universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Spezifikation von Sicherheitsanforderungen durch ViewPoint Integration“

verfasst von / submitted by

Fata Kabiljagić, Bsc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Diplom-Ingenieurin (Dipl.-Ing.)

Wien, 2018 / Vienna 2018

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 926
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Wirtschaftsinformatik
Betreut von / Supervisor:	Univ.-Prof. Dipl.-Ing. DDr. Gerald Quirchmayr

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Wien, September 2018

Unterschrift

_____

ABSTRACT (DEUTSCH)

Die Entwicklung von großen Systemen erfordert kooperative Arbeitsprozesse, die sehr komplex sind. Diese Prozesse sind mit vielen Menschen, die unterschiedlichen Perspektiven und Vorstellungen haben, verbunden und beinhalten viele Aktivitäten und Produkte, als Ergebnis dieser Aktivitäten. Die kooperativen Arbeitsprozesse können durch die Entwicklung von Modellen, die diese Prozesse abbilden können und durch Software-Tools unterstützt werden. Die Anforderungsanalyse ist ein Teil jedes Entwicklungsprozesses. Da die heutigen Systeme viele Schwachstellen haben können und vielen Bedrohungen ausgesetzt sind, sollte bei einer Anforderungsanalyse nicht nur die Funktionalität, sondern auch die Sicherheit berücksichtigt werden.

Diese Arbeit beschäftigt sich im theoretischen Teil mit der Analyse der Sicherheitsanforderungen (Schutzzielen, Sicherheitsmechanismen, einigen Security Requirements Methoden…), der ViewPoint-Integration (einigen Modellen, die die verschiedenen Sichten der Beteiligten spezifizieren, analysieren und verwalten) und mit den kooperativen Technologien. Weiters wird im Rahmen dieser Arbeit eine Modellierungsmethode, die den Prozess der Spezifikation der Sicherheitsanforderungen unterstützen sollte, entwickelt und beschrieben. Die Modellierungsmethode basiert auf dem Metamodell für kooperative Prozesse von Selmin Nurcan und Collete Rolland.

Anschließend wird eine Fallstudie vorgestellt, die zur Evaluierung dieser Modellierungsmethode dient.

ABSTRACT (ENGLISH)

The development of large systems requires cooperative work processes that are very complex. These processes are associated with many people who have different perspectives and perceptions and include many activities and products. The collaborative work processes can be supported by the development of models that can map these processes and by software tools. Requirements analysis is part of every development process. Because today's systems can have many vulnerabilities and are exposed to many threats, a requirements analysis should consider not only functionality but also security.

In the theoretical part this thesis deals with the security requirements analysis (protection goals, security mechanisms, some security requirements methods ...), the ViewPoint integration (some models that specify, analyze and manage the different views of the actors) and the cooperative technologies.

A further part of this thesis is the development and description of a modelling method that should support the process of specifying security requirements. The modelling method is based on the meta-model for cooperative processes by Selmin Nurcan and Collete Rolland.

Finally, a case study will be presented to evaluate this modelling method.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

TABELLENVERZEICHNIS

1 EINFÜHRUNG

Die Entwicklung von großen und komplexen Systemen ist immer mit vielen Menschen verbunden. Jeder Mensch hat eine eigene Sicht, die durch seine Fähigkeiten, Verantwortlichkeiten, Wissen und Expertise definiert ist. Bei einer großen Anzahl der Beteiligten, besonders wenn das System eine Vielzahl von unterschiedlichen Technologien einsetzt, ist es unvermeidlich, dass sich die Sichten überschneiden. Diese Schnittpunkte sind aber nicht offensichtlich, da das Wissen in jeder Sicht anders dargestellt wird. Deswegen wird eine Koordination benötigt. Ein weiteres Problem entsteht, wenn die Entwicklung von den Beteiligten durchgeführt wird. Da kann es passieren, dass in verschieden Stadien der Ausarbeitung verschiedene Sichten zum Vorschein kommen und jede kann Gegenstand einer jeweils anderen Entwicklungsstrategie sein. Das Problem, wie die Entwicklung in einem solchen Umfeld (mit vielen Akteuren, verschiedenen Repräsentationssystemen, unterschiedlichen Entwicklungsstrategien und unterschiedlichem Domänenwissen) gesteuert und organisiert werden kann, nennt man "the multiple perspective problem".[1]

Die Entwicklung von solchen Systemen erfordert kooperative Arbeitsprozesse. Die Menschen können in der kooperativen Gruppenkommunikation und Arbeitsprozessen durch Softwaretools unterstützt werden. Die Möglichkeiten und Auswirkungen von technologischer Unterstützung auf kooperativen Arbeitsprozesse werden im Forschungsgebiet Computer Supported Cooperative Work (CSCW) untersucht.[2]

Die Unterstützung der Umsetzung der kooperativen Arbeitsprozesse erfordert unter anderem die Entwicklung von Modellen, die kooperative

[1] Anthony Finkelstein u. a., „Requirements Engineering Through Viewpoints“, 1993.
[2] Selmin Nurcan und Colette Rolland, „Meta-modelling for cooperative processes“, 1997.

Arbeitsprozesse abbilden können. Selmin Nurcan und Colette Rolland schlagen ein Prozess-Metamodell vor, das sowohl klar definierte als auch ungeordnete Arbeitsprozesse und deren Interaktionen behandeln kann. So kann eine Vielzahl von kooperativen Arbeitsprozessen abgebildet werden.[3]

Die Anforderungsanalyse ist ein Teil jedes Entwicklungsprozesses. Neben der Funktionalität soll in einer Anforderungsanalyse auch die Sicherheit berücksichtigt werden. Sicherheitsanforderungen versuchen, die Eigenschaften zu beschreiben, die Software von Bedrohungen schützen.[4]

Die Spezifikation von Sicherheitsanforderungen ist auch ein kooperativer Prozess in dem viele Beteiligte, die unterschiedliche Sichten und Vorstellungen haben, involviert sind.

Ziel der Arbeit „Spezifikation von Sicherheitsanforderungen durch ViewPoint Integration“ ist auf Basis des Metamodells von Rolland und Nurcan eine Modellierungsmethode zu entwickeln die den Prozess der Spezifikation von Sicherheitsanforderungen unterstützen sollte.

Die Arbeit ist wie folgt aufgebaut:

Kapitel 2, „Security Requirements Analyse“, behandelt Prozesse in der Softwareentwicklung und in der agilen Softwareentwicklung. Dabei werden die grundlegenden Aktivitäten der Softwareprozessmodelle wie das Wasserfallmodell, das V-Modell, usw., und die Auswirkung dieser Modelle auf Kundenanforderungen diskutiert. Weiters wird hier Requirements Engineering allgemein betrachtet: wichtige Begriffe, Aktivitäten im Requirements Engineering, die Klassifikationen der Anforderungen. Anschließend wird hier auf Security Requirements Engineering eingegangen: Schutzziele, Sicherheitsmechanismen, Integration des Security Requirements Engineering in den gesamten Softwarelebenszyklus, die Security Requirements Methoden MSRA und SQUARE und einige Modelle für die sichere Software (Common Criteria, ISO 27002, ISO IEC 21827…), Security Building Blocks, sowie agile Security Requirements Engineering und Frameworks für Risiko- und Security Management in der agilen Softwareentwicklung.

[3] Nurcan und Rolland.

[4] Paulus Sachar, *Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering* (Heidelberg: dpunkt.verlag GmbH, 2011)., Seite 72

Kapitel 3, „ViewPoint Integration“, beschreibt einige Modelle (TARA, ViewPoint, FOREST), die in den neunziger Jahren am Imperial College in London entwickelt wurden um „multiple perspective problem“ zu lösen und die Sichten der Beteiligten zu spezifizieren, zu analysieren und zu verwalten.

Kapitel 4, „Kooperative Technologien“, gibt einen Überblick über kooperative Technologien: Begriffe, computerunterstütztes kooperatives Arbeiten und Klassifikation von Groupware.

Kapitel 5, „Entwicklung des ViewPoint Integration-basierten Modells konzentriert sich auf die Phasen der Entwicklung der Modellierungsmethode. Für die Entwicklung wird der vom OMILab empfohlenes OMI Lifecycle verwendet und die Phasen Creation, Design und Development näher betrachtet: Ziele und Anforderungen der zu entwickelnden Methode werden behandelt; das Metamodell wird präsentiert, die einzelnen Klassen werden diskutiert und graphisch dargestellt; die Modellierungsplattform ADOxx wird kurz vorgestellt und der Prozess der Entwicklung der Modellierungsmethode beschrieben.

Kapitel 6, „Fallstudie: Spezifikation der Sicherheitsanforderungen und die Erstellung einer Sicherheitsstrategie“ benutzt eine Fallstudie um zu zeigen wie man die entwickelte Modellierungsmethode nutzen könnte um die Sicherheitsanforderungen zu spezifizieren. Hierbei handelt es sich um die Implementierung eines SCADA-Systems in einem fiktiven Unternehmen. Dabei wird eine Strategie zur Erstellung der Cybersicherheit ausgearbeitet und der Prozess der Anforderungsspezifikation beschrieben.

Kapitel 7, „Evaluation und Diskussion der Ergebnisse“ diskutiert die Ergebnisse der Fallstudie.

Kapitel 8, „Zusammenfassung“ ist eine Zusammenfassung der vorliegenden Arbeit.

