetings-Systems stellen zunächst die Informationen zu Arbeitsplätzen und Räumen Ressourcen dar, welche durch das System bereitgestellt und verarbeitet werden. Diese Informationen beschreiben die Eigenschaften von Arbeitsplätzen und stellen die Grundlage für die durch die Benutzer durchgeführte Suche sowie Filterung dar. Eine Filterung wird vorgenommen, um für die von den Benutzern durchzuführenden Aktivitäten eine Umgebung mit geeigneter Ausstattung auszuwählen. Daher verlassen sich die Benutzer auf die Korrektheit der Informationen und verbinden einen Wert mit dieser Ressource, welcher in einem persönlichen Vorteil begründet liegt. Die Informationen werden dabei durch die Arbeitsplätze selbst unter Einsatz von geeigneter Sensortechnologie bereitgestellt.

Des Weiteren stellen Reservierungen Ressourcen dar, welche von Wert für die Benutzer sind. Die Reservierung von Arbeitsplätzen ist eine essentielle Funktionalität des SmartMeetings-Systems, auf deren Korrektheit sich die Benutzer verlassen. Auch hierbei liegt der Wert der Ressource in dem persönlichen Vorteil des Benutzers, welcher durch die Reservierung in der Lage ist entsprechende Lernaktivitäten an dem vorgesehenen Arbeitsplatz durchzuführen. Da die Reservierungen zu einem bestimmten Zeitpunkt stattfinden, welche in die Tagesplanung der Benutzer einfließt, ist die Korrektheit der Reservierungsinformationen grundlegende Voraussetzung für das SmartMeetings-System.

Ermittlung des Schutzbedarfes

Analog zum Vorgehen beim KITCampusGuide wird der Schutzbedarf der identifizierten Ressourcen des SmartMeetings-Systems unter Einsatz der Schutzkategorien das BSI-Grundschutzhandbuch analysiert. Hierbei ist festzustellen, dass die Arbeitsplatzinformationen Parallelen zu den POI-Daten im KITCampusGuide aufweisen. Auch hier führen fehlerhafte Daten über einen Arbeitsplatz zu unerwünschten Ergebnissen bei der Suche nach geeigneten Plätzen für einen Benutzer. Durch die fehlerhaften Daten wird eine Aufgabenerfüllung des Benutzers verhindert, wodurch gegebenenfalls negative Auswirkungen auf den persönlichen Mehrwert des Benutzers resultieren.

An dieser Stelle profitiert die Analyse des Schutzbedarfes von den im Referenzmodell der Sicherheitsarchitektur bereits dokumentierten Schutzbedürfnisse für ähnliche Ressourcen. Durch Wiederverwendung der bisher erstellten Informationen lassen sich diese Artefakte somit ebenfalls auf die Arbeitsplatzinformationen des SmartMeetings-Szenarios anwenden und für diese ein Schutzbedarf von niedrig bis mittel festlegen. Durch den Vorlagenkatalog wurden somit die Erfahrungsinformationen dokumentiert, wodurch ein Vergleich der unterschiedlichen Anwendungsszenarien ermöglicht und die Bestimmung des Schutzbedarfes unterstützt wird.

Ähnlich verhält es sich mit dem Schutzbedarf für Reservierungsinformationen für Arbeitsplätze. Aufgrund der bereits dokumentierten Schadensszenarien und Schutzbedürfnisse im Sicherheitsanforderungskatalog lässt sich für Reservierungen ein mittlerer Schutzbedarf bestimmen. In diesem Fall führen etwa Manipulationen der Reservierungsinformationen zur Behinderungen im Arbeitsablauf von Benutzern. Zudem lassen sich durch unbefugte Einsicht in die Informationen Rückschlüsse über den Aufenthaltsort von Benutzern ziehen, wodurch deren Privatsphäre verletzt wird.

Festlegung von Schutzzielen

Ausgehend von den durch Einsatz der Vorlagen ermittelten Schutzbedürfnisse der Arbeitsplatz- und Reservierungsinformationen können anschließend angemessene Schutzziele für diese Ressourcen abgeleitet und festgelegt werden. Zunächst sollen die über das SmartMeetings-System bereitgestellten Informationen über die Ausstattung und den Zustand eines Arbeitsplatzes korrekt vorliegen. Demnach soll es unbefugten Personen oder Systemen nicht erlaubt sein, diese Informationen zu ändern. Da das Software-System das IoT-Paradigma einsetzt und die Eigenschaften der Arbeitsplätze durch Sensorabfragen verwaltet, ist demnach einer Sabotage der Sensorinformation vorzubeugen. Hierdurch werden die Schutzziele der Integrität und der Verfügbarkeit für die Ressource der Arbeitsplatzinformationen festgelegt, wodurch die Unveränderbarkeit und der Zugriff auf die Informationen durch das SmartMeetings-System garantiert werden muss.

Für die Reservierungsinformationen wird neben der Integrität zudem die Vertraulichkeit als Schutzziel zugeteilt. Grund hierfür ist zum einen, dass das SmartMeetings-System eine Garantie der Unveränderbarkeit der Informationen durch unbefugte Personen nach der Eintragung in das SmartMeetings-System vollbringen muss, so dass die Reservierungsfunktionalität vollständig erbracht werden kann. Zudem sollen die Reservierungsinformationen ausschließlich durch befugte Personen, zum Beispiel die Teilnehmer einer Lerngruppe, einsehbar sein. Der Zusammenhang zwischen Ressourcen, deren Wert und Schutzbedarf sowie den zugewiesenen Schutzzielen ist in Abbildung 88 dargestellt.

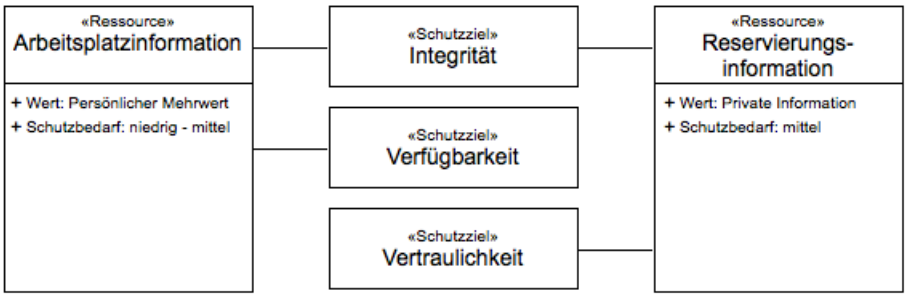

Abbildung 88: Schutzziele und Schutzbedarf des SmartMeetings-System

Bedrohungsanalyse

Bei der sich anschließenden Bedrohungsanalyse lassen sich die zuvor beschriebenen Angriffsszenarien erneut einsetzen, um Angriffe und deren Risiken sowie Auswirkungen für das SmartMeetings-System zu untersuchen. Die zutreffenden Angriffsszenarien lassen sich aus dem Vorlagenkatalog durch die

Filterung der Vorlagen mittels der zuvor bestimmten Werte für den Schutzbedarf und die Schutzziele der Ressourcen auswählen.

Wie schon erwähnt, stellt die Sabotage von Informationen zu Arbeitsplätzen ein unerwünschtes Ereignis im System dar. Ein solcher Angriff hat zum Ziel, die Integrität der Informationen zu untergraben. Hierbei kann sich zum Beispiel ein Einzeltäter einen persönlichen Vorteil verschaffen, indem er einen sonst akzeptablen Arbeitsplatz durch negative Eigenschaften ausstattet, um ihn zu seinem persönlichen Vorteil zu nutzen.

Hinsichtlich der Reservierungsinformation von Arbeitsplätzen stellen Sabotageangriffe ebenfalls unerwünschte Ereignisse dar, da hierdurch die Integrität der Ressource untergraben und eine faire Verteilung der Arbeitsplätze verhindert wird. Durch die Einsicht von unbefugten Personen auf die Reservierungsinformationen wird zudem die Privatsphäre der Benutzer verletzt. Ein Diebstahl der Information über den Aufenthaltsort der Benutzer an den reservierten Arbeitsplätzen lässt Rückschlüsse auf das Verhalten einzelner Personen zu. Dies ist durch das SmartMeetings-System zu unterbinden.

Eine weitere Bedrohung stellt die Manipulation oder das Abfangen von Benachrichtigungen über Reservierungsinformationen an Lerngruppenmitglieder dar. Hierdurch werden Mitglieder der Lerngruppe nicht über die Reservierung eines Arbeitsraumes informiert, wodurch die Lerngruppe in ihrer gemeinsamen Aktivität behindert wird und ein gemeinsamer Nachteil entsteht.

Formulierung von Sicherheitsanforderungen

Durch die bisher durchgeführten Analyseaktivitäten wurden die Sicherheitsanforderungsvorlagen auf die notwendigen Anforderungen reduziert, um den Schutzbedarf des SmartMeetings-Systems zu definieren. Die absichtliche Redundanz der Anforderungen bestätigt hierbei zum einen den hohen Grad an Wiederverwendung der Vorlagen und zum anderen die Aussage von Firesmith, dass auf

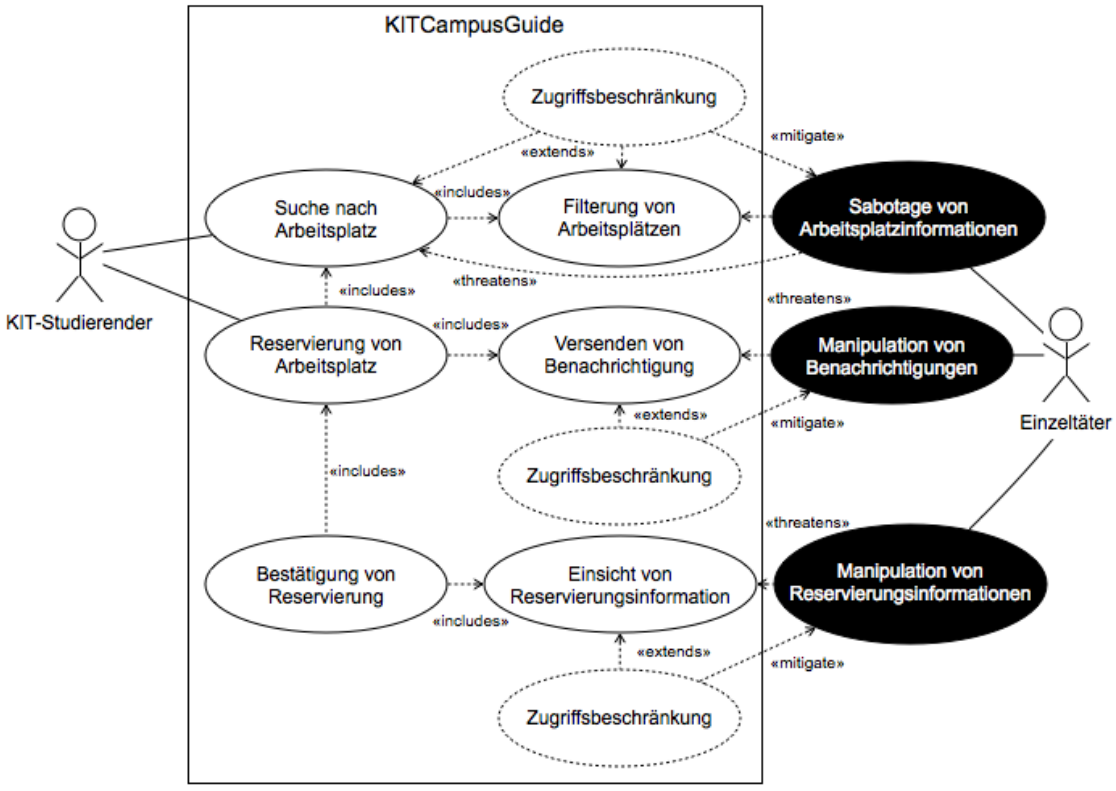

Abbildung 89: Bedrohungen und Sicherheitsanforderungen für das SmartMeetings-System

einem hohen Abstraktionsniveau verschiedene Software-Systeme ähnliche Sicherheitsanforderungen besitzen [Fi04].

Für die Arbeitsplatzinformationen werden ausgehend von den analysierten Bedrohungen Sicherheitsanforderungen definiert, durch welche das SmartMeetings-System vor unerwünschten Manipulationen geschützt werden soll und die Integrität sowie die Verfügbarkeit gewahrt wird. Durch diese Anforderung wird zum einen die Funktionalität des Eintragens von Arbeitsplatzinformationen eingeschränkt. Diese Informationen sind zunächst unabhängig von den Sensorinformationen im System abgelegt. Die Aktualität der Informationen wird durch die Abfrage der Sensorinformation an den Arbeitsplätzen gewährleistet. Daher ist diese Abfrage ebenfalls vor Manipulationen zu schützen.

Bezüglich der Reservierungsinformationen besteht folglich die Sicherheitsanforderung, dass diese ebenfalls vor unerwünschter Manipulation sowie Einsicht geschützt werden sollen. An dieser Stelle sollen die Schutzziele der Integrität und der Vertraulichkeit gewährleistet werden. Somit wird die Einschränkung, dass die Einsicht der Reservierungsinformation nur für die von der jeweiligen Person bzw. der zugehörigen Lerngruppe durchgeführten Reservierungen möglich ist, definiert. Ebenso ist eine Bestätigung einer Reservierung am Arbeitsplatz nur durch die Person durchführbar, welche die Reservierung zuvor erstellt hat.

Durch die agile Entwicklung werden die Sicherheitsanforderungen ebenfalls als User Stories ausgedrückt. In Tabelle 8 sind die Sicherheitsanforderungen so formuliert, dass sie an die Struktur zur Formulierung von User Stories angepasst sind. In Abbildung 89 werden die Anwendungsfälle des SmartMeetings-Systems zusammen mit den analysierten Angriffen sowie den entsprechenden Sicherheitsanforderungen in Form von Missbrauchs- und Sicherheitsanwendungsfällen dargestellt.

Tabelle 8: Missbrauchs-und Sicherheitsanwendungsfälle sowie zugehörige User Stories

Missbrauchs-anwendungsfall	Sicherheits-anwendungsfall	Zugehörige User Stories
Sabotage von Arbeitsplatz-informationen	Zugriffs-beschränkung	- Als KIT-Studierender möchte ich korrekte Informationen über einen Arbeitsplatz erhalten, um die Eignung des Arbeitsplatzes für meine Zwecke einschätzen zu können. - Als Facility Manager möchte ich die Modifikation von Informationen zu Arbeitsplätzen nicht für alle Benutzer zulassen, um die Korrektheit der Informationen gewährleisten zu können.
Manipulation von Benachrichtigungen	Zugriffs-beschränkung	- Als KIT-Studierender möchte ich die Benachrichtigungen über Reservierungen an meine Lerngruppe versenden können, ohne dass andere Benutzer diese Informationen einsehen oder ändern können.
Einsicht von Reservierungs-informationen	Zugriffs-beschränkung	- Als KIT-Studierender möchte ich die Informationen zu meinen Reservierungen einsehen können, ohne dass jemand anderes diese einsehen oder verändern kann.

Resümee

Der Mehrwert der aufbereiteten Sicherheitsanforderungsvorlagen hat sich in der Analyse der Sicherheitsanforderungen des SmartMeetings-Systems deutlich gezeigt. Durch die entstehenden Redundanzen reduziert sich die Lernkurve beim Einsatz der Vorlagen, wodurch eine effiziente und strukturierte Ableitung der Sicherheitsanforderungen etabliert wird. Ebenso werden durch die Wiederverwendung der Vorlagen agile Entwicklungsmethoden unterstützt, da hierdurch eine rasche

und leichtgewichtige Analyse vorgenommen werden kann. Zwar sind die Vorlagen mit zugehörigen Sicherheitsmaßnahmen verknüpft, durch die abstrakte Formulierung der Sicherheitsanforderungen bleibt die Entscheidung über deren Umsetzung jedoch der Entwurfsphase vorbehalten.

7.2.2 Entwurf der Sicherheitsarchitektur und -maßnahmen

Aus den mit Hilfe der wiederverwendbaren Vorlagen erstellten Sicherheitsanforderungen des Smart-Meetings-Systems können in der sich anschließenden Entwurfsphase abstrakte Sicherheitsmaßnahmen abgeleitet werden. Hierbei werden ähnlich wie zur Entwicklung des KITCampusGuides die Verbindungen der Sicherheitsanforderungen zu Taktiken und zugehörigen abstrakten Sicherheits-maßnahmen untersucht. Durch die bereits in der Entwicklung des KITCampusGuides analysierte wiederverwendbare Sicherheitsarchitektur am KIT, orientiert sich die Entwicklung der Sicherheits-funktionalität des SmartMeetings-Systems an dem bereits aufbereiteten Sicherheitwissen.

Auswahl von Taktiken

Die für die Ressourcen definierten Sicherheitsanforderungen beeinflussen die Auswahl der Taktiken, welche die Sicherheitsmaßnahmen zur Umsetzung der Anforderungen bestimmen. Aufgrund des hohen Abstraktionsgrades stellt sich auch hier zunächst eine Redundanz bei der Auswahl der Sicher-heitsmaßnahmen ein, welche jedoch gewollt ist und den Entwicklungsprozess für Sicherheits-maßnahmen beschleunigt. Im weiteren Verlauf des Entwurfs werden mehrheitlich die bereits analysierten Zugriffskontrollmaßnahmen modelliert.

Bei der Manipulation von Informationen bezüglich Arbeitsplätzen kann eine wiederherstellende Maßnahme hilfreich sein, um den zuletzt gültigen Stand der Informationen in dem System nach Eintritt einer Manipulation zu rekonstruieren. Dennoch tritt durch den Angriff bereits eine Einschränkung der Funktionalität für den Benutzer ein, welche es zu verhindern gilt. Eine alleinige Erkennung des Angriffs ist in diesem Fall ebenfalls nicht ausreichend, um die Schutzziele der Ressourcen zu garantieren. Aufgrund der definierten Sicherheitsmaßnahmen lassen sich daher die bereits beim KITCampusGuide eingesetzten vorbeugenden Taktiken der Zugriffskontrolle, Authentifizierung und Autorisierung einsetzen.

Eine ähnliche Situation lässt sich bei der Auswahl der Taktiken für die Reservierungsinformationen erkennen. Auch hier setzen wiederherstellende bzw. detektierende Maßnahmen die Sicherheits-anforderungen nur ungenügend um. Daher wird zur Prävention einer Manipulation und einer Einsicht von Reservierungsinformationen die Sicherheitsmaßnahme Zugriffskontrolle eingesetzt.

Die Zugriffskontrolle soll dabei zum einen durchsetzen, dass eine unbefugte Änderung der Arbeits-platzinformationen verhindert wird. Zum anderen soll die Einsicht in die Reservierungsinformationen lediglich für zugelassene Benutzer ermöglicht werden. Die Authentifizierung und die Autorisierung ergeben sich anschließend als notwendige Voraussetzungen für die Zugriffskontrolle, wie im Muster-sprachenbaum in Kapitel 6 beschrieben wurde.

Auswahl der Architekturkomponenten

In Abbildung 90 sind die fachlichen Komponenten der Architektur des SmartMeetings-System sowie die Komponenten der Sicherheitsarchitektur dargestellt. Die Funktionalität der Arbeitsplatzsuche wird von der Arbeitsplatz-Manager-Komponente implementiert, welche zusätzliche Informationen über die Arbeitsplatzausstattung aus dem Gebäudemanagementsystem des KIT erhält. Dieses umfasst zusätzlich die zuvor nicht erwähnte Funktionalität des Eintragens von Arbeitsplätzen und zugehöriger Informationen. Die Reservierungen der Arbeitsplätze werden durch eine Reservierungskomponente verwaltet. Die Architektur zeigt zunächst eine logische Aufteilung der Funktionalität auf einem hohen Abstraktionsniveau auf.

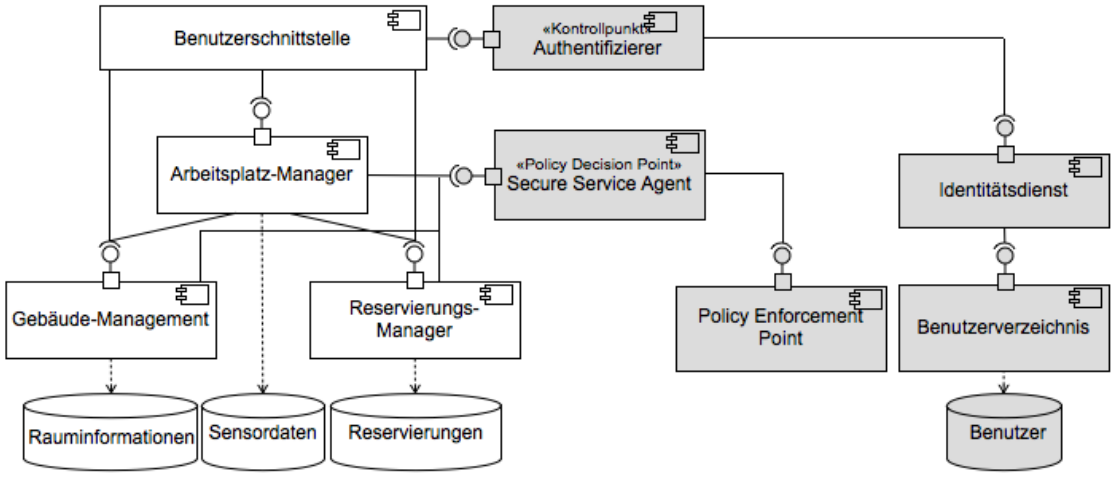

Abbildung 90: Fachliche und Sicherheitskomponenten des SmartMeetings-Systems

Bei der Betrachtung der zugehörigen Sicherheitsarchitektur fällt auf, dass sich diese auf dem gegebenen Abstraktionsniveau nicht allzu sehr von der im KITCampusGuide eingesetzten Sicherheitsarchitektur unterscheidet. Dies ist die wesentliche Eigenschaft des in Kapitel 4 vorgestellten Referenzmodells, durch welches die bestehende Sicherheitsinfrastruktur zur Aufbereitung von Sicherheitswissen berücksichtigt wird. Durch die am KIT existierenden Sicherheitsprodukte wird eine zugehörige Architektur weitestgehend vorgegeben, welche in den meisten Fällen zur Umsetzung der Sicherheitsfunktionalität von neuen Software-Systemen eingesetzt werden kann. Die Berücksichtigung der vorgegebenen Sicherheitsarchitektur ermöglicht eine effiziente Integration der neuen Software-Systeme.

Auswahl von Sicherheitsrichtlinien

Trotz der wiederverwendbaren Sicherheitskomponenten und -dienste werden zusätzliche Sicherheitsrichtlinien benötigt, um den Zugriff auf die verschiedenen fachlichen Funktionen und Ressourcen zu kontrollieren. Hierbei kann jedoch auf die vorgegebenen Richtlinienmodelle, welche durch die Sicherheitsarchitektur unterstützt und im Falle des KITCampusGuide eingesetzt wurden, zurückgegriffen werden.

Daher wird auch für das SmartMeetings-System eine Authentifizierung von Subjekten durch einen wissensbasierten Nachweis mittels Benutzername und Passwort durchgeführt. Somit wird die Authentifizierungsstärke der Nachweise auch in diesem Fall durch eine Ein-Faktor-Authentifizierung umgesetzt. Als Resultat können die bereits eingesetzten Benutzerkonten der Studierenden wiederverwendet werden, wodurch zusätzlich eine mehrmalige Authentifizierung der Benutzer umgangen wird, da der Shibboleth-Identitätsdienst des SCC für ein Single Sign-On eingesetzt werden kann.

Durch die Sicherheitsarchitektur wird bereits eine rollenbasierte Zugriffskontrolle unterstützt, welche auch in diesem Fall angewendet werden kann. Hierbei wird jedoch anhand des untersuchten Funktionsausschnitts des SmartMeetings-Systems lediglich eine Rolle eingesetzt, um die Zugriffsrechte zu kapseln. Dies macht jedoch insofern Sinn, da zahlreichen Studierenden diese Rolle zugewiesen wird und somit eine separate Zuweisung der entsprechenden Rechte an einzelne Studierende unterbunden wird.

In Abbildung 91 sind die Rechte der Rolle „ROLLE_BENUTZER“ für die identifizierten Ressourcen dargestellt. Wie analysiert, soll die Rolle eine Leseberechtigung auf Arbeitsplatzinformationen erhalten, um eine Suche nach Arbeitsplätzen und Filterung nach deren Eigenschaften durchführen zu können. Ebenso wird eine Lese- und ein Schreibrecht auf Reservierungsinformationen zugewiesen,

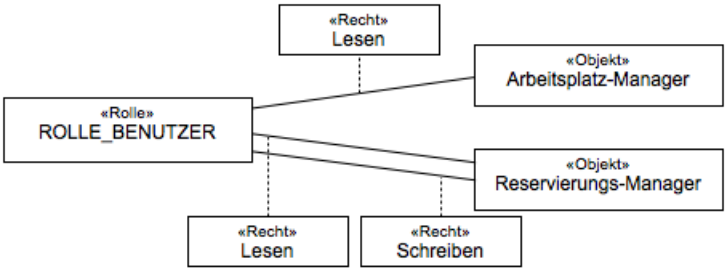

Abbildung 91: Rollen und Rechtezuweisung für das SmartMeetings-System

unter der Voraussetzung, dass der Nutzer die Reservierung selbst erstellt hat oder Teilnehmer einer Lerngruppe ist, an welche die Reservierung versendet wurde.

Feinentwurf und Technologieabbildung

Die Umsetzung der abstrakten Architektur wird im Gegensatz zum KITCampusGuide durch die OpenIoT-Middleware umgesetzt, durch welche das Paradigma des Internets-der-Dinge in dem Software-System realisiert wird. In Abbildung 92 wird der Feinentwurf des SmartMeeting-Systems unter Einsatz der Middleware dargestellt.