2 SECURITY REQUIREMENTS ANALYSE

Die stetig wachsende Nutzung des Internets in den neunziger Jahren führte dazu, dass immer mehr Systeme auf das Internet zugriffen und verschiedenen Bedrohungen ausgesetzt waren. Die Softwareentwickler standen vor einer neuen Herausforderung, sichere und verlässliche Systeme zu entwickeln und zu implementieren.
Heute werden alle Systeme so entworfen, dass sie die externen Angriffe abwehren können, und nach den Angriffen die Wiederherstellung erlauben. Deshalb wird Security-Engineering, das sich mit der Entwicklung von sicheren Systemen beschäftigt, immer wichtiger.
Eine der wichtigsten Aufgaben in jedem Entwicklungsprozess ist die Anforderungsanalyse und die Anforderungsspezifikation. Bei der Entwicklung der meisten großen Systemen werden die Anforderungen in einer frühen Phase vor der Implementierung spezifiziert. Selten wird der agile Ansatz, bei dem die Ermittlung der Anforderungen und die Implementierung gleichzeitig stattfinden, für die Entwicklung von großen Systemen genutzt[5].

2.1 PROZESSE IN DER SOFTWAREENTWICKLUNG

Um ein Softwareprodukt zu entwerfen muss eine Menge von zusammengehörigen Aktivitäten durchgeführt werden. Diese Menge von Aktivitäten nennt man ein Softwareprozess. Es gibt viele unterschiedliche Softwareprozesse, die alle folgende vier für das Software-Engineering grundlegende Aktivitäten enthalten müssen[6]:

[5] Ian Sommerville, *Software Engineering*, 9. Aufl. (Pearson Deutschland, 2012)., Seite 416, 116
[6] Sommerville., Seite 54

- Softwarespezifikation – hier wird die Funktion der Software und die Beschränkungen ihrer Benutzung definiert
- Softwareentwurf und Softwareimplementierung – die Software die diese Spezifikation erfüllt wird erstellt und implementiert
- Softwarevalidierung – zur Sicherstellung, dass die Kundenwünsche erfüllt sind, wird die Software validiert
- Softwareevolution – um mit den sich verändernden Bedürfnissen des Kunden Schritt zu halten, muss sich die Software weiterentwickeln

Softwareprozesse sind sehr komplex und beinhalten außer Aktivitäten auch Produkte als Ergebnis einer Prozessaktivität, Rollen der Personen die an den Aktivitäten beteiligt sind, sowie Vor- und Nachbedingungen, also die Aussagen die wahr sind bevor und nachdem eine Aktivität durchgeführt wurde.[7]

Um die Prozesse vereinfacht darzustellen verwendet man Vorgehensmodelle. Jedes Vorgehensmodell liefert nur einen Teil der Information über den Prozess (z.B. ein Prozessaktivitätsmodell zeigt die Abfolge von Aktivitäten und nicht die Rollen der Beteiligten an diesen Aktivitäten).[8]

Im Folgenden werden einige Vorgehensmodelle und deren Auswirkungen auf Kundenanforderungen beschrieben.

Diese Vorgehensmodelle werden häufig zusammen verwendet, besonders bei der Entwicklung komplexer Systeme. Sie schließen sich also gegenseitig nicht aus.[9]

2.1.1 Das Wasserfallmodell

Das Wasserfallmodell (Abbildung 1.) ist ein Plangesteuertes Prozess, wo alle Prozessaktivitäten zeitlich und inhaltlich geplant werden müssen, bevor man anfangen kann an ihnen zu arbeiten.[10]

[7] Sommerville., Seite 55
[8] Sommerville., Seite 56
[9] Sommerville., Seite 56
[10] Sommerville., Seite 57

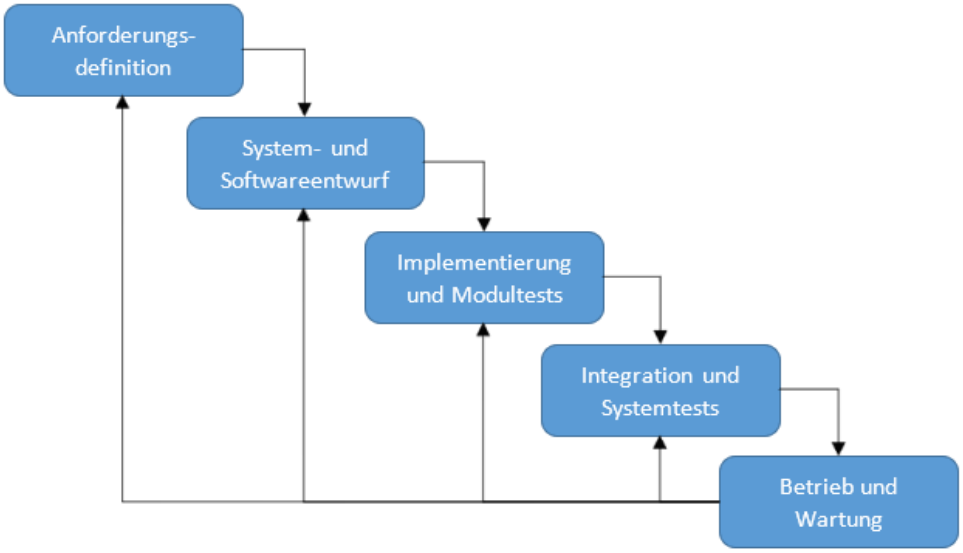


Abbildung 1. Das Wasserfallmodell

Die Phasen des Wasserfallmodells sind[11]:

- **Die Anforderungsdefinition**: Die Anforderungen werden in Zusammenarbeit mit den Systembenutzern analysiert und definiert.
- **System- und Softwareentwurf**: Systemarchitektur wird festgelegt.
- **Implementierung und Modultests**: Die Software wird durch eine Menge von Programmen oder Programmeinheiten entworfen, Module werden getestet.
- **Integration und Systemtests**: Die Programme oder Programmeinheiten werden integriert und als Ganzes getestet. Hier soll sichergestellt werden, dass die Softwareanforderungen erfüllt werden. Nach dieser Phase wird das Softwaresystem an den Kunden geliefert.
- **Betrieb und Wartung**: Diese Phase ist meistens die längste Phase innerhalb des Lebenszyklus. Hier wird das System installiert und zum Gebrauch freigegeben. Weiters wird das System gewartet, d.h. Fehler, die in früheren Phasen nicht entdeckt wurden werden korrigiert, falls neue Anforderungen aufgedeckt werden, werden Systemeinheiten und das System verbessert.

Das Problem beim Wasserfallmodell ist die starre Aufteilung des Projekts in verschiedene Phasen. So ist es schwierig auf die neuen Anforderungen des

[11] Sommerville., Seite 57

Kunden zu reagieren. Entdeckt man z.B. in der letzten Phase des Projekts Fehler oder Lücken in den ursprünglichen Anforderungen, kann es dazu führen, dass einige oder alle der vorherigen Prozessphasen wiederholt werden müssen. Das Wasserfallmodell sollte nur dort verwendet werden, wo es unwahrscheinlich ist, dass es während des Entwicklungsprozesses zu gravierenden Änderungen der Anforderungen kommt und wo diese gut durchdacht sind.[12]

2.1.2 Das V-Modell

Das V-Modell (Abbildung 2.) wurde auf Basis des Wasserfallmodells entwickelt und ist ähnlich wie Wasserfallmodell sequentiell aufgebaut. In diesem Modell wurden zum ersten Mal die Teststufen eingebaut, so dass für jede konstruktive Phase auf der linken Seite, eine korrelierende Testphase auf der rechten Seite existiert. Die Testphasen sind in Validierung (Wurde das passende Produkt entwickelt?) und Verifikation (Wurde das Produkt korrekt entwickelt?) aufgeteilt.[13]

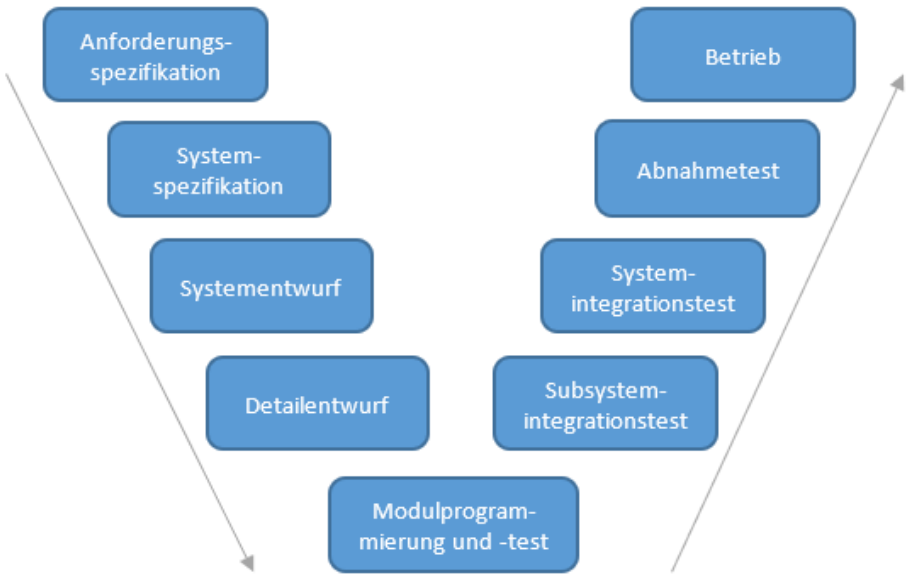


Abbildung 2. Das V-Modell[14]

[12] Sommerville., Seite 58
[13] Thomas Grechenig u. a., *Softwaretechnik mit Fallbeispielen aus realen Entwicklungsprojekten* (Pearson Studium, 2010)., Seite 376
[14] Sommerville, *Software Engineering*.