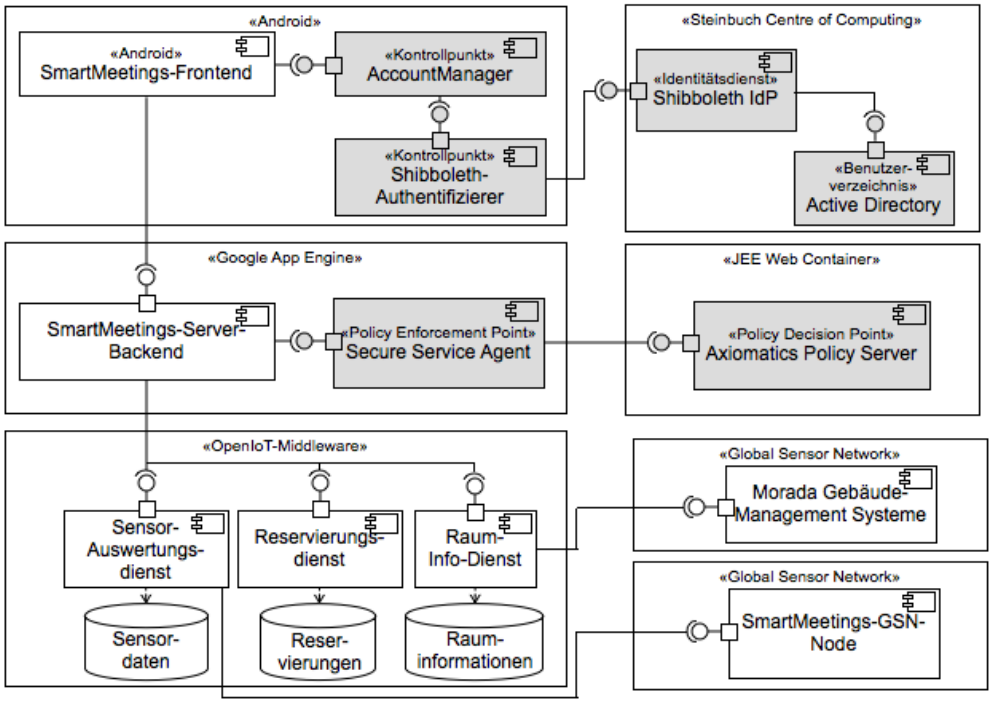

Abbildung 92: Technologische Komponenten des SmartMeetings-Systems

Die OpenIoT-Middleware stellt zunächst Dienste bereit, um die von den Arbeitsplätzen verfügbaren Sensorinformationen einzulesen und zu speichern. Hierzu wird zunächst eine Instanz eines Global-Sensor-Networks-Knotens (GSN) eingesetzt, welcher die Abfrage der Sensorinformationen und die Übertragung in die OpenIoT-Middleware übernimmt. Die Speicherung der Daten erfolgt in der Middleware anhand von semantischen Speichern unter Einsatz des Ressource Description Frameworks (RDF). Zudem lassen sich Dienste an der Middleware registrieren, welche einen Klienten über die Änderung von Sensorinformationen informieren. Die Middleware wird genutzt, um die Arbeitsplatz-informationen des KIT-Gebäudemanagementsystems einzulesen und dem SmartMeetings-System zur Verfügung zu stellen. Dabei werden lediglich die statischen Daten aus dem Gebäude-Management-

System geladen, wie zum Beispiel die unterschiedlichen Eigenschaften der Räume oder deren Ausstattung. Die Aktualisierung der Eigenschaften geschieht hingegen dynamisch über Sensoraktualisierungen.

Des Weiteren wird im Gegensatz zum KITCampusGuide keine Web-basierte Benutzerschnittstelle erstellt, sondern eine Java-basierte Anwendung für das Android-Betriebssystem entwickelt. Hierzu wird zunächst ein Server-seitiges Software-System eingesetzt, welches die Hauptfunktionalität des SmartMeetings-Systems kapselt und eine Abstraktion der technischen OpenIoT-Middleware vornimmt. Dieses Server-Backend wird auf der von Google angebotenen Platform-as-a-Service-Umgebung (PaaS) Google App Engine ausgeführt. Auf die Server-seitigen Komponenten greift die Client-seitige Android-Anwendung zu, welche die Benutzerschnittstelle der SmartMeetings-Anwendung darstellt.

Da die Sicherheitsarchitektur am KIT bereits vorgegeben ist, ändert sich die Realisierung der Komponenten im Vergleich zur KITCampusGuide-Anwendung nur geringfügig. Aufgrund der andersartigen Realisierung der Fachfunktionalität mittels anderer Technologien ist die Anpassung der sicherheitsbezogenen Komponenten notwendig, welche in unmittelbarer Nähe zur Fachfunktionalität bereitgestellt werden. In der Abbildung ist das Active Directory Benutzerverzeichnis der Übersichtlichkeit wegen nicht dargestellt.

Hierzu gehört zunächst die Authentifizierung von Benutzern auf den mobilen Android-Geräten. Für die Umsetzung des Kontrollpunkt-Musters bzw. dessen Spezialisierung eines Authentifikators wird in der Android-Plattform die dafür zuständige Klasse „AccountManager“ durch ein spezielles Modul erweitert. Während der AccountManager die Verwaltung der Authentifizierungsnachweise auf dem Mobilgerät übernimmt, behandelt das Erweiterungsmodul die Kommunikation und die Behandlung von temporären Authentifizierungs-Tokens des Shibboleth-Identitätsdienstes des KIT.

Die Umsetzung des Policy Enforcement Points wird erneut durch den Einsatz des Secure-Service-Agent-Musters durchgeführt. Hierzu wird eine spezielle Komponente entwickelt, welche im Gegensatz zu der im KITCampusGuide betriebenen Komponente auf der Google App Engine ausgeführt werden kann. Ebenso ist eine spezielle SSA-Komponente für die OpenIoT-Middleware denkbar, um die Rechte bei den von der Middleware bereitgestellten Diensten feingranular bestimmen zu können. Jedoch handelt es sich bei der Plattform um einen Prototypen, so dass während der Entwicklung eine solche Komponente nicht vorhanden ist. Die Zugriffskontrolle wird somit lediglich auf der Ebene der Server-seitig betriebenen SmartMeetings-Anwendung durchgeführt.

Resümee

Während des Entwurfs der Sicherheitsfunktionalität wurde der Mehrwert des Einsatzes von bestehendem Wissen und bewährten Methoden zur Modellierung von Sicherheitsmaßnahmen deutlich. Durch die bereits im KITCampusGuide eingesetzte Sicherheitsinfrastruktur des SCC konnten die daraus aufbereiteten Sicherheitsmaßnahmen übernommen und für die Modellierung der spezifischen Funktionalität des SmartMeetings-Systems angepasst werden. Hieraus resultiert eine direkte Übernahme der abstrakten Sicherheitsarchitektur des KIT für die SmartMeetings-Anwendung. Ebenso konnten die Konventionen der Autorisierungsrichtlinien des KITCampusGuide übernommen und durch den Axiomatics Policy Server umgesetzt werden.

7.2.3 Bewertung und Ausblick

Mit diesem Tragfähigkeitsnachweis wurde die Anwendung des sicherheitsbasierten Entwicklungsvorgehens auf Software-Systeme demonstriert, welche nach dem konzeptionell neuartigen Paradigma des Internets-der-Dinge aufgebaut sind. Hierzu wurde eine erste Analyse der benötigten Sicherheitsfunktionalität auf Anwendungsebene durchgeführt und notwendige Sicherheitsmaßnahmen zur

Authentifizierung, Autorisierung und Zugriffskontrolle umgesetzt. Ebenso wurde der Einsatz des Entwicklungsvorgehens in einem agilen Entwicklungsprozess veranschaulicht, wobei die Aktivitäten und Ergebnisartefakte an das agile Vorgehen angepasst wurden.

Die zuvor eingesetzten Vorlagen für Sicherheitsanforderungen konnten auch im Falle des Smart-Meetings-Systems erfolgreich zur Ableitung und Spezifikation von spezifischen Sicherheitsanforderungen eingesetzt werden. Hierbei wurden Anforderungen für die Ressourcen der Arbeitsplatzinformationen und der Reservierungen, sowie die zugehörigen funktionalen Anforderungen in unterschiedlichen Entwicklungszyklen des agilen Vorgehens spezifiziert. Auf diese Weise wurden die Sicherheitsanforderungen iterativ erhoben und das System um entsprechende zusätzliche Funktionalität erweitert. Dadurch wurde gezeigt, dass die Aktivitäten des in dieser Arbeit vorgestellten sicherheitsbasierten Entwicklungsvorgehens auf ein agiles Prozessmodell angepasst werden können.

Im Vergleich zum vorhergehenden Tragfähigkeitsnachweis wurde das System zum einen durch eine Benutzeroberfläche für das Android-Betriebssystem für Mobilgeräte und zum anderen durch die Realisierung der Funktionalität auf der Cloud-basierten Ausführungsumgebung Google App Engine umgesetzt. Ebenso wurde das neuartige Paradigma des Internets-der-Dinge eingesetzt, um Sensordaten in das Software-System zu integrieren. Dabei wurde deutlich, dass die bereitgestellte Funktionalität der Sicherheitsarchitektur größtenteils wiederverwendbar ist. Es war lediglich eine Anpassung der sicherheitsbezogenen Komponenten, wie etwa ein spezifischer Policy Enforcement Point, zur Integration der fachlichen Funktionalität notwendig. Durch die Sicherheitsmustersprachen konnte hierzu ebenfalls eine Unterstützung bei der Umsetzung der Komponenten geliefert werden.

In der weiteren Entwicklung des SmartMeetings-Systems wird der Schutz der OpenIoT-Middleware vor Missbrauch und die Einschränkung der Verarbeitung der Sensordaten eine grundlegende Rolle spielen. So sind im momentanen Entwicklungsstatus der OpenIoT-Middleware keine Sicherheitsvorkehrungen vorgesehen, da es sich um einen Prototypen handelt. Für einen realen Einsatz sind jedoch sinnvolle Sicherheitsmaßnahmen entscheidend. In dieser Hinsicht sind jedoch noch offene Fragestellung zur Umsetzung von Sicherheitsmaßnahmen für die in der Middleware eingesetzten semantischen Datenspeicher zu klären und bewährte Methoden zu erarbeiten, welche anschließend aufbereitet und in ein Referenzmodell für eine Sicherheitsarchitektur der Middleware einfließen können. Durch die Analyse der Sicherheitsanforderungen und die Modellierung der Sicherheitsmaßnahmen des SmartMeetings-Systems wurde ein erster Hinweis auf notwendige Maßnahmen für die OpenIoT-Middleware gegeben. Die Absicherung des SmartMeetings-Systems wurde somit lediglich auf der Anwendungsebene vorgenommen. Für eine mehrschichtige Absicherung sind die weiteren Ebenen des Systems, wie zum Beispiel Datenhaltung und Ausführungsumgebung, zu berücksichtigen.

7.3 Resümee

In diesem Kapitel wurden die Beiträge dieser Arbeit eingesetzt, um die Sicherheitsfunktionalität von zwei Software-Systemen aus unterschiedlichen Domänen strukturiert zu entwickeln. Dabei wurde das in Kapitel 4 vorgestellte sicherheitsbasierte Software-Entwicklungsvorgehen eingesetzt, um die Phasen eines fachlichen Entwicklungsprozesses um sicherheitsbezogene Aktivitäten zu erweitern. Die Aktivitäten wurden an die unterschiedlichen Entwicklungsprozesssmodelle der Software-Systeme angepasst. Zum einen wurde ein traditioneller Entwicklungsprozess nach dem Wasserfallprinzip angewendet, zum anderen wurde das agile Entwicklungsrahmenwerk Scrum eingesetzt. Hierdurch wurde zunächst die Anpassbarkeit der Aktivitäten des sicherheitsbasierten Entwicklungsvorgehens demonstriert. Des Weiteren wurde durch die Anwendung der Aktivitäten in verschiedenen Prozessmodellen deren generische Einsatzfähigkeit veranschaulicht.

Durch die im sicherheitsbasierten Entwicklungsvorgehen vorgegebenen Aktivitäten waren die durchzuführenden Schritte zur Analyse und Spezifizierung von Sicherheitsanforderungen sowie zur Modellierung von Sicherheitsmaßnahmen jederzeit klar definiert. Die bereitgestellten wiederverwendbaren sicherheitsbezogenen Entwicklungsartefakte trugen zudem zu einer strukturierten und methodischen Entwicklung der Sicherheitsfunktionalität für die untersuchten Software-Systeme bei. So konnte durch die Instanziierung der Sicherheitsanforderungsvorlagen eine zielgerichtete Analyse der Anforderungen der Software-Systeme durchgeführt werden, in dem die in den Vorlagen verknüpften Werte als Vorgabe zur Untersuchung der Sicherheitsprobleme und notwendigen Schutzbedürfnisse der Software-Systeme eingesetzt wurden. In der Entwurfsphase boten die Sicherheitsmustersprachen eine zweckmäßige Unterstützung bei der Auswahl von geeigneten Lösungen zu Sicherheitsproblemen auf unterschiedlichen Abstraktionsniveaus der Entwurfsmodelle.

Mit Hilfe der in Kapitel 5 vorgestellten Sicherheitsanforderungsvorlagen wurde die Analyse von Sicherheitsanforderungen der Software-Systeme durchgeführt. Durch den hohen Wiederverwendungsgrad der aufbereiteten Sicherheitsanforderungen, konnten die Sicherheitsanforderungen der Software-Systeme effizient bestimmt und spezifiziert werden. Hierbei wurden auch die damit verbundenen Aktivitäten der Ressourcenidentifikation und Wertbestimmung sowie die Festlegung des Schutzbedarfes und der Schutzziele einer Ressource unterstützt. Mittels einer Filterung der Vorlagen anhand der bestimmten Werte für Ressourcen konnten angemessene Sicherheitsanforderungen für die Software-Systeme ermittelt werden. Dabei wurde deutlich, dass die Beiträge dieser Arbeit als umfassendes Rahmenwerk dienen, in welches bestehende Ansätze zur sicherheitsbasierten Entwicklung, wie zum Beispiel Missbrauchs- und Sicherheitsanwendungsfälle sowie Sicherheitsmuster, integriert werden können.

Die Modellierung von Sicherheitsmaßnahmen wurde durch die in Kapitel 6 besprochenen Sicherheitsmustersprachen, welche auf dem vorgestellten Variabilitätsmodell basieren, veranschaulicht. Durch die Sicherheitsmustersprachen wurde die Ableitung von Sicherheitsmaßnahmen aus den spezifizierten Anforderungen unterstützt. Zudem wurde durch die Kategorisierung der verschiedenen Sicherheitsmuster in unterschiedliche Abstraktionsniveaus eine Hilfestellung bei der durchgängigen Erstellung einer Sicherheitsarchitektur sowie deren Feinentwurf gegeben. In jedem Verfeinerungsschritt lag es den Entwicklern frei, die notwendigen Sicherheitsmaßnahmen hinsichtlich der gegebenen Sicherheitsprobleme der Software-Systeme auszuwählen und zu verfeinern. Durch den Einsatz von Sicherheitsmustern, wurden zum einen bewährte Methoden zur Lösung eingesetzt und zum anderen die Auswahlmöglichkeit für Entwickler klar beschrieben. Des Weiteren wurde eine Integration in die fachliche Entwicklung ermöglicht.