Wie beim Wasserfallmodell wird auch beim V-Modell versucht alle Anforderungen bereits im Vorfeld vollständig zu definieren. Das passiert in der ersten Phase der Systementwicklung bevor die ersten Entwurfsentscheidungen, oder Realisierungsentscheidungen getroffen werden.[15]

Das V-Modell wurde im Jahr 2005 zum V-Modell XT aktualisiert. Die Neuerungen sind explizite Darstellung der Schnittstelle zwischen Auftragnehmern und Auftraggebern und umfassendere Möglichkeiten der Anpassung (tailoring) durch ein System von Vorgehensbausteinen.[16]

2.1.3 Das Spiralmodell

Bei dem Spiralmodell (Abbildung 3.) wird die Softwareentwicklung in vier Phasen durchgeführt. Diese Phasen durchlaufen zyklisch, wobei jeder Zyklus zu einer immer weiteren Detaillierung führt.[17]

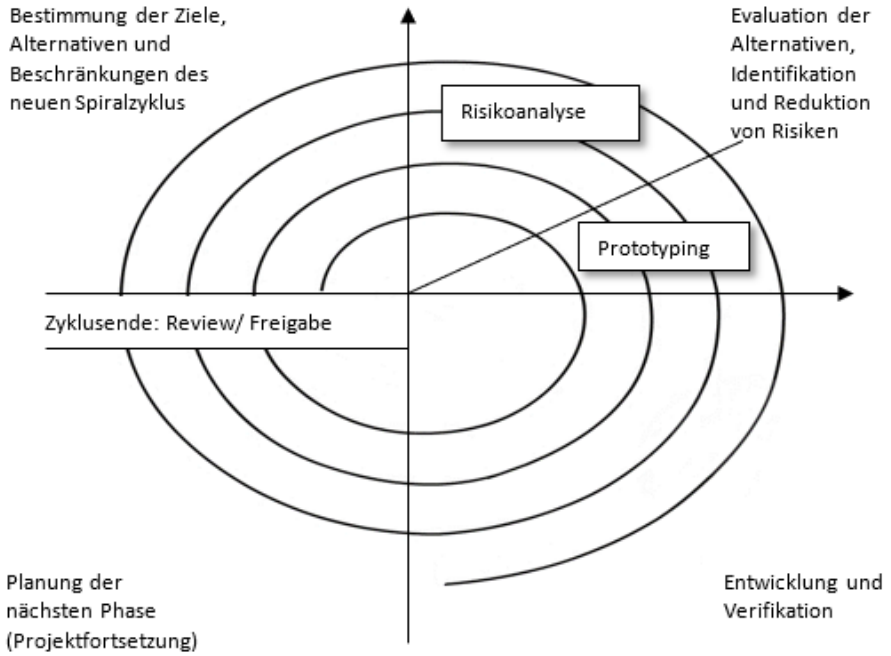


Abbildung 3. Das Spiralmodell

[15] Klaus Pohl und Chris Rupp, *Basiswissen Requirements Engineering*, 4. Aufl. (dpunkt.verlag GmbH Heidelberg, 2015)., Seite 5

[16] Manfred Broy und Andreas Rausch, „Das neue V - Modell® XT – Ein anpassbares Vorgehensmodell für Software und System Engineering“, 2005, https://doi.org/10.1007/s00287-005-0488-z.

[17] Birgit Vogel-Heuser, *Systems Software Engineering* (München: Oldenburg Industrieverlag, 2003)., Seite 28

In der ersten Phase werden die Ziele, Alternativen und Randbedingungen bestimmt. Die zweite Phase dient zur Risikoanalyse (es wird versucht die Risiken zu reduzieren) und zur Evaluation der Alternativen. Die dritte Phase umfasst die Entwicklung und die Verifikation des Produktes. In der vierten Phase wird die nächste Phase geplant, wobei zwischen der vierten Phase und der ersten Phase des nächsten Zyklus bewusst entschieden werden muss, ob das Projekt weitergeführt werden soll. Bevor die nächste Phase eingeleitet wird, werden hier alle bekannten Fakten (besonders die Wirtschaftlichen) geprüft.[18]

Da das risikogesteuerte Vorgehen des Spiralmodells die Komplexität eines Systems berücksichtigt, ist dieses Modell für sehr große und komplexe Projekte geeignet. Der Nachteil des Spiralmodells ist, dass die Erstellung von Zeit- und Kostenplanen am Anfang des Projekts schwer möglich ist, da sich die Anzahl der notwendigen Durchläufe erst im Projektverlauf ergibt.[19]

Im Gegensatz zu Wasserfall- und V-Modell müssen bei dem Spiralmodell die Anforderungen am Anfang nicht bis ins Detail geklärt werden. Diese dürfen sich im Laufe des Projekts mit jeder Iteration ändern und können auch ergänzt werden.[20]

2.1.4 Inkrementelle Entwicklung

Bei der inkrementellen Entwicklung (Abbildung 4.) wird eine Anfangsimplementierung entwickelt. Danach werden die Benutzer aufgefordert diese zu kommentieren und auf die Verbesserungsmöglichkeiten hinzuweisen. Die Implementierung wird dann über mehrere Versionen hinweg verbessert, bis ein angemessenes System entstanden ist.[21]

---

[18] Vogel-Heuser., Seite 28f

[19] Grechenig u. a., *Softwaretechnik mit Fallbeispielen aus realen Entwicklungsprojekten*., Seite 377

[20] Eckhart Hanser, *Agile Prozesse: Von XP über Scrum bis MAP* (Springer Verlag Berlin, 2010)., Seite 5

[21] Sommerville, *Software Engineering*., Seite 59

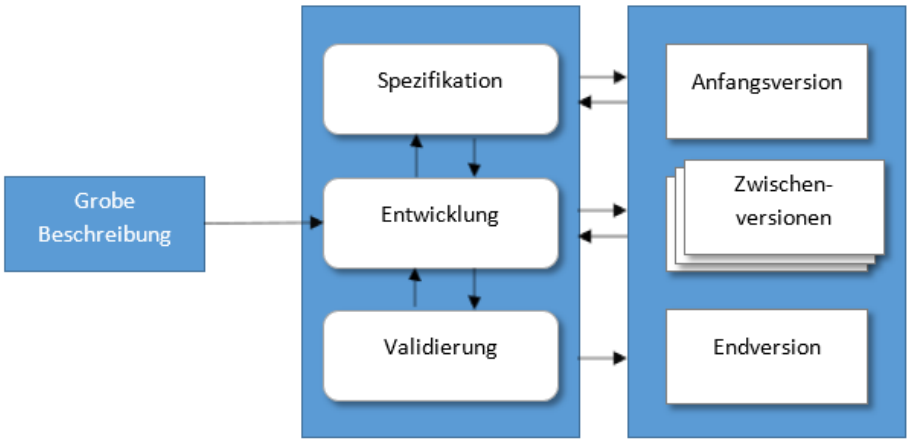


Abbildung 4. Inkrementelle Entwicklung

Die Spezifikation, Entwicklung und Validierung werden gleichzeitig ausgeführt und die Rückmeldungen werden untereinander ausgetauscht.[22]

Inkrementelle Softwareentwicklung ist für die meisten Geschäftsfälle, E-Commerce und individuellen Systeme besser geeignet als Wasserfallansatz; sie ist ein wesentlicher Teil des agilen Ansatzes, der später noch genauer erläutert wird.

Inkrementelle Softwareentwicklung hat gegenüber Wasserfallmodell sowohl Vorteile als auch Nachteile.

Die wichtigsten Vorteile sind: die Kosten für die Anpassung der an sich ändernden Kundenanforderungen sind geringer, die Rückmeldungen der Kunden zu bereits fertiggestellten Teilen der Entwicklungsarbeit ist leichter zu bekommen und die Software kann schneller ausgeliefert und installiert werden, auch wenn die gesamte Funktionalität noch nicht enthalten ist.

Die Nachteile sind: der Prozess ist nicht sichtbar und der Fortschritt kann nur an Zwischenversionen gemessen werden, die Systemstruktur wird schwächer, wenn neue Inkremente hinzugefügt werden. Das wird besonders bei großen komplexen Systemen, wenn verschiedene Teams verschiedene Teile des Systems entwickeln, zum Problem, weil große Systeme einen stabilen Rahmen oder Architektur benötigen und, weil die Verantwortlichkeiten von Teams die an Teilen des Systems arbeiten, bezüglich dieser Architektur klar festgelegt werden müssen.[23]

[22] Sommerville., Seite 59
[23] Sommerville., Seite 60f

2.1.5 Wiederverwendungsorientiertes Software-Engineering

Es gibt eine große Anzahl an wiederverwendbaren Komponenten, die in einem Großteil aller Softwareprojekte, unabhängig davon welcher Entwicklungsprozess eingesetzt wurde, verwendet werden. Wenn Mitarbeiter eines Projekts von einem Entwurf oder von Code wissen, der in dem Projekt verwendet werden könnte, wird dieser herausgesucht, nach ihren Bedürfnissen verändert und in das System eingebaut. Manchmal werden eigenständige käufliche Systeme (COTS-Systeme, Commercial Off-The-Shelf System) benutzt um eine spezielle Funktion beizusteuern (z.B. Tabellenkalkulation oder Textverarbeitung).[24]

Die Abbildung 5. zeigt ein allgemeines Prozessmodell für Wiederverwendungsorientierte Entwicklung.[25]