Die Entwicklung der Sicherheitsfunktionalität für die Software-Systeme entsprach dabei der Anwendungsphase des sicherheitsbasierten Entwicklungsprozesses. Die zur Unterstützung der Entwicklung eingesetzten Artefakte basierten auf den bereits in den Kapitel 5 und 6 beschriebenen Beispielen zur Aufbereitung von Sicherheitsartefakten aus bestehenden Sicherheitsinfrastrukturen. In den Tragfähigkeitsnachweisen wurden diese zusätzlich mit der am KIT bestehenden Sicherheitsinfrastruktur verknüpft.

Durch das im Kapitel 4 vorgestellte Referenzmodell für Sicherheitsarchitekturen wurden die bestehenden Sicherheitsprodukte, welche am KIT eingesetzt werden, hinsichtlich ihrer bereitgestellten Funktionalität aufbereitet. Die dadurch erstellten Sicherheitsartefakte der Aufbereitungsphase lieferten einen konkreten Mehrwert zur Realisierung der Sicherheitsfunktionalität der Software-Systeme. Es konnte anschaulich demonstriert werden, wie durch die Wiederverwendung von bestehendem Sicherheitswissen in einem Software-Entwicklungsprozess die Entwicklung der Sicherheitsfunktionalität von Software-Systemen effizient durchgeführt werden kann.

Im Gegensatz zu den in Kapitel 3 untersuchten Ansätzen zeichnet sich das vorgestellte Entwicklungsvorgehen durch eine durchgängige und nachvollziehbare Entwicklung von Sicherheitsmaßnahmen aus. Insbesondere wird dabei die Abbildung der Entwicklungsartefakte zwischen den Entwicklungsphasen berücksichtigt. Hierbei wird die Umsetzung von Sicherheitsanforderungen auf abstrakte Sicherheitsmaßnahmen sowie die Realisierung der Maßnahmen durch konkrete Technologien klar definiert. Zudem wird die Verfeinerung der Entwicklungsartefakte innerhalb einer Entwicklungsphase unterstützt. Dabei wird ein Rahmen geschaffen, in welchem die bestehenden Ansätze integriert werden und dadurch zielgerichtet eingesetzt werden können.

8 Bewertung und Ausblick

Das folgende Kapitel dient der Zusammenfassung und Bewertung der in der vorliegenden Arbeit erzielten Ergebnisse. Die Zusammenfassung gibt dabei die wesentlichen Punkte der Beiträge in den Kapiteln 4 bis 6 wieder. Anschließend erfolgt die Bewertung dieser Beiträge auf Basis des in Kapitel 3.1 vorgestellten Anforderungskataloges an ein sicherheitsbasiertes Entwicklungsvorgehen. Aus den Beiträgen dieser Arbeit ergeben sich weiterführende Fragestellungen, welche jedoch nicht im Rahmen dieser Arbeit behandelt wurden. Ein Ausblick auf diese Themen schließt diese Arbeit ab.

8.1 Beiträge der Arbeit

Zur Entwicklung von sicheren Software-Systemen steht eine Vielzahl an existierendem Sicherheitswissen bereit, welches zur Analyse von Sicherheitsanforderungen und Modellierung von Sicherheitmaßnahmen eingesetzt werden kann. Hierzu muss dieses Wissen jedoch so aufbereitet werden, dass es sich in einen fachlichen Entwicklungsprozess integrieren lässt. In der Literatur werden hierfür bereits Ansätze vorgestellt, welche auf wiederverwendbaren Sicherheitsanforderungen und Sicherheitsmustern basieren. Diese bilden die Grundlage für eine Integration von Sicherheitswissen in einem Entwicklungsprozess unter Einsatz von wiederverwendbaren sicherheitsbezogenen Entwicklungsartefakten.

Die vorliegende Arbeit hatte zum Ziel, ein Vorgehen für die durchgängige und systematische Entwicklung von Sicherheitsfunktionalität eines Software-Systems zu schaffen, welches bestehendes Sicherheitswissen berücksichtigt. Im Zentrum stand dabei die Unterstützung von Entwicklern bei der Navigation und der Auswahl des Sicherheitswissens, so dass ein optimaler Schutz für ein Software-System umgesetzt werden kann. Zur Behandlung dieser Fragestellungen wurden als Ergebnis dieser Arbeit drei Beiträge vorgestellt, deren Tragfähigkeit bezüglich der Anwendbarkeit und Effektivität in der Praxis anhand von zwei konkreten Anwendungsfällen nachgewiesen wurde.

8.1.1 Sicherheitsbasierte Analyse und Entwurf

Der erste Beitrag umfasst ein Vorgehensmodell für die Aufbereitung von Sicherheitswissen und dessen Einsatz in der Entwicklung von sicheren Software-Systemen. Hierbei wurden insbesondere die Analysephase zur Identifizierung von Sicherheitsanforderungen und die Entwurfsphase zur Modellierung von Sicherheitsmaßnahmen fokussiert. Als Ausgangspunkt für dieses Vorgehensmodell diente das Paradigma der Software-Produktlinien, wodurch eine Trennung zwischen der Aufbereitung und dem Einsatz des Sicherheitswissens vorgenommen wurde.

Die Aufbereitungsphase des Vorgehens hat zum Ziel, existierendes Sicherheitswissen für den Einsatz in einem Software-Entwicklungsprozess vorzubereiten. Hierzu werden sicherheitsbezogene Wissensquellen, wie beispielsweise bestehende Software-Systeme, existierende Sicherheitsstandards und -produkte sowie geschäftliche Sicherheitsrichtlinien und gesetzliche Vorschriften, herangezogen, um wiederverwendbare Sicherheitsanforderungen bzw. Sicherheitsmaßnahmen zu identifizieren und als Entwicklungsartefakte zu beschreiben.

Die Artefakte der Aufbereitung wurden dabei als Teil eines Referenzmodells für Sicherheitsarchitekturen spezifiziert. Dieses Modell teilt eine Sicherheitsarchitektur in verschiedene Sichten ein, welche für die Aufbereitung und die Berücksichtigung von Sicherheitswissen in einem Entwicklungsprozess relevant sind. Als Basis dient dabei eine dienstorientierte Betrachtung der bereitgestellten Sicherheitsfunktionalität, wie sie bereits in [EB+07] behandelt wurde. Die so in einer Dienstsicht

definierten Konventionen und Richtlinien für sicherheitsbezogene Schnittstellen und Standards, dienen als Rahmen für die Aufbereitung der Sicherheitsmaßnahmen und Sicherheitsanforderungen in einer Entwicklungssicht.

In einer Anwendungsphase werden die beschriebenen Entwicklungsartefakte anschließend eingesetzt, um eine effektive Entwicklung von Sicherheitsmaßnahmen für ein konkretes Software-System zu ermöglichen. Hierbei wird durch die zuvor spezifizierten Artefakte eine initiale Auswahl an zu berücksichtigenden Sicherheitsanforderungen und -maßnahmen vorgegeben und eine Entscheidungsunterstützung beim Einsatz der Artefakte geliefert.

8.1.2 Vorlagen für Sicherheitsanforderungen

Die Aufbereitungsphase des im ersten Beitrag vorgestellten Entwicklungsvorgehens stellt wiederverwendbare Sicherheitsanforderungen für den Einsatz in einem Software-Entwicklungsprozess bereit. Zur Spezifikation und Dokumentation der wiederverwendbaren Sicherheitsanforderungen wurde im zweiten Beitrag ein Domänenmodell vorgestellt, mittels welchem Anforderungen als Vorlagen beschrieben und in einem Katalog abgelegt werden können.

Das Domänenmodell legt dabei die wesentlichen Konzepte zur Beschreibung von Sicherheitsanforderungen und deren Relationen untereinander fest. Ziel hierbei ist es, eine einheitliche Definitionsgrundlage zu erstellen, um eine Kommunikationsgrundlage zwischen Entwicklern und Sicherheitsexperten zu schaffen. Dabei werden die Konzepte in unabhängige Module gekapselt, um eine flexible Anpassung an bestehende Domänenmodelle zu ermöglichen. Das Domänenmodell ist dabei zunächst auf die Grundelemente beschränkt, wodurch eine Erweiterung des Modells um zusätzliche Konzepte zur Anpassung an spezifische Entwicklungsprozesse ermöglicht wird.

Durch das Domänenmodell gestützt, werden Vorlagen für Sicherheitsanforderungen beschrieben, welche unter Berücksichtigung des Schutzbedarfes und der Schutzziele einer Ressource, abstrakte Anforderungen zur Vermeidung oder Verminderung von Bedrohungen gegenüber der Ressource darstellen. Durch die Verwendung von Parametern, können die Vorlagen bei der Entwicklung von konkreten Software-Systemen beispielsweise durch Festlegung von fachlichen Ressourcen instanziiert werden. Hierdurch wird die Spezifikation von Sicherheitsanforderungen erleichtert. Beim Aufbau der Vorlagen wurde auf die relevanten Vorarbeiten von Firesmith und Sindre et al. [Fi04] [SF+03] zurückgegriffen.

8.1.3 Variabilitätsmodell für Sicherheitsmuster

Im Kontext des sicherheitsbasierten Entwicklungsvorgehens werden die instanziierten Sicherheitsanforderungen in der Entwurfsphase auf Sicherheitsmaßnahmen abgebildet. Für den iterativen Verfeinerungsprozess der Sicherheitsmaßnahmen von einer abstrakten Architekturmodellierung zu einem technologienahen Entwurf wurde im dritten Beitrag ein Variabilitätsmodell vorgestellt, mittels dessen alternative Sicherheitsmaßnahmen spezifiziert werden können. Als Beschreibungsmittel für Sicherheitsmaßnahmen wurde auf dem Ansatz von Sicherheitsmustern aufgebaut [SF+05].

Das Variabilitätsmodell ermöglicht es, unterschiedliche Beziehungen zwischen Sicherheitsmustern zu beschreiben. Hierdurch werden notwendige und optionale Abhängigkeiten zwischen den Mustern definiert, so dass eine Auswahl von Sicherheitsmaßnahmen getroffen werden kann. Das Variabilitätsmodell beschreibt dabei eine baumartige Hierarchie der Sicherheitsmuster, in der durch verschiedene Abstraktionsstufen der Spezialisierungsgrad der Muster zunimmt. Eine Modularisierung für eine spezifische Sicherheitsmaßnahme wird durch abstrakte Muster an der Wurzel des Variabilitätsmodells vorgenommen. Diese werden in weiteren Schichten durch ein oder mehrere spezifischere Sicherheits-

muster umgesetzt. Die Kindknoten des Baumes bestehen aus Sicherheitsmustern, welche für spezifische Sicherheitsstandards oder -produkte gelten. Zur Umsetzung des Variabilitätsmodells wurde die Semantik der Funktionsmodelle angepasst, mit deren Hilfe bereits alternative und gemeinsame Funktionen beschrieben werden können [CE02].

Die in einem Variabilitätsmodell beschriebenen Beziehungen führen zu Mustersprachen für die Sicherheitsmaßnahmen. Als Beispiel für relevante Sicherheitsmustersprachen wurden bekannte Sicherheitsmodelle und -standards für die Bereiche Authentifizierung, Autorisierung und Zugriffskontrolle aufbereitet und in die Hierarchie des Variabilitiätsmodells eingeordnet. Diese beschreiben wiederverwendbare Sicherheitsmaßnahmen für die angegebenen Bereiche, welche in zukünftigen Entwicklungen eingesetzt werden können.

8.1.4 Tragfähigkeitsnachweise

Die Tragfähigkeit der in dieser Arbeit entwickelten Beiträge im Sinne ihrer Anwendbarkeit und Effektivität wurde anhand der Entwicklung von zwei Software-Systemen demonstriert. Die Beiträge wurden dazu genutzt, um das Sicherheitsbedürfnis der Software-Systeme zu analysieren und entsprechende Sicherheitsfunktionalität zu entwickeln. Die betrachteten Software-Systeme sind in unterschiedlichen fachlichen Domänen angesiedelt, wodurch die Wiederverwendbarkeit der entwickelten Sicherheitsartefakte auf heterogene Anwendungsgebiete demonstriert wird.

Zunächst wurde mit dem KITCampusGuide der Forschungsgruppe C&M ein Software-System untersucht, welches die Benutzer in alltäglichen Aktivitäten auf dem Campusgelände des Karlsruher Instituts für Technologie (KIT) unterstützt. Hierzu werden den Benutzern Informationen über wichtige Orte auf dem Campus in einer kartenbasierten Darstellung angezeigt und eine Navigationsunterstützung bei Wegen über den Campus bereitgestellt. Mittels der in dieser Arbeit vorgestellten Beiträge wurden die Sicherheitsanforderungen des KITCampusGuides analysiert und durch Einsatz von geeigneten Sicherheitsmaßnahmen umgesetzt. Hierbei wurde insbesondere darauf geachtet, dass die dargestellten Daten des KITCampusGuide korrekt sind und nicht mit Absicht der Rufschädigung von Angreifern sabotiert werden können. Ebenso wurden die bei der Navigationsunterstützung abgerufenen Positionsdaten eines Benutzers vor Diebstahl und Manipulation durch Angreifer geschützt, so dass einer Überwachung von Personen vorgebeugt wurde. Die dabei entwickelten Sicherheitsmaßnahmen berücksichtigen die bestehenden Sicherheitstechnologien, welche bereits am KIT eingesetzt werden, so dass das entwickelte Software-System in die Anwendungslandschaft des KIT integriert werden konnte.

Als zweites Software-System wurde mit dem SmartMeetings-Projekt ein Forschungsprototyp untersucht, welcher im Rahmen des EU-Projektes OpenIoT in der Forschungsgruppe C&M in Kooperation mit dem Fraunhofer IOSB entwickelt wurde. In diesem System wird das Paradigma des Internets-der-Dinge (IoT) eingesetzt, um die Belegung von Arbeitsräumen und -plätzen für Studierende am KIT effizient zu verwalten und Reservierungen für die Arbeitsplätze zu vergeben. Durch den Einsatz des IoT-Paradigmas werden hierzu die Arbeitsplätze mit Sensoren ausgestattet, deren Daten in einer semantischen Datenbank verarbeitet werden. Durch diese Vorgehensweise entstehen neue Sicherheitsprobleme, welche zum Teil noch nicht untersucht wurden. Ziel der Anwendung der vorgestellten Beiträge dieser Arbeit war es daher, das existierende Sicherheitswissen einzusetzen, um die Sicherheitsanforderungen des SmartMeetings-Systems zu analysieren und notwendige Sicherheitsmaßnahmen umzusetzen, so dass eine erste Absicherung des Software-Systems erfolgen konnte.

Die Anwendbarkeit des in dieser Arbeit vorgestellten sicherheitsbasierten Entwicklungsvorgehens sowie der dazugehörigen Entwicklungsartefakte konnte durch den Einsatz in der Entwicklung der Software-Systeme geeignet demonstriert werden. Das aufbereitete Sicherheitswissen in Form der

entwickelten Vorlagen für Sicherheitsanforderungen und dem Variabilitätsmodell für Sicherheitsmuster unterstützten die strukturierte Entwicklung der Sicherheitsfunktionalität der Software-Systeme. Die Untersuchung der Ergebnisse zeigte, dass durch den Einsatz der Beiträge die gewünschten Resultate für den Schutzbedarf umgesetzt werden konnten. Zudem konnten die an ein sicherheitsbasiertes Entwicklungsvorgehen gestellten Anforderungen erfüllt werden, wodurch die Effektivität der entwickelten Konzepte dieser Arbeit nachgewiesen werden konnte.

8.2 Diskussion der Ergebnisse

Der in Kapitel 3.1 aufgestellte Anforderungskatalog wird im Folgenden verwendet, um die Beiträge dieser Arbeit und die damit erzielten Ergebnisse kritisch zu diskutieren. Hierdurch wird bewertet, inwiefern auf den motivierten Handlungsbedarf in Kapitel 3.3 eingegangen wurde und die an diese Arbeit gestellten Anforderungen erfüllt wurden.

8.2.1 Integrationsfähigkeit

Die Integrationsfähigkeit forderte die Unabhängigkeit der spezifizierten Aktivitäten und Artefakte eines Entwicklungsvorgehens von spezifischen Entwicklungsprozessmodellen. Zudem sollten die Aktivitäten und Artefakte flexibel an beliebige Prozessmodelle angepasst werden können.

Das im Beitrag B1 vorgestellte sicherheitsbasierte Entwicklungsvorgehen definiert Aktivitäten, welche zunächst unabhängig von einem spezifischen Entwicklungsmodell sind. Durch die Aufteilung des Vorgehens bleiben die Aktivitäten der Aufbereitungsphase unabhängig von den Aktivitäten der Anwendungsphase, wodurch letztere in beliebige Entwicklungsprozesse integriert und an diese angepasst werden können. Ebenso wird eine Trennung zwischen den einzelnen Entwicklungsphasen vorgenommen, so dass die Aktivitäten jeder Phase modularisiert und austauschbar sind. Durch die Ergebnisartefakte der einzelnen Phasen wird eine klar definierte Schnittstelle durch die benötigten Ein- und Ausgabeartefakte definiert. Hierzu dienen insbesondere die durch die Aufbereitungsphase bereitgestellten Entwicklungsartefakte. Diese basieren auf etablierten und inhärent flexiblen softwaretechnischen Werkzeugen, wie beispielsweise Anwendungsfälle und Entwurfsmuster, so dass deren Integration und Anpassung an spezifische Umstände eines Entwicklungsprozesses ebenfalls möglich wird.

Durch die Trennung der Aufbereitung von der Anwendung des Sicherheitswissens wird zudem ein konzeptionell neuer Ansatz zur Entwicklung von sicheren Software-Systemen eingesetzt, wodurch die durchzuführenden Aktivitäten in der Anwendungsphase reduziert werden und somit eine leichtgewichtige Integration ermöglicht wird. Die Aufbereitungsphase stellt in der Entwicklung eine neue Ebene dar, welche in Software-Entwicklungsprozessen außerhalb des Paradigmas der Software-Produktlinien nicht explizit betrachtet wird. Hierdurch wird die Komplexität bei der Entwicklung von Sicherheitsfunktionalität auf die zuständigen Sicherheitsexperten verlagert und fachliche Entwickler entlastet. Dabei stellen abgeschlossene Aufbereitungsaktivitäten eine initiale Mehrbelastung dar, dessen Kosten-Nutzen-Effekt sich gegebenenfalls erst langfristig herausstellt.

Eine konkrete Integration der Aktivitäten und der Artefakte wurde in den Tragfähigkeitsnachweisen demonstriert. Hierbei wurde das sicherheitsbasierte Entwicklungsvorgehen sowohl in ein traditionelles Wasserfallmodell als auch in eine agile Entwicklungsmethode erfolgreich integriert, wodurch die Anpassungsfähigkeit des Ansatzes nachgewiesen wurde.

8.2.2 Nachvollziehbarkeit

Die Forderung nach einer nachvollziehbaren Entwicklung der sicherheitsbezogenen Funktionalität bezog sich auf die Plausibilität von Entscheidungen im Entwicklungsprozess. In den Beiträgen dieser Arbeit wurde zur Umsetzung dieser Anforderung eine durchgängige Betrachtung der relevantesten Entwicklungsphasen vorgenommen. Das vorgestellte Entwicklungsvorgehen konzentriert sich auf die Anforderungsanalyse und die Entwurfsphase. Durch die Aufbereitung von existierendem Sicherheitswissen wird eine Verknüpfung zu bestehenden Sicherheitsprodukten und -standards hergestellt, wodurch der Anteil der Eigenentwicklungen in der Implementierungsphase reduziert wird.

Des Weiteren wurde eine strikte Trennung zwischen den konzeptionellen Phasen vorgenommen, so dass eine klare Zuordnung von Aktivitäten und Entwicklungsartefakten zu den einzelnen Phasen besteht. Durch die vorgestellten Entwicklungsartefakte wird eine Schnittstelle zwischen den Phasenaktivitäten erstellt und somit eine Modularisierung der Aktivitäten erreicht.

Die modularen Aktivitäten der Entwicklungsphasen werden durch die vorgestellten Entwicklungsartefakte verknüpft. Zum einen sind die in den Vorlagen beschriebenen Sicherheitsanforderungen mit ein oder mehreren abstrakten Sicherheitsmaßnahmen verbunden, welche in der zugehörigen Mustersprache den Ausgangspunkt für die iterative Verfeinerung der Maßnahmen darstellen. Zum anderen ist das Ziel der Verfeinerung der Maßnahmen die Abbildung auf existierende Sicherheitsstandards oder bereitgestellte Sicherheitsdienste, so dass hier eine Verknüpfung zur Implementierungs- und Betriebsphase besteht.

Das Variabilitätsmodell, auf welches sich die Sicherheitsmustersprachen stützen, stellt die Möglichkeit bereit, alternative Sicherheitsmaßnahmen in Form von Mustern einzuordnen und in Beziehung zu setzen. Hierdurch wird eine Auswahl zwischen alternativen Sicherheitsmustern sowie eine Entscheidungsunterstützung ermöglicht, welche durch die Musterbeschreibungen stattfindet. An dieser Stelle werden jedoch Vorkenntnisse über die eingeordneten Sicherheitsmuster vorausgesetzt, da die Anforderungen der Dokumentation der Entwurfsentscheidungen in dieser Arbeit nicht direkt im Variabilitätsmodell vorgenommen wurde. Sind die notwendigen Vorkenntnisse vorhanden, so lassen sich die getroffenen Entscheidungen bei einer Evolution der Sicherheitsfunktionalität eines Software-Systems nachvollziehen. Durch die Vorlagen werden ebenfalls unterschiedliche alternative Sicherheitsanforderungen dargestellt und eine Auswahl von unterschiedlichen Anforderungen basierend auf den identifizierten Werten für Schutzbedarf, Bedrohung, etc. ermöglicht.

8.2.3 Wiederverwendbarkeit

Zentraler Fokus der vorliegenden Arbeit war die Umsetzung der Anforderung der Wiederverwendung von existierendem Sicherheitswissen. Diese Anforderungskategorie beinhaltet die Aufbereitung und den Einsatz von existierenden Sicherheitsmodellen, -standards und -technologien in der Entwicklung von Sicherheitsfunktionalität eines Software-Systems. Hierbei wird die Definition der verwendeten Sicherheitskonzepte sowie eine Entscheidungsunterstützung bei der Auswahl der verschiedenen Sicherheitsmaßnahmen verlangt.

Durch das im Beitrag B2 im Kapitel 5 vorgestellte Domänenmodell für Sicherheitsanforderungen wurde eine grundlegende Definition von notwendigen Sicherheitskonzepten vorgenommen, welche für die Entwicklung von Sicherheitsanforderungen und -maßnahmen und zur einheitlichen Kommunikation zwischen Entwicklern und Sicherheitsexperten eingesetzt werden kann. Diese Definition ist die Basis zur Strukturierung von Vorlagen für Sicherheitsanforderungen, welche wiederverwendbare Kombinationen von Schutzzielen, Bedrohungen und entsprechenden Sicherheitsanforderungen darstellen. Durch den Einsatz der Vorlagen wird eine effiziente Analyse der

Anforderungen ermöglicht, da unterschiedliche Software-Systeme zumeist gleiche oder ähnliche Sicherheitsanforderungen besitzen.

Weiterhin wird für den Entwurf von Sicherheitsmaßnahmen durch die Baumstruktur des Variabilitätsmodells im Beitrag B3 im Kapitel 6 eine Kategorisierung der Sicherheitsmustersprachen hinsichtlich der von ihnen behandelten Sicherheitsproblemen durchgeführt. Für das Variabilitätsmodell selbst wurde der Ansatz von Funktionsmodellen als Grundlage verwendet, um optionale sowie verpflichtende Beziehungen zwischen verschiedenen Sicherheitsmaßnahmen aufzuzeigen. Hierdurch wird eine Auswahl und Entscheidungsfindung beim Einsatz der Sicherheitsmuster ermöglicht. Ebenso wurden Sicherheitsmuster zur Beschreibung der Maßnahmen eingesetzt, da diese bewährte Lösungen zu existierenden Sicherheitsproblemen beschreiben. In diesem Rahmen wurden zur Aufbereitung bekannte Sicherheitskonzepte wie zum Beispiel verschiedene Zugriffskontrollmodelle sowie Architekturlösungen für Authentifizierung und Zugriffskontrolle eingeordnet, so dass ein strukturierter Einsatz dieser Lösungsverfahren ermöglicht wurde.

Grundlage der Aufbereitung von Sicherheitswissen in dem vorgestellten sicherheitsbasierten Entwicklungsvorgehen ist die Berücksichtigung von existierenden Sicherheitsinfrastrukturen. Diese werden aus den in einer Organisation eingesetzten Sicherheitsprodukten, -rahmenwerken und -standards gebildet. Durch das im Kapitel 6 eingeführte Referenzmodell für Sicherheitsarchitekturen wird das in einer Sicherheitsinfrastruktur vorhandene Wissen für die Aufbereitung in einem Software-Entwicklungsprozess vorbereitet. Dies ermöglicht die Einschränkung der ansonsten zahlreichen zu berücksichtigenden Sicherheitsmaßnahmen in dem Variabilitätsmodell für Sicherheitsmustersprachen. Durch die Einschränkung werden nur diejenigen Sicherheitsmaßnahmen in der Entwicklung berücksichtigt, welche in der Sicherheitsinfrastruktur vorhanden sind, wodurch sich das entwickelte Software-System integrieren lässt.

8.2.4 Praktikabilität

Die Anforderung der Praktikabilität wurde in der vorliegenden Arbeit durch Berücksichtigung und den Einsatz von verbreiteten Standards zur Beschreibung von Entwicklungsartefakten umgesetzt. So wurde das Domänenmodell, welches die Grundlage für die Sicherheitsanforderungsvorlagen darstellt, auf Basis des UML-Metamodells definiert. Die Sicherheitsanforderungen fanden durch Missbrauchs- und Sicherheitsanwendungsfälle konkrete Modellierungselemente, welche sich in UML-Anwendungsfalldiagramme einbetten lassen. Ebenso werden die Sicherheitsmuster durch UML-Struktur- und Verhaltensdiagramme beschrieben. Die für das Variabilitätsmodell verwendeten Funktionsmodelle stellen zwar ein bekanntes Werkzeug in der Software-Produktlinienentwicklung dar, jedoch weicht die in dieser Arbeit verwendete Semantik der Modelle von dem ursprünglichen Einsatz ab. Ebenso ist der Einsatz der Funktionsmodelle in der allgemeinen Software-Entwicklung unüblich, wodurch sich die Praktikabilität reduziert.

Eine Automatisierung der Entwicklung von sicherheitsbezogener Funktionalität wurde in der vorliegenden Arbeit nicht explizit betrachtet. Jedoch bietet die Vorgehensweise durch die dienstorientierte Berücksichtigung von bestehender Sicherheitsfunktionalität sowie die Aufbereitung und Ableitung von existierendem Sicherheitswissen gemäß dem Paradigma der Software-Produktlinien eine gute Ausgangsbasis für eine Integration von Ansätzen zur generativen Entwicklung, wie beispielsweise der modellgetriebenen Software-Entwicklung.