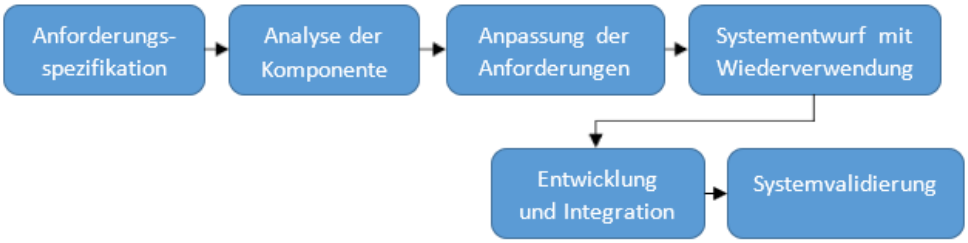


Abbildung 5. Wiederverwendungsorientiertes Software-Engineering

Nach der Anforderungsspezifikation werden die Komponenten, mit denen diese Spezifikation implementiert werden könnte, gesucht. Danach werden die Anforderungen im Hinblick an die gefundenen Komponenten analysiert und an die verfügbaren Komponenten angepasst. Falls die Veränderungen nicht möglich sind, muss nach alternativen Lösungen gesucht werden. Wenn keine Komponenten vorhanden sind, muss an dieser Stelle eine neue Software entworfen werden. Wenn passende Komponente gefunden wurde, wird der Rahmen für das System so angelegt, dass er dazu passt (er wird entweder entworfen, oder ein vorhandener Rahmen wird wiederverwendet). In der letzten Phase wird die Software, die nicht

[24] Sommerville., Seite 61
[25] Sommerville., Seite 62

beschafft werden kann, entwickelt und zusammen mit den COTS-Systemen und den anderen Komponenten integriert.[26]

Die Vorteile des Wiederverwendungsorientierten Software-Engineering sind, dass sich die Menge der zu entwickelnden Software verringert und das die Software schneller geliefert wird, somit verringern sich auch die Kosten. Die Nachteile sind, dass man bei den Anforderungen Kompromisse machen muss, und das wiederum kann dazu führen, dass die wirklichen Bedürfnisse des Benutzers nicht erfüllt werden.[27]

2.2 AGILE SOFTWAREENTWICKLUNG

Im Gegensatz zu den in dem Kapitel 2.1 beschriebenen traditionellen Modellen in der Softwareentwicklung, die von einer Reihe von vorgegebenen Prozessen und fortlaufender Dokumentation während des Entwicklungsprozesses abhängig sind, basiert die agile Softwareentwicklung auf der Idee der inkrementellen und iterativen Entwicklung. In der agilen Entwicklung wird anstelle eines einzelnen großen Prozessmodells, der Entwicklungslebenszyklus in kleinere Teile unterteilt, die als "Inkremente" oder "Iterationen" bezeichnet werden. Jeder dieser Inkremente beinhaltet jede der Phasen der Softwareentwicklung.[28]

Die agilen Methoden sind am besten für solche Systeme geeignet, bei denen sich die Anforderungen an das zu entwickelnde System im Laufe des Entwicklungsprozesses ändern. So ist eine stätige Zusammenarbeit mit dem Kunden notwendig. Die Dokumentation wird minimiert indem eher informelle Kommunikation eingesetzt wird (keine formellen Sitzungen mit schriftlichen Unterlagen).[29]
Die wichtigsten Werte der agilen Softwareentwicklung laut Agile Manifesto[30] sind[31]:

---

[26] Sommerville., Seite 62
[27] Sommerville., Seite 62
[28] Yu Beng Leau u. a., „Software Development Life Cycle AGILE vs Traditional Approaches“, 2012., Seite 2
[29] Sommerville, *Software Engineering*., Seite 87ff
[30] Ein symbolischer Manifest für agile Softwareentwicklung auf das sich viele der führenden Entwickler geeinigt haben.
[31] Kent Beck u. a., „Manifesto for Agile Software Development“, 2001, http://agilemanifesto.org/.

- Individuen und Interaktionen werden höher geschätzt als Prozesse und Werkzeuge
- Funktionierende Software wird höher geschätzt als umfassende Dokumentation
- Zusammenarbeit mit dem Kunden wird höher geschätzt als Vertragsverhandlung
- Reagieren auf Veränderung wird höher geschätzt als das Befolgen eines Plans

Es gibt viele verschiedene agile Methoden. Eine der bekanntesten, SCRUM, wird im Folgenden beschrieben.

2.2.1 SCRUM

Scrum ist eine einfache Methode für die agile Software-Entwicklung. Sie passt sich an wechselnde Anforderungen an und gibt die Software in kleinen Veröffentlichungszyklen, den Sprints, frei. So wird diese jederzeit in einem potenziell lieferbaren, ordnungsgemäß integrierten und getesteten Zustand gehalten.[32]

Scrum definiert folgende drei Rollen: Scrum Master (räumt alle Probleme, die die Infrastruktur und Organisation betreffen aus dem Weg), Product Owner (definiert und priorisiert die Anforderungen) und das Entwicklungsteam.[33]

Es gibt drei Phasen in Scrum [34]:

- Allgemeine Planungsphase – Ziele des Projekts werden festgelegt, Architektur wird entworfen.
- Sprint-Zyklen – mehrere Zyklen in denen Inkremente des Systems entwickelt werden.
- Projektabschlussphase – die erforderliche Dokumentation (Benutzerhandbücher…) wird vervollständigt, das Projekt wird beendet.

---

[32] Sakshi Sachdeva, „Agile Methodologies“, *(IJCSIT) International Journal of Computer Science and Information Technologies* 7(1), 2016 (2016)., Seite 4

[33] Henning Wolf und Stefan Roock, *Agile Softwareentwicklung*, 4. Aufl. (Heidelberg: dpunkt.verlag, 2015)., Seite 6

[34] Sommerville, *Software Engineering*., Seite 104

Die Zentrale Phase, das Sprint-Zyklus, dauert normalerweise zwei bis vier Wochen und läuft in vier Phasen ab[35]:

- Bewertungsphase – Der Produkt-Backlog (einer Liste der Aufgaben, die ausgeführt werden müssen) wird überprüft und die Prioritäten werden gesetzt. Der Kunde kann in dieser Phase neue Aufgaben oder Anforderungen anbringen.
- Auswahlphase – die Funktionalitäten, die während dieses Sprints entwickelt werden sollen werden ausgewählt.
- Entwicklung der Software - hier werden tägliche Treffen aller Teammitglieder abgehalten um den Fortschritt zu überprüfen.
- Besprechung – die Arbeit wird den Projektbeteiligten vorgestellt und der nächste Sprint-Zyklus beginnt.

2.3 Requirements Engineering

Anforderungsanalyse ist ein Teil des Softwareentwicklungsprozesses und ist von großer Bedeutung für die erfolgreiche Entwicklung von Systemen. Missverständliche, unvollständige, oder falsche Anforderungen sind mit hohen Kosten verbunden. Je später die Fehler im Verlauf des Entwicklungsprojekts entdeckt werden, desto höher können diese ausfallen. Um einen erfolgreichen Projektablauf zu ermöglichen, müssen die Fehler und Lücken in Anforderungsdokumenten bereits in Requirements Engineering erkannt und behoben werden.[36]

2.3.1 Begriffe

Requirements Engineering wird durch IREB[37] wie folgt definiert:

[35] Sommerville., Seite 104f
[36] Pohl und Rupp, *Basiswissen Requirements Engineering*., Seite 2
[37] International Requirements Engineering Board

>> Das Requirements Engineering ist ein systematischer und disziplinierter Ansatz zur Spezifikation und zum Management von Anforderungen mit den folgenden Zielen:

(1) Die relevanten Anforderungen zu kennen, Konsens unter den Stakeholdern über die Anforderungen herzustellen, die Anforderungen konform zu vorgegebenen Standards zu dokumentieren und die Anforderungen Systematisch zu managen.
(2) Die Wünsche und Bedürfnisse der Stakeholder zu verstehen, zu dokumentieren sowie die Anforderungen zu Spezifizieren und zu managen um das Risiko zu minimieren, dass das System nicht den Wünschen und Bedürfnissen der Stakeholder entspricht. << [38]

Eine Anforderung wird durch IEEE[39] wie folgt definiert: >>

(1) A condition or capability needed by a user to solve a problem or achieve an objective.
(2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed documents.
(3) documented representation of a condition or capability as in (1) or (2). << [40]

Die Haupttätigkeiten im Requirements Engineering sind[41]:

- Ermitteln – Anforderungen werden mit Hilfe von verschiedenen Techniken ermittelt
- Dokumentieren – Die erarbeiteten Anforderungen werden durch die Dokumentation beschrieben
- Prüfen und abstimmen – um die geforderten Qualitätskriterien zu sichern werden die dokumentierten Anforderungen geprüft und abgestimmt.
- Verwalten – Anforderungen werden verwaltet (strukturiert, für unterschiedliche Rollen aufbereitet und umgesetzt).