8.2.5 Resümee

Die vorangegangenen Abschnitte bestätigen, dass der in Kapitel 3.1 vorgestellte Anforderungskatalog durch die in dieser Arbeit vorgestellten Beiträge abgedeckt wurde. Die Ergebnisse dieser Arbeit

greifen den in Kapitel 3.3 motivierten Handlungsbedarf auf und lösen die in Kapitel 1.3 aufgezeigten Problemstellungen zufriedenstellend. Eine zusammenfassende Übersicht über die erfüllten Anforderungen ist in Tabelle 9 dargestellt.

Tabelle 9: Bewertung der Beiträge dieser Arbeit

Bewertungskriterien	Umsetzung in dieser Arbeit
AK1: Integrationsfähigkeit	
Unabhängigkeit von Entwicklungsprozessen	- Unabhängig definierte Entwicklungsaktivitäten (Beitrag B1) - Trennung von Zuständigkeiten durch Separation von Aufbereitung und Anwendung von Sicherheitswissen (Beitrag B1)
Anpassungsfähigkeit	- Klar definierte Ein- und Ausgabeartefakte zu Entwicklungsaktivitäten (Beitrag B1) - Unterstützung unterschiedlicher Entwicklungsmodelle
AK 2: Nachvollziehbarkeit	
Trennung von Entwicklungsphasen	- Trennung und Modularisierung von Entwicklungsphasen (Beitrag B1)
Abbildung zwischen Phasen	- Verknüpfung von Sicherheitsanforderungen und abstrakten Sicherheitsmaßnahmen (Beitrag B2) - Berücksichtigung von bestehenden Sicherheitstechnologien (Beitrag B3)
Verfeinerung von Artefakten	- Kategorisierung und Verknüpfung von Entwicklungsartefakten unterschiedlicher Abstraktionsstufen mittels Variabilitätsmodell (Beitrag B3)
AK 3: Wiederverwendbarkeit	
Definition von Sicherheitskonzepten	- Definition von Domänenmodell für Sicherheitsanforderungen (Beitrag B2) - Definition von Variabilitätsmodell zur Verknüpfung von Sicherheitsmaßnahmen (Beitrag B3)
Kategorisierung von Maßnahmen	- Einteilung von Sicherheitsmaßnahmen in konzeptionell getrennte Sicherheitsbereiche (Beitrag B3)
Unterstützung bei Entscheidungen	- Einsatz des Paradigmas von Sicherheitsmustern zur Beschreibung von Vor- und Nachteilen von Sicherheitsmaßnahmen (Beitrag B3)
Berücksichtigung von Infrastrukturen	- Sichtenbasierten Referenzmodells zur Aufbereitung von Sicherheitswissen aus bestehenden Sicherheitsinfrastrukturen (Beitrag B1)
AK4: Praktikabilität	
Standardisierung	- Definition des Domänenmodells basierend auf UML-Meta-Metamodell (Beitrag B2) - Einsatz von Funktionsmodellen (Beitrag B3)
Werkzeugunterstützung	- Integration nicht explizit betrachtet
Automatisierung	- Grundlegende Voraussetzungen durch Domänenmodell und Variabilitätsmodell geschaffen

Das vorgestellte sicherheitsbasierte Entwicklungsvorgehen orientiert sich zum einen an dem industriell akzeptierten Paradigma der Software-Produktlinien zur separaten Aufbereitung von wiederverwendbaren sicherheitsbezogenen Entwicklungsartefakten und deren Einsatz in konkreten Software-Entwicklungsprozessen. Darauf aufbauend wird durch die Aufteilung in eine Aufbereitungs- und eine Anwendungsphase die Wiederverwendung und Integration von existierendem Sicherheitswissen in einen Software-Entwicklungsprozess verfolgt. Das Ziel hierbei ist, eine effiziente Entwicklung der Sicherheitsfunktionalität eines Software-Systems zu ermöglichen.

Hierzu werden zum einen klare Aktivitäten vorgegeben, mittels welchen Sicherheitsexperten Sicherheitswissen aufbereiten können und Software-Entwickler wiederverwendbare Entwicklungsartefakte in einen Software-Entwicklungsprozess einsetzen können. Da diese Aktivitäten unabhängig von konkreten Software-Entwicklungsprozessen beschrieben werden, stellt das sicherheitsbasierte Entwicklungsvorgehen ein Rahmenwerk bereit, welches an unterschiedliche Prozessmodelle angepasst werden kann. Eine solche Anpassung wird durch die explizite Trennung und Modularisierung der Entwicklungsphasen unterstützt, wodurch sich die Aktivitäten von einzelnen Phasen austauschen lassen. Ebenso sind die Aktivitäten durch klare Ein- und Ausgabeartefakte sowie Vor- und Nachbedingungen gekennzeichnet, so dass die einzelnen Aktivitäten einer Phase in unterschiedlicher Reihenfolge durchgeführt werden können.

Die Verknüpfung zwischen den Aktivitäten innerhalb einer Entwicklungsphase sowie zwischen den Phasen wird durch die wiederverwendbaren Entwicklungsartefakte geschaffen. Das Referenzmodell für Sicherheitsarchitekturen bietet dabei eine zentrale Ablage für die Entwicklungsartefakte der Aufbereitung, welche das Sicherheitswissen von existierenden Sicherheitsinfrastrukturen wiedergibt. Somit wird eine Verbindung zwischen abstrakten Sicherheitsmodellen und bestehenden Sicherheitsprodukten, -rahmenwerken und -standards geschaffen, welche eine effiziente Entwicklung von Sicherheitsfunktionalität ermöglicht und die Integration von Software-Systemen in eine Sicherheitsinfrastruktur vereinfacht.

Die Vorlagen für Sicherheitsanforderungen stellen dabei grundlegende Mittel zur Spezifizierung von Sicherheitsanforderungen bereit. Die Bestandteile der Vorlagen werden durch ein zugrunde liegendes Domänenmodell eindeutig definiert, wodurch eine Kommunikationsgrundlage zwischen Sicherheitsexperten und Software-Entwicklern geschaffen wird. Die Modularisierung des Domänenmodells ermöglicht des Weiteren die Anpassung und Erweiterung der Vorlagen an spezifische Bedürfnisse eines Entwicklungsprozesses. Eine Verknüpfung der Sicherheitsanforderungen zu relevanten Sicherheitsmaßnahmen stellt den Übergang zu der Entwurfsphase bereit. Die Vorlagen beinhalten Kombinationen von Sicherheitsanforderungen und zugehörigen Konzepten, welche durch praktische Erfahrung als bewährt angesehen sind und somit in unterschiedlichen Software-Systemen angewendet werden können. Durch den Einsatz der Vorlagen lässt sich daher eine Effizienzsteigerung bei der Analyse von Sicherheitsanforderungen erreichen.

Für den Entwurf von Sicherheitsmaßnahmen wurde auf dem Ansatz von Sicherheitsmustern aufgebaut, mit deren Hilfe bewährte Sicherheitsmodelle und -lösungen gemäß des Paradigmas von Entwurfsmuster beschrieben werden. Hierdurch wird ebenfalls eine Kommunikationsgrundlage geschaffen und die Integration von sicherheitsbasierter und fachlicher Entwicklung ermöglicht. Die Beschreibung von Abhängigkeiten durch den Einsatz von Funktionsmodellen ermöglicht des Weiteren die Auswahl von alternativen und verpflichtenden Sicherheitsmaßnahmen, um einen angemessenen Bedarf an Sicherheitsfunktionalität für ein Software-System umzusetzen. Der Musteransatz erlaubt hierbei eine Entscheidungsunterstützung bei der Selektion von alternativen Maßnahmen. Ebenfalls wird durch die Verknüpfung von Sicherheitsmustern aus unterschiedlichen Abstraktionsebenen die iterative Verfeinerung von abstrakten Sicherheitsarchitekturen zu technologischen Sicherheitslösungen

unterstützt, welche durch die im Referenzmodell hinterlegten Sicherheitsinfrastrukturen umgesetzt werden können.

Insgesamt bieten die vorgestellten Konzepte einen wesentlichen Beitrag zur Realisierung des Zieles der sicherheitsbasierten Software-Entwicklung, indem durch den Einsatz von existierendem Sicherheitswissen eine effiziente Implementierung von Sicherheitsfunktionalität für ein Software-System geschaffen wird. Die vorliegende Arbeit bietet dabei einen Rahmen, in welchem sich bestehende Ansätze sehr gut eingliedern lassen, um eine strukturierte sicherheitsbasierte Entwicklung zu ermöglichen.

8.3 Ausblick

Trotz der in der vorliegenden Arbeit vorgestellten Beiträge für die Entwicklung von sicheren Software-Systemen bleiben Fragestellungen offen, welche im Rahmen der Arbeit nicht behandelt wurden und sich aus der Anwendung der Beiträge in den Tragfähigkeitsnachweisen ergeben haben. Dies verdeutlicht die Komplexität, welche mit der Entwicklung von sicheren Software-Systemen verbunden ist. Die im Folgenden vorgestellten Fragestellungen dienen somit als mögliche Weiterführung dieser Arbeit.

Variabilität und Verfeinerung von Angriffen

In dieser Arbeit wurde der Fokus auf die Entwicklung von Sicherheitsmaßnahmen durch die iterative Verfeinerung von Sicherheitsmustern gelegt. Hierbei wurde deutlich, dass die Modellierung der Maßnahmen eine ebenso intensive Betrachtung von Angriffen und Angreifern benötigt, da diese die Motivation für die Maßnahmen darstellen. Bedrohungen und Angriffe wurden lediglich zur Erhebung und Festlegung von Sicherheitsanforderungen genutzt. Eine verfeinerte Untersuchung der möglichen Angriffe und deren Risiko, Eintrittswahrscheinlichkeit und Auswirkungen sowie genaue Vorgehensweisen stellt jedoch auch eine Unterstützung bei der Auswahl und Umsetzung von spezifischen Sicherheitsmaßnahmen dar.

Das in dieser Arbeit vorgestellte Variabilitätsmodell für Sicherheitsmuster stellt eine gute Grundlage dar, um eine solche verfeinerte Untersuchung zu ermöglichen. In Ergänzung lassen sich bereits mittels Angriffsbäumen [Sc01] Voraussetzungen und mögliche alternative Vorgehensweisen für Angriffe analysieren. Diese sind somit eine Ergänzung zu dem in dieser Arbeit vorgestellten Entwicklungsvorgehen, deren Integration zu untersuchen ist.

Automatisierte Generierung von Sicherheitsfunktionalität

Der hohe Grad an Wiederverwendung von Sicherheitsfunktionalität, welche durch entsprechende Produkte und Rahmenwerke bereitgestellt wird, erhöht die Redundanz bei der Umsetzung von sicherheitsbezogener Funktionalität. Um die Fehlerquote bei der Implementierung weiter zu reduzieren ist eine automatisierte Generierung von sicherheitbezogenen ausführbaren Programmcode empfehlenswert. Hierbei umfasst dieser beispielsweise Konfigurationseinstellungen für die Sicherheitsprodukte sowie Adapter für den fachlichen Programmcode.

In den vergangenen Jahren wurden solche generativen Ansätze vor allem durch das Paradigma der modellgetriebenen Software-Entwicklung vorangetrieben, in welchem aus objektorientierten Modellen vollständige Software-Systeme oder Teile davon semi-automatisch generiert wurden. Die Model Driven Architecture (MDA) der Object Management Group (OMG) stellt dabei eine Standardisierungsbemühung dar. Verschiedene Ansätze betrachten die Erzeugung von Sicherheitsmaßnahmen aus Modellen (engl. Model-Driven Security, [LB+02] [Jü05]) und können als Komplement zu den Ansätzen dieser Arbeit angesehen werden. Durch das Domänenmodell für Sicherheitsanforderungen wurden bereits erste Grundsteine für eine automatisierte Generierung gelegt.

Das Referenzmodell für Sicherheitsarchitekturen kann in diesem Zusammenhang als eine Zielplattform für den erzeugten Programmcode angesehen werden, für welches jedoch ein entsprechendes Metamodell zu erstellen ist.

Betrachtung weiterer Entwicklungsphasen

Weitere Entwicklungsphasen wurden in Prämisse P3 aus den Betrachtungen für das Entwicklungsvorgehen ausgeschlossen. Dennoch ist vor allem die entwickelte oder generierte Sicherheitsfunktionalität gegenüber den aufgestellten Anforderungen durch intensives Testen zu validieren und verifizieren. Hierbei ist die Simulation von Angriffen durch entsprechende Testfälle aus den Sicherheitsanforderungen abzuleiten.

Die bereitgestellten Sicherheitsanforderungsvorlagen können als Basis genutzt werden, um zugehörige Testfälle und -szenarien zu verknüpfen, welche während und nach der Implementierung der Sicherheitsfunktionalität durchgeführt werden. Es ist zu untersuchen, ob auch diese Testfälle wiederverwendbar aufbereitet und in der Anwendungsphase des Entwicklungsvorgehens eingesetzt werden können.

Ebenso wurde die Rückführung von Erfahrungen aus dem Betrieb eines durch das vorgestellte Entwicklungsvorgehen entwickelte Software-System ausgeblendet. Die Aktualität der Sicherheitsmaßnahmen ermöglicht einen ausreichenden Schutz gegenüber neuen Angriffstechniken. Daher muss die Sicherheitsarchitektur durch Erkenntnisse aus dem laufenden Betrieb weiterentwickelt werden, um die Aktualität zu gewährleisten. Hierbei ergeben sich Fragestellungen zur gegebenenfalls automatisierten Detektion und Analyse von Angriffen sowie zur Verknüpfung zu den Entwicklungsartefakten des präsentierten Entwicklungsvorgehens.

Aufbereitung weiterer Sicherheitsbereiche

In der vorliegenden Arbeit wurden die erarbeiteten Beiträge anhand der zentralen Sicherheitsbereiche Authentifizierung, Autorisierung und Zugriffskontrolle demonstriert. Durch die Untersuchung von weiteren Sicherheitsbereichen und die Einordnung der dort vorhandenen Sicherheitslösungen wird eine Weiterentwicklung der Sicherheitsanforderungsvorlagen und Mustersprachen angeregt, so dass die Stabilität der vorgestellten Beiträge erhöht wird.