[38] Pohl und Rupp, *Basiswissen Requirements Engineering*., Seite 4
[39] Institute of Electrical and Electronics Engineers – weltweiter Berufsverband von Ingenieuren aus den Bereichen Elektrotechnik und Informationstechnik
[40] IEEE, *IEEE Standard Glossary of Software Engineering Terminology*, *Office*, Bd. 121990, 1990, https://doi.org/10.1109/IEEESTD.1990.101064.
[41] Pohl und Rupp, *Basiswissen Requirements Engineering*., Seite 4f

2.3.2 Der Prozess des Requirements -Engineering

Die im Kapitel 2.3.1. beschriebenen Tätigkeiten sind immer ein Bestandteil der Requirements Engineering Prozesse. Laut Sommerville[42] umfasst ein solcher Prozess folgende Aktivitäten:

Durchführbarkeitsstudie – Eine kurze Studie, deren Ergebnis eine Entscheidung ermöglicht, ob es sinnvoll ist das Projekt weiter fortzufahren. Sie sollte möglichst frühzeitig im Requirements Engineering Prozess stattfinden und die Fragen über das Nutzen für das Unternehmen und das Kosteneffektivität des Systems beantworten.[43]

Anforderungserhebung- und Analyse – In dieser Phase werden die Anforderungen durch die Projektbeteiligten mit Hilfe von verschiedenen Techniken (Dokumentationen, die Geschäftsprozesse beschreiben; Interviews, Szenarios, Brainstorming…) gesammelt, klassifiziert und priorisiert. Da es viele unterschiedliche Quellen für die Anforderungen gibt (Projektbeteiligte, Anwendungsgebiete, schon vorhandene Systeme), die das Problem auf unterschiedlicher Weise darstellen, kann diese Phase durch ViewPoints - Integration unterstützt werden um diese Anforderungen zu strukturieren.[44] ViewPoints- Integration wird in Kapitel 3 behandelt.

Anforderungsspezifikation – In dieser Phase werden die gesammelten Informationen, die während der Anforderungsanalyse gewonnen wurden dokumentiert. Dabei unterscheidet man zwei Arten von Anforderungen: Benutzeranforderungen (Beschreibung der Dienste, die dem Nutzer zur Verfügung gestellt werden sollten) und Systemanforderungen (Beschreibung der Dienste, Funktionen und Beschränkungen des Systems).[45]

Anforderungsvalidierung – Hier wird mit Hilfe von verschiedenen Techniken (Anforderungsreviews, Prototypen, Testfallerzeugung) geprüft, ob die Anforderungen die Qualitätskriterien[46] wie Vollständigkeit, Konsistenz usw.

[42] Sommerville, *Software Engineering*., Seite 150
[43] Sommerville., Seite 64, 133
[44] Sommerville., Seite 133ff
[45] Sommerville. , Seite 65, 115
[46] Siehe Abschnitt 2.3.4.

erfüllen, und die etwaige Fehler im Anforderungsdokument werden korrigiert.[47]

Anforderungsmanagement – Nach der Installation und während der regelmäßigen Nutzung des Systems, entstehen neue Anforderungen. Benutzer entdecken neue Bedürfnisse und Prioritäten. Das führt dazu, dass es zu Änderungen der Systemanforderungen kommt. Anforderungsmanagement verwaltet und steuert diese Änderungen. Dabei müssen die Auswirkungen der Anforderungsänderungen abgeschätzt werden können.[48]

Die Abläufe innerhalb des Requirements Engineering Prozesses werden nicht einfach nacheinander ausgeführt. Das ist ein iterativer Prozess mit verschachtelten Aktivitäten. Die Abbildung 6. zeigt das Spiralmodell des Requirements Engineering Prozesses[49] in dem man diese Verschachtelung sieht.

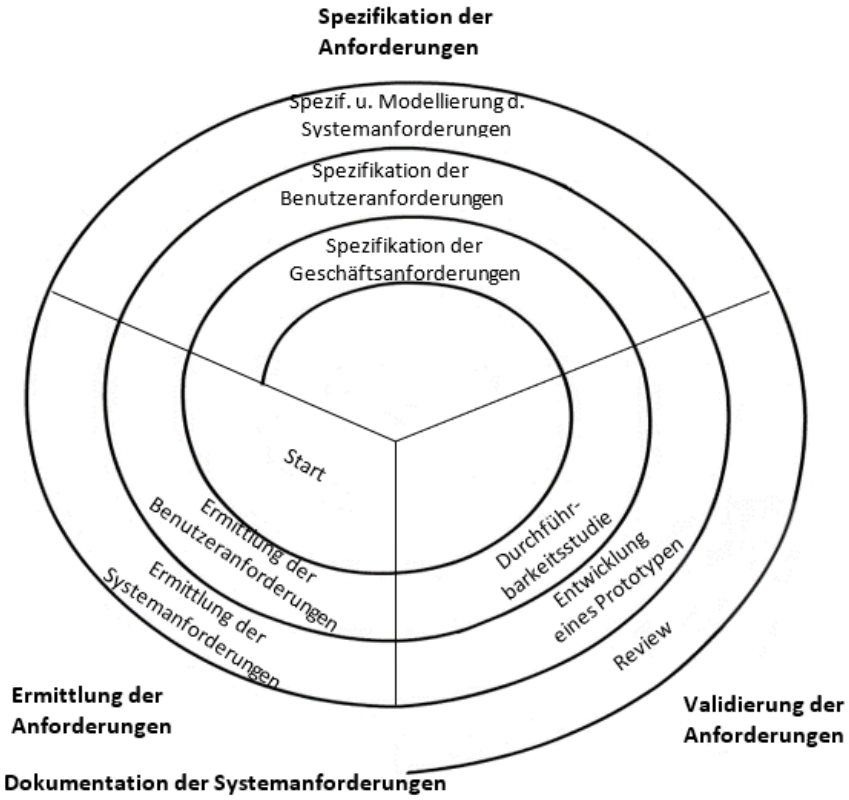


Abbildung 6. Das Spiralmodell des Requirements Engineering Prozesses

[47] Sommerville, *Software Engineering*., Seite 65, 144f
[48] Sommerville., Seite 146
[49] Sommerville., Seite 132

2.3.3 Die Klassifikation von Anforderungen

Es gibt unterschiedliche Ansätze um Anforderungen zu unterteilen. IREB unterscheidet z.B. drei Arten von Anforderungen: Funktionale Anforderung, Qualitätsanforderung und Randbedingungen.[50]

Die SOPHISTen und Rupp unterteilen die Anforderungen einerseits nach ihrer Art und ihrem Inhalt und andererseits nach der rechtlichen Verbindlichkeit von Anforderungen.[51]

Einteilung von Anforderungen nach ihrer Art:

- Funktionale Anforderungen – beschreiben die Funktionalität, die das geplante System zur Verfügung stellen soll[52]
- Technologische Anforderungen – legen die Anforderungen fest, die einen direkten Einfluss auf die in der Entwicklung zu verwendeten Technologien haben.[53]
- Qualitätsanforderungen – definieren qualitative Eigenschaften, die Prozesse oder am Prozess beteiligte Personen, einzelne Funktionen des Systems oder das ganze System aufweisen müssen.[54]
- Anforderungen an die Benutzungsoberfläche – spezifizieren die Bedienung und das Erscheinen (Optik, Akustik oder Haptik) der Benutzeroberfläche des Produkts.[55]
- Anforderungen an sonstige Lieferbestandteile – legen alle zu liefernden Produkte, die für den Betrieb und die Anwendung des Systems dienen fest.[56]
- Anforderungen an durchzuführende Tätigkeiten – beschreiben alle durchzuführenden Tätigkeiten für die Einführung und den Betrieb des Produktes.[57]

---

[50] Pohl und Rupp, *Basiswissen Requirements Engineering*., Seite 8f
[51] Chris Rupp und die SOPHISTen, *Requirements-Engineering und Management; Aus der Praxis von klassisch bis Agil*, 6. Aufl. (Carl Hanser Verlag München, 2014)., Seite 17
[52] Pohl und Rupp, *Basiswissen Requirements Engineering*., Seite 8
[53] Rupp und die SOPHISTen, *Requirements-Engineering und Management; Aus der Praxis von klassisch bis Agil*., Seite 277
[54] Rupp und die SOPHISTen., Seite 281
[55] Rupp und die SOPHISTen., Seite 286
[56] Rupp und die SOPHISTen., Seite 292
[57] Rupp und die SOPHISTen., Seite 294

- Rechtlich-vertragliche Anforderungen – spezifizieren alle Rechte und Pflichten in Bezug auf Entwicklung und Verwendung des zu erstellenden Produktes.[58]

Der Begriff „nicht-funktionale Anforderungen wird auch oft verwendet. Dazu gehören alle Anforderungen die keine funktionale Anforderungen sind (Technologische Anforderungen, Qualitätsanforderungen, Anforderungen an die Benutzeroberfläche…).[59]

Bei den Sicherheitsanforderungen begrenzen wir uns auf die zwei Arten von Anforderungen: funktionalen und nicht-funktionalen Anforderungen (ohne weitere Unterteilungen). Funktionale Sicherheitsanforderungen beschreiben eine bestimmte Funktion die Sicherheit der Software erfordert (wie z.B. Verschlüsselung oder Authentifizierung), die nicht-funktionale Sicherheitsanforderungen beschreiben die Sicherheitseigenschaften die die Software haben soll, oder die, die für den Schutz der Informationen bereitgestellt werden sollen.[60]

Im Kapitel 2.4 wird speziell auf die Sicherheitsanforderungen näher eingegangen.

Die Anforderungen nach der rechtlichen Verbindlichkeit beschreiben die Grad der Bedeutung der Angaben in der Anforderungsspezifikation für den Stakeholder. So wird festgelegt, welche Anforderungen für das Funktionieren des Systems unverzichtbar sind und welche z.B. in einer späteren Version realisiert werden könnten. Und bei Verträgen z.B. wird festgelegt, welche Teile des Vertrags rechtlich einklagbar sind. Die rechtliche Verbindlichkeit wird durch die Verwendung bestimmter Modalwerben ausgedrückt. Die SOPHISTen verwenden dabei drei Schlüsselwörter: „shall“ (rechtlich verbindliche, zwingend zu erfüllende Anforderung), „should“ (Anforderungen, von deren Umsetzung unter bestimmten Voraussetzungen abgesehen werden kann) und „will“ (ermöglicht zukünftige Anforderungen anzukündigen).[61]

[58] Rupp und die SOPHISTen., Seite 296
[59] Rupp und die SOPHISTen., Seite 17
[60] Paulus Sachar, „Geeignete Anforderungen für sichere Software“, 2012.
[61] Rupp und die SOPHISTen, *Requirements-Engineering und Management; Aus der Praxis von klassisch bis Agil*., Seite 18f

2.3.4 Qualitätsanforderungen im Requirements-Engineering

Um qualitativ und quantitativ hochwertige Anforderungen zu schreiben müssen einige Qualitätskriterien erfüllt werden. Im Standard ISO/IEC/IEEE 29148-2011 wurden sowohl Qualitätskriterien für einzelne Anforderungen als auch Qualitätskriterien für Anforderungsspezifikationen festgelegt.[62]

Qualitätskriterien für einzelne Anforderungen:

- **Notwendig –** Die Anforderung ist aktuell und definiert nur die notwendige Eigenschaft, Leistung, Einschränkung oder Qualitätsfaktor. Die Anforderung, die einen geplanten Gültigkeitsdatum hat, soll eindeutig markiert werden.
- **Frei implementierbar** – Die Anforderung beschreibt nur das, was notwendig und ausreichend für das System. Sie vermeidet unnötige Einschränkungen des architektonischen Entwurfs und hat das Ziel, umsetzungsunabhängig zu sein.
- **Eindeutig** – Die Anforderung ist einfach und leicht verständlich und so formuliert, dass sie nur in einer Weise interpretiert werden kann.
- **Konsistent** – Die Anforderung ist frei von Konflikten mit anderen Anforderungen.
- **Vollständig** – Die Anforderung ist messbar und beschreibt die geforderte Funktionalität und Merkmale des Systems ausreichend.
- **Atomar** – Die Anforderungsbeschreibung enthält nur eine Anforderung.
- **Realisierbar** – Die Anforderung ist mit akzeptablen Risiko technisch realisierbar und passt in Systembeschränkungen (z. B. Kosten, Zeitplan, technische, rechtliche, regulatorische).
- **Verfolgbar** – alle Eltern-Kind-Beziehungen für die Anforderung sind identifiziert, so dass alle abgeleiteten Anforderungen von ihrer Quelle bis zu Implementierung verfolgt werden können.
- **Prüfbar** – Die Anforderung ist prüfbar, d. H. es kann nachgewiesen werden, dass das System die Anforderung erfüllen kann. Die Überprüfbarkeit wird verbessert, wenn die Anforderung messbar ist.

[62] International Organization for Standardization. u. a., *Systems and software engineering - life cycle processes - requirements engineering = Ingénierie des systèmes et du logiciel - processus de cycle de vie - ingénierie des exigences.*, 1. Aufl. (ISO, 2011)., Seite 11f

Qualitätskriterien für Anforderungsspezifikationen:

- **Vollständig** – Die Anforderungsspezifikation enthält alles was für die Definition des Systems oder Systemelements, das spezifiziert wird, relevant ist.
- **Konsistent** – Die Anforderungsspezifikation hat weder doppelten Anforderungen noch individuellen Anforderungen, die widersprüchlich sind. Derselbe Begriff wird für dieselbe Beschreibung in allen Anforderungen verwendet.
- **Erschwinglich** – Die Anforderungsspezifikation kann durch eine Lösung erfüllt werden, die innerhalb der Beschränkungen des Lebenszyklus (z. B. Kosten, Zeitplan, technische, rechtliche, regulatorische) durchführbar ist.
- **Begrenzt** - Die Anforderungsspezifikation ist nur auf die zu erfüllenden Benutzeranforderungen begrenzt.

2.4 SICHERHEIT UND RISIKOFAKTOREN

2.4.1 Begriffe

Die Sicherheit (Security) wird durch ISO (International Standards Organisation) als *„die Minimierung der Verwundbarkeit von Werten und Ressourcen“* definiert.[63]

Präziser ausgedruckt kann man die Sicherheit wie folgt definieren:

Funktionssicherheit (Safety) eines Systems ist die Eigenschaft eines Systems, dass es unter allen normalen Betriebsbedingungen funktioniert. D.h. die spezifizierte Soll-Funktionalität der Komponenten stimmt mit der realisierten Ist-Funktionalität überein und das System nimmt keine

[63] ISO, zit. nach Eckert Claudia, *IT-Sicherheit : Konzepte - Verfahren - Protokolle*, 8. Aufl. (Oldenbourg Verlag München, 2013)., Seite 6

funktional unzulässigen Zustände an. Funktionssicherheit ist die Grundlage für die Informationssicherheit und Datensicherheit eines Systems.[64]

Informationssicherheit (Security) ist die Eigenschaft eines funktionssicheren Systems nur die Systemzustände anzunehmen, die keine unautorisierte Informationsgewinnung oder Informationsänderung erlauben.[65]

Datensicherheit (Protection) ist die Eigenschaft eines funktionssicheren Systems nur die Systemzustände anzunehmen, die keine unautorisierten Zugriffe auf Daten und auf Systemressourcen erlauben. Datensicherheit umfasst auch die Maßnahmen zur Datensicherung.[66]

Ein System kann eine **Schwachstelle** haben (eine Schwäche, oder einen Punkt an dem das System verwundbar werden kann). Die Schwachstelle in einem IT System, über die die Sicherheit des Systems gefährdet werden kann, indem die Sicherheitsdienste des Systems umgegangen, getäuscht oder unautorisiert modifiziert werden nennt man die **Verwundbarkeit**.[67]

Eine **Bedrohung** des Systems nutzt eine oder mehrere Schwachstellen oder Verwundbarkeiten, um die Authentizität von Personen zu gefährden, oder einen Verlust der Informationsvertraulichkeit, oder der Verfügbarkeit, oder der Datenintegrität zu erreichen.[68]

Das **Bedrohungsrisiko** ist die Wahrscheinlichkeit, dass ein Ereignis, das Schaden verursacht, eintritt und die Höhe des dadurch hervorgerufenen Schadens.[69]

Das **Risikomanagement** bedeutet die Überwachung, die Identifizierung und die Kontrolle von Risiken. Ziel des Risikomanagements ist es, die identifizierten Risiken, basierend auf deren Schweregrad, zu priorisieren und die Anwendungen erstellen, die die Risiken reduzieren können.[70]

---

[64] Eckert Claudia., Seite 6
[65] Eckert Claudia.
[66] Eckert Claudia.
[67] Eckert Claudia., Seite 16
[68] Eckert Claudia., Seite 17
[69] Eckert Claudia., Seite 18
[70] Jouko Ahola u. a., *Handbook of The Secure Agile Software Development Life Cycle* (University of Oulu, 2014)., Seite 29

2.4.2 Die Schutzziele

Die zu schützenden Güter informationssicherer bzw. datensicherer Systeme sind Informationen bzw. Daten.[71]

Die Schutzziele sind:

- **Vertraulichkeit** (Confidentiality) – das System wird von unautorisierten Informationsgewinnung geschützt. Um dieses Ziel zu erreichen ist es erforderlich die Berechtigungen und Kontrollen so festzulegen, dass ausgeschlossen werden kann, dass die Informationen zu unautorisierten Personen gelangen.[72]
- **Verfügbarkeit** (Availability) – die Dienstleistungen und Funktionen des Systems oder auch die Informationen stehen zum geforderten Zeitpunkt zur Verfügung.[73]
- **Integrität** (Unversehrtheit) – die Daten werden von unerlaubten Veränderungen, von den Manipulationen des Erstellungszeitpunktes, oder von verfälschten Angaben zum Autor geschützt. Ziel ist es, dass die Daten unverändert und vollständig bleiben.[74]
- **Authentizität** – Die Authentisierung ist die Prüfung und Verifizierung der Identität einer Person bei der Anmeldung in einem System, oder auch die Prüfung der Identität von IT-Komponenten und Anwendungen.[75]
- **Verbindlichkeit** – Die Aktionen in einem System sind zuordenbar, so, dass es nicht möglich ist, dass eine Person im Nachhinein die Durchführung einer Aktion abstreiten kann. Eine große Bedeutung haben die Verbindlichkeitseigenschaften eines Systems besonders im Bereich des E-Commerce und E-Business.[76]
- **Anonymisierung und Pseudonymisierung** – Bei der Anonymisierung werden die personenbezogenen Daten so verändert, dass es gar nicht mehr, oder sehr schwierig ist (mit hohen Kosten und großen

[71] Eckert Claudia, *IT-Sicherheit : Konzepte - Verfahren - Protokolle*., Seite 7
[72] Eckert Claudia., Seite 10
[73] Bundesamt für Sicherheit in der Informationstechnik, *Leitfaden Informationssicherheit*, 2012., Seite 14
[74] Bundesamt für Sicherheit in der Informationstechnik., Seite 14
[75] Bundesamt für Sicherheit in der Informationstechnik.
[76] Eckert Claudia, *IT-Sicherheit : Konzepte - Verfahren - Protokolle*.