In Anlehnung an die Initiative des Open Web Application Security Projects (OWASP [OWASP]) sollte, um hierbei ein möglichst allgemein anerkanntes Entwicklungsvorgehen sowie wiederverwendbare Entwicklungsartefakte zu schaffen, die Festlegung von Anforderungsvorlagen und Mustersprachen in einem gemeinschaftlichen Vorgehen erarbeitet werden. Hierbei sollten vor allem bestehende Definitionen von existierenden Sicherheitsstandards berücksichtigt werden.

Anhang

A. Abkürzungsverzeichnis

A

ABAC Attribute-based Access Control (dt. Attributebasierte Zugriffskontrolle)

ACL Access Control List (dt. Zugriffskontrollliste)

API Application Programming Interface

ASP.NET Active Server Pages .NET

B

BDSG Bundesdatenschutzgesetz

BSI Bundesamt für Sicherheit in der Informationstechnik

C

C&M Cooperation & Management

CCRA Common Criteria Recognition Arrangement

CIM Computational-Independent Model (dt. rechnerunabhängiges Modell)

D

DAC Discretionary Access Control (dt. benutzerbestimmte Zugriffskontrolle)

DBMS Datenbank-Managementsystem

DSL Domain Specific Language (dt. Domänenspezifische Sprache)

E

EU Europäische Union

G

GIS Geografisches Informationssystem

GP Generative Programming (dt. generative Programmierung)

GPS Global Positioning System

GSHB BSI IT-Grunschutzhandbuch

GSN Global Sensor Network

GUI Graphical User Interface (dt. grafische Benutzerschnittstelle)

H

HMS Horizontale Mustersprache

HTTP Hyper Text Transfer Protocol

HPL Horizontal Pattern Language (dt. Horizontale Mustersprache)

I

IdP Identity Provider (dt. Identitätsdienst)

IoT Internet of Things

IEEE Institute of Electrical and Electronics Engineers

IETF Internet Engineering Task Force

IOSB Fraunhofer Institut für Optronik, Systemsteuerung und Bildauswertung

ISO International Organization for Standardization

IT Informationstechnologie (engl. Information Technology)

IWA Intercepting Web Agent

J

JEE Java Platform, Enterprise Edition

JSP JavaServer Page

K

KCG KITCampusGuide

KIT Karlsruher Institute für Technologie

L

LDAP Lightweight Directory Access Protocol

M

MAC Mandatory Access Control (dt. Systembestimmte Zugriffskontrolle)

MBAC Metadata-based Access Control (dt. Metadatenbasierte Zugriffskontrolle)

MDA Model-Driven Architecture (dt. modellgetriebene Architektur)

MDSD Model-Driven Software Development

MLS Multi-level Security
(dt. Mehrschichtige Autorisiserung)
MOF Meta Object Facility

N

NFC Near Field Communication
(dt. Nahfeldkommunikation)

O

OASIS Organization for the Advancement of Structured Information Standards
OMG Object Management Group
OSM OpenStreetMap
OWASP Open Web Security Project

P

PaaS Platform as a Service
PAP Policy Administration Point
(dt. Verwaltungspunkt)
PEP Policy Enforcement Poin
(dt. Durchsetzungspunkt)
PDP Policy Decision Point
(dt. Entscheidungspunkt)
PIM Platform-Independent Model
(dt. plattformunabhängiges Modell)
PIP Policy Information Point
(dt. Informationspunkt)
PKI Public Key Infrastructure
PM Platform Model
(dt. Plattformmodell)
PPS Permission Policy Set
POI Point of Interest
PSI Platform-Specific Implementation
(dt. plattformspezifische Implementierung)
PSM Platform-Specific Model
(dt. plattformspezifisches Modell)

R

RBAC Role-based Access Control
(dt. Rollenbasierte Zugriffskontrolle)
RB-RBAC Rule-based RBAC
(dt. Regelbasierte RBAC)
RDF Resource Description Framework
RPS Role Policy Set
RUP Rational Unified Process

S

SaaS Software as a Service
SAML Security Assertion Markup Language
SCC Steinbuch Centre for Computing
SOA Service-Oriented Architecture
(dt. dienstorientierte Architektur)
SPL Software-Produktlinie
SQL Structured Query Language
SSA Secure Service Agent
SSL Secure Sockets Layer
SSO Single Sign-On

T

TLS Transport Layer Security

U

UIS Umweltinformationssystem
UML Unified Modeling Language
UP Unified Process
USB Universal Serial Bus

V

VMS Vertikale Mustersprache
VPL Vertical Pattern Language
(dt. Vertikale Mustersprache)

W

W3C World Wide Web Consortium

X

XACML Extensible Access Control Markup Language
XML Extensible Markup Language

B. Abbildungsverzeichnis

C. Tabellenverzeichnis

D. Literaturverzeichnis

[Al79] C. Alexander, *The Timeless Way of Building*. New York: Oxford University Press, 1979.

[An08] R. J. Anderson, *Security Engineering*. 2nd ed. Indianapolis, Ind.: Wiley, 2008.

[AS02] M. A. Al-Kahtani und R. Sandhu, „A Model for Attribute-Based User-Role Assignment, “ *Computer Security Applications Conference*, S. 353-262, 2002.

[AXIOMATICS] Axiomatics, „Axiomatics Policy Server“, Axiomatics Inc. 2013.

[Ba06] J. Barnes, Implementing the IBM Rational Unified Process and Solutions: A Guide to Improve your Software Development Capabilities and Maturity, 25 ed. IBM Press, 2006.

[Bi77] K. J. Biba, „Integrity Considerations for Secure Computer Systems“, MITRE Corp, Bedford, MA, TR-3153, Apr. 1977.

[BC+12] L. Bass, P. Clements und R. Kazman, *Software Architecture in Practice*. Addison-Wesley Professional, 2012.

[BD10] B. Brügge und A. H. Dutoit, *Object-Oriented Software Engineering - Using UML, Patterns and Java*, 3rd ed. Boston; München : Pearson, 2010.

[BDSG] *Bundesdatenschutzgesetz (BDSG)*, aktualisiert 1990, S. 1-38.

[BF+08] F. A. Braz, E. B. Fernandez und M. VanHilst, „Eliciting Security Requirements through Misuse Activities”, 19th International Conference on Database and Expert Systems Application, 2008, S. 328-333.

[BH+07] F. Buschmann, K. Henney und D. C. Schmidt, *On Patterns and Pattern Languages*. Chichester: John Wiley & Sons Ltd, 2007.

[BING-MAPS] Microsoft, „Bing Maps“, http://maps.bing.com, 2013.

[BK+07] F. Buschmann, K. Henney, und D. C. Schmidt, *A Pattern Language for Distributed Computing*, XXXI.ed, vol. 4. Chichester: Wiley, 2007.

[BL73] D. E. Bell und L. J. LaPadula, „Secure Computer Systems: A Mathematical Model," MITRE Corp, Bedford, MA, MTR-2547, Nov. 1973.

[BM05] S. Barnum und G. McGraw, „Knowledge for Software Security," *IEEE Security & Privacy*, vol. 3, no. 2, S. 74-78, 2005.

[BM07] E. Bertino und L. Martino, „A Service-oriented Approach to Security - Concepts and Issues," *Proceedings of the 11th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS '07)*, S. 31-40, 2007.

[BN89] D. F. C. Brewer und M. J. Nash, „The Chinese Wall Security Policy," IEEE Symposium in Research in Security and Privacy, Los Alamitos, CA, 1989, S. 215-228.

[BR+99] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad und M. Stal, *A System of Patterns*, XVI. vol. 1. Chichester: John Wiley & Sons, 1999.

[BSI-GSHB] Bundesamt für Sicherheit in der Informationstechnik, „IT-Grundschutz-Katalog"

[Ca04] L. J. Camp, „Digital Identity, " *IEEE Technology and Society Magazine*, vol. 23, no. 3, S. 34-41, 2004.

[CB+10] P. Clements, F. Bachmann, L. Bass, D. Garlan, J. Ivers, L. Reed, R. Nord und J. Stafford, *Documenting Software Architecture - Views and Beyond*, 2nd ed. Upper Saddle River, NJ: Addison-Wesley Professional, 2010.

[CC09] Common Criteria Committee, „Common Criteria for Information Technology Security Evaluation", 2009.

[CE02] K. Czarnecki und U. Eisenecker, *Generative Programming*. Boston; München: Addison-Wesley, 2002.

[CN02] P. C. Clements und L. M. Northrop, *Software Product Lines - Practices and Patterns*, 1st ed. Uppere Saddle River, NJ: Addison-Wesley, 2002.

[CN+05] C. Steel, R. Nagappan und R. Lai, *Core Security Patterns*, 1st ed. Upper Saddle River, N. J.: Prentice Hall International, 2005.

[DA11] A. Dikanski und S. Abeck, „A View-based Approach for Service-Oriented Security Architecture Specification", *The Sixth International Conference on Internet and Web Applications and Services (ICIW)*, St. Maarten, The Netherland Antilles, 2011.

[DE+09] A. Dikanski, C. Emig und S. Abeck, „Integration of a Security Product in Service-oriented Architecture", *The Third International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, Athens, Greece, 2009, S. 1-7.

[DF05] N. A. Delessy und E. B. Fernandez, „Patterns for the eXtensible Access Control Markup Language", *12th Pattern Languages of Programs Conference (PLoP2005)*, S. 7-10, 2005.

[DF+07] N. A. Delessy, E. B. Fernandez und M. M. Larrondo-Petrie, „A Pattern Language for Identity Management", *International Multi-Conference on Computing in the Global Information Technology*, S. 31-31, 2007.

[DPA11] ZeitOnline, Reuters, dpa, „Systematischer Hacker-Angriff auf Regierungen und Firmen“, *zeit.de*, 03-Aug-2011. [Online]. Available: http://www.zeit.de/digital/internet/2011-08/Internet-Hackerangriff-Datenraub. [Accessed: 04-Aug-2011].

[DPA11b] dpaReuters, „Daten von Millionen Playstation-Kunden gestohlen,“ *zeit.de*, 27-Apr-2011.

[DS+12] A. Dikanski, R. Steinegger und S. Abeck, „Identification and Implementation of Authentication and Authorization Patterns in the Spring Security Framework“, *The Sixth International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*, 2012, S. 14-20.

[Ec09] C. Eckert, *IT-Sicherheit*, 6 ed. München: Oldenbourg, Wissenschaftsverlag, 2009.

[EB+07] C. Emig, F. Brandt, S. Kreuzer und S. Abeck, „Identity as a Service-Towards a Service-Oriented Identity Management Architecture“, *LNCS*, vol. 4606, S. 1-8, 2007.

[EK+08] C. Emig, S. Kreuzer, S. Abeck, J. Biermann und H. Klarl, „Model-Driven Development of Access Control Policies for Web Services,“ *The 9th IASTED International Conference Software Engineering and Applications*, vol. 632, S. 165-169, 2008.

[ES+06] C. Emig, H. Schandua und S. Abeck, „SOA-aware Authorization Control,“ *International Conference on Software Engineering Advances*, S. 62–65, 2006.

[Fe04] E. B. Fernandez, „A Methodology for Secure Software Design“, The 2004 International Conference on Software Engineering Research and Practice (SERP’04), S. 21-24, 2004.

[Fe99] E. B. Fernandez, „Coordination of Security Levels for Internet Architectures“, *The Tenth International Workshop on Database and Expert Systems Applications*, Florence, 1999, S. 837–841.

[Fi03a] D. G. Firesmith, „Engineering Security Requirements”, *Journal of Object Technology*, vol. 2, no. 1, S. 53-68, 2003.

[Fi03b] D. G. Firesmith, „Security Use Cases“, *Journal of Object Technology*, vol. 2, S. 53-64, Mai 2003.

[Fi04] D. G. Firesmith, „Specifying Reusable Security Requirements“, *Journal of Object Technology*, vol. 3, Kap. 1, S. 61-75, 2004.

[Fo04] M. Fowler, *Analysis Patterns - Reusable Object Models*, 16 ed. Addison-Wesley, 2004.

[FE09] S. Fenz und A. Ekelhart, „Formalizing Information Security Knowledge“, *The 4th International Symposium on Information, Computer, and Communications Security*, 2009, S. 183-194.

[FH06] T. E. Fægri und S. O. Hallenstein, „A Software Product Line Reference Architecture for Security“, in *Software Product Lines*, no. 8, T. Käkölä und J. C. Dueñas, Eds. Berlin, Heidelberg: Springer, 2006, S. 276-326.

[FL06] E. B. Fernandez and M. M. Larrondo-Petrie, „A Methodology to Develop Secure Systems Using Patterns,“ in *Integrating Security and Software Engineering: Ad-*

vanced and Future Visions, Kap. 5, H. Mouratidis und P. Giorgini, Eds. Hershey, London, Melbourne, Singapore: IDEA Group Publishing, 2006, S. 107-126.

[FP01] E. B. Fernandez und R. Pan, „A Pattern Language for Security Models“, *Conference on Pattern Languages of Programs*, 2001.

[FP+08] E. B. Fernandez, G. Pernul und M. M. Larrondo-Petrie, „Patterns and Pattern Diagrams for Access Control“, *TrustBus*, S. 38-47, 2008.

[FS87] A. Fiat und A. Shamir, „How to Prove Yourself: Practical Solutions to Identification and Signature Problems”, *Advances in Cryptology-Crypto'86*, 1987.

[FV+06] E. B. Fernandez, M. VanHilst, M. M. Larrondo-Petrie und S. Huang, „Defining Security Requirements Through Misuse Actions“, in *Advanced Software Engineering: Expanding the Frontiers of Software Technology*, vol. 219, Kap. 10, S. F. Ochoa und G.-C. Roman, Eds. Santiago, Chile: Springer US, 2006, pp. 123-137.