Zeitaufwand verbunden), die persönlichen und sachlichen Einzelangaben einer bestimmten Person zuzuordnen. Pseudonymisierung ist eine schwächere Form der Anonymisierung. Hier werden die Zuordnungsvorschriften verwendet (z.B. die Pseudonymen) um die personenbezogenen Daten so zu verändern, dass die persönliche und sachliche Einzelangaben ohne Kenntnis der Zuordnungsvorschrift nicht mehr einer Person zugeordnet werden können.[77]

Ein IT System kann unterschiedliche Schwachstellen besitzen und ist vielen Bedrohungen ausgesetzt, wodurch die Schutzziele beeinträchtigt werden können. Die Gefährdungsfaktoren können wie folgt klassifiziert werden[78]:

- Höhere Gewalt: z.B. Blitzschlag, Überschwemmung, Feuer, Erdbeben…
- Fahrlässigkeit: z.B. Irrtum, unsachgemäße Behandlung, Fehlbedienung…
- Vorsatz: z.B. Manipulation, Einbruch, Hacking, Vandalismus, Spionage, Sabotage…
- Technisches Versagen: z.B. Stromausfall, Fehlfunktionen, Hardware-Ausfall…
- Organisatorische Mängel: z.B. ungeschultes Personal, Raubkopie, unberechtigter Zugriff…

[77] Eckert Claudia., Seite 13
[78] Eckert Claudia., Seite 17

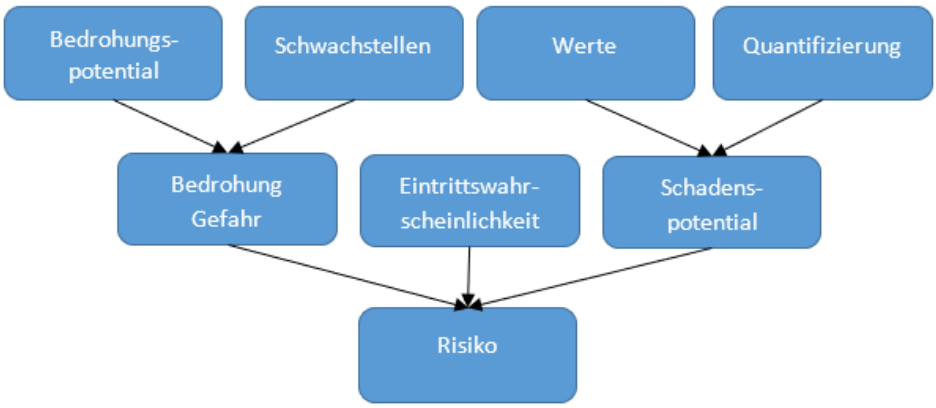


Abbildung 7. Zusammenhang zwischen Schwachstellen, Bedrohungen und Risiken[79]

Die Abbildung 7. Zeigt den Zusammenhang zwischen Schwachstellen, Bedrohungen und Risiken. Um herauszufinden welches Risiko in einem System besteht, wird zuerst geklärt welche konkreten Bedrohungen und Gefahren für das System zu beachten sind. Weiters wird die Eintrittswahrscheinlichkeit für die Bedrohungen abgeschätzt. Dazu wird ein Schadenspotential ermittelt, dass ein erfolgreicher Angriff bringen würde. Die Ermittlung des Schadenspotentials ist sehr schwierig. Dabei werden die Sicherheitsrelevanten Objekte erfasst und deren Bedeutung innerhalb des Systems ermittelt. Das setzt großes Fachwissen über organisatorische Strukturen und Unternehmensprozesse, und gute Kenntnisse über die technischen Komponenten und deren Sicherheitsprobleme voraus.[80]

2.5 Security Requirements Engineering

2.5.1 Sicherheitsmechanismen

Die im Kapitel 2.4.2 beschriebenen Schutzziele stellen im Wesentlichen die Sicherheitsanforderungen, die an heutige IT Systeme gestellt werden dar. Im

[79] Eckert Claudia., Seite 18
[80] Eckert Claudia., Seite 18

Folgenden werden die Sicherheitsmechanismen mit denen die Sicherheitsanforderungen erfüllt werden können beschrieben[81]:

- Vertraulichkeit – hier können Verschlüsselungsverfahren eingesetzt werden um die Informationen gegenüber Unbefugten geheim zu halten
- Verfügbarkeit – hier ist es möglich z.B. durch redundante Systeme die Verfügbarkeit zu gewährleisten
- Integrität – die unautorisierten Änderungen von Daten können mit Hilfe von Message Authentication Code, digitaler Signaturen sowie One-Way- Hashfunktionen erkannt werden
- Authentizität – die Prüfung der Identität der Person die sich angemeldet hat kann mittels Authentifizierungsverfahren (bzw. Authentifizierungsprotokollen erfolgen, und die Echtheit der Daten und Anwendungen kann durch digitale Signaturen und Message Authentication Codes geprüft werden
- Verbindlichkeit – die Nichtabstreitbarkeit wird durch Non-Repudiation-Zertifikate, Non-Repudiation Tokens und Non-Repudiation Protokolle ermöglicht. Diese basieren auf digitalen Signaturen oder Message Authentication Codes in Verbindung mit den Diensten Timestamping Services, Notary Services und Evidence Recording[82]
- Anonymisierung und Pseudonymisierung – Anonymisierungsdienste versuchen durch das Unterdrücken von Daten und Verschleierungstechniken (Verschlüsseln, Ersetzen durch Standardmuster) Anonymisierung und Pseudonymisierung zu erreichen. Pseudonymisierung kann man auch durch die Einführung von Pseudonymen erreichen.[83]

Wie im Kapitel 2.3.3 schon beschrieben werden zwei Arten von Sicherheitsanforderungen unterschieden: funktionale und nicht funktionale Sicherheitsanforderungen. Laut Sachar Paulus haben die nicht-funktionalen Sicherheitsanforderungen hier eine größere Bedeutung, da sich alle Sicherheitsfunktionen (funktionale Sicherheitsanforderungen) auf damit zu

[81] Petra Wohlmacher, „Sicherheitsanforderungen und Sicherheitsmechanismen bei IT-Systemen“, 2000.
[82] Wohlmacher.
[83] Eckert Claudia, *IT-Sicherheit : Konzepte - Verfahren - Protokolle*., Seite 13

erzielende Sicherheitseigenschaften (nicht-funktionale Sicherheitsanforderungen) zurückführen lassen. Die funktionalen Anforderungen sind in der Regel das Ergebnis einer Architekturentscheidung mit der man bestimmte nicht-funktionale Sicherheitsanforderungen umsetzt.[84]

Das Ermitteln der Anforderungen erweist sich oft als schwierig, da sich die Kunden selten Gedanken über den Sicherheitsanforderungen machen. Es gibt zwei Möglichkeiten dieses Problem zu lösen. Eine Möglichkeit ist dem Kunden zu helfen die Kompetenz zu erlangen die Sicherheitsanforderungen selbst zu bestimmen (z.B. in einem Workshop). Eine andere Möglichkeit ist, dass man es für den Kunden übernimmt. Da besteht aber die Gefahr, nicht alle Risiken zu erfassen.[85]

Bei der Ermittlung von geeigneten Anforderungen gibt es zwei Herangehensweisen[86]:

- Über die Geschäftsprozesse des Kunden – hier wird der mögliche Schaden des Angriffs in Vordergrund gestellt.
    - Vorteil: diese Herangehensweise ist näher an der Denkweise des Kunden und erlaubt eine Business-orientierte Auseinandersetzung mit der Problematik
    - Nachteil: der technische Fokus bzw. Hintergrund fehlt. So können Fehleinschätzungen der Wahrscheinlichkeit und der Machbarkeit von Angriffen entstehen
- Über die geplante Softwarearchitektur – dieser Ansatz widmet sich eher den möglichen Angriffen
    - Vorteil: Betrachtet nur tatsächlich technisch relevante Aspekte
    - Nachteil: Die Gefahr, dass der Fokus auf Risiken gelegt wird, die in der Praxis für den Kunden nur eine geringe Relevanz haben

[84] Sachar, „Geeignete Anforderungen für sichere Software“.
[85] Sachar.
[86] Sachar.

2.5.2 Sicherheit im Softwarelebenszyklus

Die Sicherheit sollte in jeder Phase des Softwarelebenszyklus beachtet werden. Die Anforderungen an die Sicherheit des zu erstellenden Systems sollten von Anfang an und in jeder Phase bedacht und umgesetzt werden. Dabei sollen alle beteiligten Rollen mit den Sicherheitsanforderungen der jeweiligen Phase vertraut sein. Denn, je früher im Softwarelebenszyklus die Sicherheitsanforderungen beachtet und umgesetzt werden, desto kostengünstiger ist es. Die Kosten der Behebung von Sicherheitslücken können in späteren Phasen bis zu 80fach höher sein als zu Beginn des Lebenszyklus.[87]

Die im Kapitel 2.1. beschriebenen Vorgehensmodelle eignen sich alle für sichere Softwareentwicklung. Es gibt aber, je nach Typ, verschiedene Stärken [88]:

- **Das Wasserfallmodell** z.B. eignet sich sehr gut für Softwaresicherheit. Aufgrund einer definierten Anforderungsphase ist die Menge der Sicherheitsanforderungen bekannt. Außerdem werden in diesem Modell die Informationen bezüglich Sicherheit von Phase zur Phase übergeben und diesbezügliche Umsetzungen werden vereinbart.
- **V-Modell** ist aufgrund der Testphasen sehr gut geeignet für die Kontrolle der Sicherheitsanforderungen. Da die nachträgliche Aufnahme von Anforderungen in diesem Modell nicht vorgesehen ist, ist die Sicherheit des fertigen Produktes sehr stark von den Sicherheitszielen, die am Anfang des Prozesses definiert wurden abhängig.
- **Das Spiralmodell** ist für die Sicherheit sehr gut geeignet, da die Spirale jede Phase der Entwicklung mehrmals durchläuft und in jedem Durchlauf neue Sicherheitsanforderungen definiert werden können.
- Die **agilen Methoden** sind für sichere Software gut geeignet, wenn die Sicherheitseigenschaften außerhalb des Sprint-Rhythmus

[87] Rosemaria Giesecke und Stefan Fünfröcken, „Einfach zu mehr Software-Sicherheit: Unterstützung zur sicheren Softwareentwicklung“, 2007.
[88] Sachar, *Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering*., Seite 48ff

verwaltet werden und wenn die dafür spezielle Maßnahmen eingeführt werden, weil sie schwäche in der Sicherstellung und im Nachweis von nicht funktionalen Sicherheitsanforderungen haben.[89]

2.5.3 Security Requirements Engineering Methoden: MSRA und SQUARE

Es existieren mehrere Methoden des Security Requirements Engineering. Im Folgenden werden zwei Multilaterale Methoden, Multilateral security requirements analysis (MSRA) und Security quality requirements engineering (SQUARE), kurz vorgestellt.