[FW+08] E. B. Fernandez, H. Washizaki, N. Yoshioka, A. Kubo und Y. Fukazawa, „Classifying Security Patterns“, *LNCS*, vol. 4976, S. 342-347, 2008.

[FY07] E. B. Fernandez und X. Yuan, „Securing Analysis Patterns“, The 45th Annual Southeast Regional Conference, 2007, S. 288-293.

[Gl13] B. Gloger, *Scrum: Produkte zuverlässig und schnell entwickeln*, 4 ed. München: Hanser Verlag, 2013.

[GH+94] E. Gamma, R. Helm, R. Johnson und J. Vlissides, *Design Patterns*. Addison-Wesley Professional, 1994.

[GOOGLE-MAPS] Google, „Google Maps.“, http://maps.google.com

[HH+07] D. Hatebur, M. Heisel und H. Schmidt, “A Pattern System for Security Requirements Engineering”, *The Second International Conference on Availability, Reliability and Security (ARES'07).*, S. 356-365, 2007.

[HH+07a] D. Hatebur, M. Heisel und H. Schmidt, „A Security Engineering Process Based on Patterns”, *18th International Workshop on Database and Expert Systems Applications*, S. 734-738, 2007.

[HL+08] C. B. Haley, R. Laney, J. D. Moffett und B. Nuseibeh, „Security Requirements Engineering: A Framework for Representation and Analysis“, *IEEE Transactions on Software Engineering*, vol. 34, Kap. 1, S. 133-153, 2008.

[HL09] M. Howard und D. LeBlanc, *Writing Secure Code*, 2nd ed. O'Reilly Media, Inc., 2009.

[HM+06] C. B. Haley, J. D. Moffett, R. Laney und B. Nuseibeh, „A Framework for Security Requirements Engineering“, The 2006 International Workshop on Software Engineering for Secure Systems, S. 35-42, 2006.

[HS75] J. H. Saltzer und M. D. Schroeder, „The Protection of Information in Computer Systems“, *Proceedings of the IEEE*, 1975, vol. 63, Kap. 9, S. 1278-1308.

[IEEE-610.12-1990] IEEE,“IEEE Standard Glossary of Software Engineering Terminology,“ New York, 610.12-1990.

[IETF-LDAP] IETF, „Lightweight Directory Access Protocol v3,“ RFC 2251.

[IETF-X.509] D. Cooper, S. Santesson, F. S, S. Boeyen, R. Housley und W. Polk, „Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profil“, IETF, 5280.

[IK+08] J. A. Ingalsbe, L. Kunimatsu, T. Baeten und N. R. Mead, „Threat Modeling: Diving into the Deep End“, *IEEE Software*, vol. 25, Kap. 1, S. 28-34, Jan. 2008.

[ISO-27001] ISO, „Information Technology – Security Techniques – Information Security Management Systems – Requirements,“ ISO 27001:2005.

[IT03] T. Imamura und M. Tatsubori, „Patterns for Securing Web Services Messaging“, *OPSLA Workshop on Web Services and Service Oriented Architecture Best Practice and Patterns*, 2003.

[Jü05] J. Jürjens, „Model-Based Security Engineering with UML“, in *Foundations of Security Analysis and Design III*, vol. 3655, Kap. 2, A. Aldini, R. Gorrieri und F. Martinelli, Eds. Berlin, Heidelberg: Springer, 2005, S. 42-77.

[KB+07] D. Krafzig, K. Banke and D. Slama, *Enterprise SOA - Service-Oriented Architecture Best Practices*, Upper Saddle River, NJ: Prentice Hall PTR, 2007.

[KJ04] M. Kircher und P. Jain, *Patterns for Resource Management*, vol. 3. Chichester: Wiley, 2004.

[KK+11] S. Kim, D.-K. Kim, L. Lu, S. Kim und S. Park, „A Feature-based Approach for Modeling Role-based Access Control Systems“, *Journal of Systems and Software*, vol. 84, Kap. 12, S. 2035-2052, Dec. 2011.

[KS+96] N. Kano, N. Seraku, F. Takahashi und S. Tsuji, „Attractive Quality and Must-be Quality“, in *The Best on Quality*, vol. 7, J. D. Hromi, Ed. Milwaukee: Quality Press, 1996.

[La11] U. Ladurner, „Der Wurm als Bombe“, *Die Zeit*, vol. 2011, Hamburg, 16-Jun-2011.

[LB+02] T. Lodderstedt, D. Basin und J. Doser, „SecureUML: A UML-Based Modeling Language for Model-Driven Security”, *LNCS*, vol. 2460, pp. 426–441, 2002.

[Me11] S. Metzger, „Angriff der legalen Hacker“, *Handelsblatt*, Köln, S. 28, 30. Juli 2011.

[Mu10] P. Mularien, *Spring Security 3*. Birmingham: Packt Publishing, 2010.

[MC04] G. McGraw, „Software Security“, *IEEE Security & Privacy*, vol. 2, no. 2, S. 80-83, 2004.

[MF+06] D. Mellado, E. Fernández-Medina, and M. Piattini, „A Common Criteria Based Security Requirements Engineering Process for the Development of Secure Information Systems,“ *Computer Standards & Interfaces*, no. 29, S. 244-253, Jun. 2006.

[MF+08] D. Mellado, E. Fernández-Medina und M. Piattini, „Security Requirements Engineering Process for Software Product Lines: A Case Study“, *The Third International Conference on Software Engineering Advances (ICSEA '08)*, 2008, S. 1-6.

[MF+08a] D. Mellado, E. Fernández-Medina und M. Piattini, „Security Requirements Variability for Software Product Lines“, *The 3rd International Conference on Availability, Reliability and Security (ARES'08)*, 2008, S. 1413-1420.

[MF+09] D. Mellado, E. Fernández-Medina und M. Piattini, „Security Requirements Management in Software Product Line Engineering“, in *e-Business and Telecommunications*, vol. 48, J. Filipe und M. S. Obaidat, Eds. Springer, 2009, S. 250-263.

[MF+10] D. Mellado, E. Fernández-Medina und M. Piattini, „Security Requirements Engineering Framework for Software Product Lines“, *Information and Software Technology*, vol. 52, Kap. 10, S. 1094-1117, Okt. 2010.

[MF99] J. McDermott und C. Fox, „Using Abuse Case Models for Security Requirements Analysis,“ *The 15th Annual Computer Security Applications Conference*, 1999, S. 55-66.

[MS-AD] Microsoft “Active Directory”

[Nu01] B. Nuseibeh, „Weaving the Software Development Process Between Requirements and Architectures“, *The First International Workshop From Software Requirements to Architectures (STRAW'01)*, Toronto, Canada, 2001, S. 5.

[NB+10] A. Nhlabatsi, A. Bandara, S. Hayashi, C. B. Haley, J. Jürjens, H. Kaiya, A. Kubo, R. Laney, H. Mouratidis, B. Nuseibeh, T. T. Tun, H. Washizaki, N. Yoshioka und Y. Yu, „Security Patterns: Comparing Modeling Approaches“; in *Software Engineering for Secure Systems: Industrial and Research Perspectives*, no. 4, H. Mouratidis, Ed. IGI Global, 2010, S. 75-111.

[NTSISSI-4011] Committee on National Security Systems (CNSS), “National Training Standard for Information Systems Security (InfoSec) Professionals,“ NSTISSI-4011, Aug. 1998.

[OASIS-SAML] S. Cantor, J. Kemp, R. Rhilpott und E. Maler, Eds., „Assertions and Protocols for the OASIS Security Assertion Markup Language“, OASIS Standard, 2011.

[OASIS-WSS-v1.1] A. Nadalin, C. Kaler, R. Monzillo und P. Hallam-Baker, Eds., „Web Services Security: SOAP Message Security 1.1 (WS-Security 2004).“, OASIS Standard, 2004.

[OASIS-XACML-v3.0] E. Rissanen, Ed., „eXtensible Access Control Markup Language (XACML) Version 3.0,“ Jan. 2013.

[OASIS-XACML-RBAC] E. Rissanen, Ed., „XACML v3.0 Core and Hierarchical Role Based Access Control (RBAC) Profile Version 1.0,“ Aug. 2010.

[OMG-MOF-v2.4] Object Management Group (OMG), „Meta Object Facility Core Specification“, OMG Standard ptc/2010-12-08.

[OSM] OpenStreetMap Foundation, „OpenStreetMap“, http://openstreetmap.org

[OWASP] OWASP, “Open Web Security Project“, https//owasp.org

[Pa72] D. L. Parnas, „On the Criteria to be Used in Decomposing Systems into Modules“, *Communications of the ACM*, vol. 15, no. 12, S. 1053-1058, 1972.

[PB+05] K. Pohl, G. Böckle und F. van der Linden, *Software Product Line Engineering - Foundations, Principles, and Techniques*, no. XXVI. Berlin, Heidelberg: Springer, 2005.

[PF+04] T. Priebe, E. B. Fernandez, J. Mehlau und G. Pernul, „A Pattern System for Access Control“, *Research directions in Data and Applications Security*, XVIII: IFIP TC11/WG11. 3, 18th annual Conference on Data and Applications Security, July 25-28, 2004, Sitges, Catalonia, Spain, S. 235, 2004.

[Ra02] J. Ramachandran, Designing Security Architecture Solutions. Wiley, 2002.

[RH+06] R. Reussner und W. Hasselbring, Eds., *Handbuch der Software-Architektur*, 1st ed. Heidelberg: dpunkt-Verlag, 2006.

[Sa92] R. S. Sandhu, „A Lattice Interpretation of the Chinese Wall Policy“, *The 15th NIST-NSCS National Computer Security Conference*, Washington, 1992, S. 329-339.

[Sa93] R. S. Sandhu, „Lattice-Based Access Control Models“, *IEEE Computer*, vol. 26, Kap. 11, S. 9-19, 1993.

[Sc01] B. Schneier, *Secret & Lies*, 1st ed. Weinheim: dpunkt.verlag, 2001.

[Sc03] M. Schumacher, Security Engineering with Patterns: Origins, Theoretical Model, and New Applications. Heidelberg: Springer, 2003.

[Sc10] B. Schneier, „The Story Behind The Stuxnet Virus“, *forbes.com*, 10-Jul-2010. [Online]. Available: http://www.forbes.com/2010/10/06/iran-nuclear-computer-technology-security-stuxnet-worm.html. [Accessed: 01-Aug-2011].

[So07] I. Sommerville, *Software Engineering*, 8 ed. Harlow, Munich: Addison-Wesley, 2007.

[SC+96] R. Sandhu, E. Coyne, H. Feinstein und C. Youman, „Role-based Access Control Models“, *Computer*, vol. 29, no. 2, S. 38-47, 1996.

[SF+03] G. Sindre, D. G. Firesmith und A. L. Opdahl, „A Reuse-Based Approach to Determining Security Requirements“, *The 9th International Workshop on Requirements Engineering: Foundation for Software Quality*, 2003.

[SF+05] M. Schumacher, E. B. Fernandez, D. Hybertson, F. Buschmann und P. Sommerlad, *Security Patterns*. Chichester, England: John Wiley & Sons Ltd, 2005.

[SHIBBOLETH] Shibboleth Consortium. „Shibboleth“, http://shibboleth.org

[SK13] J. Sutherland und K. Schwaber, „The Scrum Guide“, *scrum.org*, Jul-2013. [Online]. Available: https://www.scrum.org/Scrum-Guides. [Accessed: Sep-2013].

[SO05] G. Sindre and A. L. Opdahl, „Eliciting Security Requirements with Misuse Cases“, *Requirements Engineering*, vol. 10, no. 1, S. 34-44, 2005.

[SPRING-SEC] SpringSource Community, “Spring Security“, Apache License, Apr-2008.

[SS94] R. Sandhu und P. Samarati, „Access Control: Principle and Practice“, *IEEE Communications Magazine*, vol. 32, no. 9, S. 40-48, 1994.

[SS+00] D. C. Schmidt, M. Stal, H. Rohnert und F. Buschmann, *Patterns for Concurrent and Networked Objects*, vol. 2. John Wiley & Sons, 2000.

[SV+07] T. Stahl, M. Völter, S. Efftinge und A. Haase, *Modellgetriebene Software-Entwicklung: Techniken, Engineering, Management*, 2nd ed. Heidelberg: Dpunkt-Verlag, 2007.

[TJ+08] I. A. Tøndel, M. G. Jaatun und P. H. Meland, „Security Requirements for the Rest of Us: A Survey“, *IEEE Software*, vol. 25, no. 1, S. 20-27, 2008.

[VG02] J. Viega und G. McGraw, *Building Secure Software*, 1st ed. Boston, Mass; München: Addision-Wesley, 2002.

[VM04] D. Verdon und G. McGraw, „Risk Analysis in Software Design“, *IEEE Security & Privacy*, vol. 2, no. 4, S. 79-84, 2004.

[Wi05] P. J. Windley, *Digital Identity - Unmasking Identity Management Architecture*, 1st ed. Beijing; Köln: O'Reilly, 2005.

[Wy07] C. Wysopal, *The Art of Software Security Testing - Identifying Software Security Flaws*, 23rd ed. Upper Saddle River, NJ; München: Addison-Wesley, 2007.

[WM05] M. E. Whitman und H. J. Mattord, *Principles of Information Security*, 2nd ed. Boston: Thomson Course Technology, 2005.

[YAHOO-MAPS] Yahoo, „Yahoo Maps.” http://maps.yahoo.com.

[YT+05] E. Yuan, J. Tong, B. Inc und V. McLean, „Attributed Based Access Control (ABAC) for Web Services“, *IEEE International Conference on Web Services (ICWS 2005)*, 2005.

E. Index

Z