Multilateral security requirements analysis (MSRA) – lehnt sich an das ViewPoint- orientierte requirements Engineering an. Ziel dieser Methode ist die Prinzipien der multilateralen Sicherheit während der Requirements Engineering Phase anzuwenden. Das erfolgt durch die Analyse der Sicherheitsbedürfnisse aller Stakeholder. Die Konflikte werden identifiziert und die verschiedene Ansichten werden konsolidiert. Die MSRA Nutzer erstellen die Sicherheitsanforderungen aus der Sicht der verschiedenen Beteiligten und durch die Abstimmung multilateraler Sicherheitsziele ergeben sich die Sicherheitsanforderungen. [90]

MSRA Methode erfolgt in folgenden Schritten[91]:

1. Identifikation von Stakeholdern: In diesem Schritt werden Stakeholder (Beteiligte, die funktionale, Sicherheits-, Privatsphären- oder Informationsinteressen in dem zukünftigen System haben) identifiziert.
2. Identifikation von Episoden: Schritt 2 identifiziert Episoden. Episoden sind ähnlich wie Szenarien, haben aber eine geringere Granularität. Sie identifizieren Gruppen von Funktionen, die für Benutzer von Bedeutung sind.
3. Ausarbeitung der Sicherheitsziele: Hier werden die Sicherheitsziele der verschiedenen Stakeholder für jede Episode identifiziert und beschrieben.

[89] Sachar., Seite 49
[90] Benjamin Fabian u. a., „A comparison of security requirements engineering methods“, 2009., Seite 16f
[91] Fabian u. a., Seite 17

4. Identifikation von Fakten und Annahmen: Die für die Festlegung von Sicherheitszielen relevanten Eigenschaften der Umgebung werden in diesem Schritt identifiziert.
5. Verfeinern der Stakeholder-Ansichten: Die Ansichten der Stakeholder unter Berücksichtigung der Beziehungen zwischen den Episoden, Fakten und Annahmen werden beschrieben.
6. Koordinierung von Sicherheitszielen: Die Konflikte zwischen den Sicherheitszielen werden identifiziert, die Kompromisse zwischen in Konflikt stehenden Zielen werden gefunden und ein konsistenter Satz von Sicherheitssystemanforderungen wird hergestellt.
7. Koordinierung von Sicherheits- und Funktionalen-Anforderungen: Die widersprüchlichen Funktionalen- und Sicherheitsanforderungen werden identifiziert und es werden die Funktionalen- gegen die Sicherheitsanforderungen und umgekehrt abgewogen.

MSRA ist in die Anforderungsanalyse integriert und kann angewendet werden, sobald die ersten funktionalen Anforderungen des Systems identifiziert sind. Sie verwendet UML-Modelle, um Episoden, die funktionalen und Sicherheitsanforderungen zu erfassen. [92]

Security quality requirements engineering methodology (SQUARE) – ist eine Methode die Sicherheitsanforderungen für IT-Systeme und -Anwendungen ermittelt, kategorisiert und priorisiert. [93] Das Ziel von SQUARE ist es, die Sicherheitsaspekte in Softwareentwicklungsprozesse zu integrieren. SQUARE bietet einen organisatorischen Rahmen für die Durchführung von Sicherheitsanforderungen.[94]

Es besteht aus 9 Schritten:

1. Definitionen vereinbaren: Aufgrund der Unterschiede im Fachwissen und Erfahrung bei den Teilnehmern von SQUARE werden in dem ersten Schritt eine gemeinsame Terminologie und Definitionen festgelegt. So wird eine effektive und klare Kommunikation während des Requirements-Engineering-Prozesses gewährleistet. [95]

[92] Fabian u. a., Seite 17f
[93] Nancy R Mead, Eric D Hough, und Theodore R Stehney Ii, „Security Quality Requirements Engineering (SQUARE) Methodology“, 2005., Seite 3
[94] Fabian u. a., „A comparison of security requirements engineering methods“.
[95] Mead, Hough, und Stehney Ii, „Security Quality Requirements Engineering (SQUARE) Methodology“., Seite 8

2. Sicherheitsziele identifizieren: In diesem Schritt einigen sich die Stakeholder auf eine Reihe priorisierter Sicherheitsziele für das Projekt. Zuerst werden allgemeine Sicherheitsziele festgelegt, die sich oft widersprechen. Die Konflikte werden beseitigt und alle Interessen der Stakeholder in Einklang gebracht. Sobald die Ziele festgelegt sind, werden sie priorisiert.[96]
3. Artefakte entwickeln: In diesem Schritt sammeln Requirements Engineering-Team und die Stakeholder einen vollständigen Satz von Artefakten des Systems. Folgende Artefakten werden gesammelt: Systemarchitekturdiagramme, Anwendungsfälle / Diagramme, Missbrauchsszenarien / Diagramme, Angriffsstrukturen und standardisierte Vorlagen und Formulare. Die Erstellung und Dokumentation von Artefakten des Systems ist die Grundlage für den Erfolg des Projekts.[97]
4. Risikobewertung durchführen: In diesem Schritt werden die Schwachstellen und Bedrohungen denen das System ausgesetzt ist identifiziert. Weiters wird die Wahrscheinlichkeit, dass sich die Bedrohungen als echte Angriffe herausstellen, und mögliche Konsequenzen eines Angriffs identifiziert. Nachdem die Bedrohungen identifiziert wurden, werden sie nach Wahrscheinlichkeiten klassifiziert. So können die Sicherheitsanforderungen, die zu einem späteren Zeitpunkt generiert werden, priorisiert werden.[98]
5. Auswahlmethode bestimmen: Hier wird eine Auswahlmethode bestimmt, die für die Anzahl und das Fachwissen der Stakeholder, die Größe und den Umfang des Projekts und das Fachwissen des Requirements Engineering-Teams geeignet ist. [99]
6. Sicherheitsanforderungen ermitteln: In diesem Schritt werden die Sicherheitsanforderungen ermittelt. Die meisten Auswahlmetoden bieten detaillierte Anleitungen zur Ermittlung der Anforderungen. Wichtig dabei ist sicherzustellen, dass die Anforderungen nachprüfbar und gegebenenfalls quantifizierbar sind und dass durch

[96] Mead, Hough, und Stehney Ii., Seite 10
[97] Mead, Hough, und Stehney Ii., Seite 12
[98] Mead, Hough, und Stehney Ii., Seite 13
[99] Mead, Hough, und Stehney Ii., Seite 14

die Implementierung der Anforderungen keine Architekturbeschränkungen ausgelöst werden.[100]

7. Anforderungen kategorisieren: Hier werden die ermittelten Anforderungen nach den folgenden Kategorien eingestuft: wesentliche, unwesentliche, auf Systemebene, auf Softwareebene und Architekturbeschränkungen, wobei Architekturbeschränkungen ein Ausschlusskriterium ist. Das ist eine minimale Menge an Kategorien und jede Iteration von SQUARE wird wahrscheinlich eine viel größere Gruppe von Kategorien erzeugen. [101]
8. Anforderungen priorisieren: In den meisten Fällen können nicht alle Sicherheitsanforderungen implementiert werden, weil Zeit, Ressourcen oder Änderungen an den Zielen des Projekts fehlen. Deshalb werden in diesem Schritt die Anforderungen priorisiert und es wird entschieden welche Anforderungen in welcher Reihenfolge implementiert werden sollen. Die Ergebnisse von der Risikobewertung (Schritt 4) und der Kategorisierung (Schritt 7) sollen bei dieser Entscheidung helfen.[102]
9. Anforderungsinspektion: Dieser letzte Schritt ist einer der wichtigsten Elemente bei der Erstellung von genauen und überprüfbaren Anforderungen. Hier werden die Anforderungen auf Mängel wie Mehrdeutigkeiten, Inkonsistenzen, oder falsche Annahmen geprüft. An diesem Punkt ist SQUARE Prozess abgeschlossen und Sicherheitsanforderungsdokument für die Stakeholder kann erstellt werden.[103]

2.5.4 Modelle für die sichere Software

Common Criteria (CC) – Evaluationskriterien für IT Sicherheit – ISO/IEC 15408 – Das Ziel dieses Modells ist die Bewertung von Sicherheitseigenschaften von Software- und Hardwareprodukten. Das ist eine Methode zur Prüfung der Sicherheit einer Software und ist sehr gut

[100] Mead, Hough, und Stehney Ii., Seite 15
[101] Mead, Hough, und Stehney Ii., Seite 17
[102] Mead, Hough, und Stehney Ii., Seite 18
[103] Mead, Hough, und Stehney Ii., Seite 19

geeignet um Fehler und Schwachstellen im Laufe des Entwicklungsprozesses zu entdecken.[104]

Dieses Modell besteht aus drei Teilen: Einführung und Allgemeines Modell, funktionale Sicherheitsanforderungen und Anforderungen an die Vertrauenswürdigkeit.[105]

Einführung und das allgemeine Modell (engl. Introduction and general Modell) – hier wird das allgemeine Konzept der Evaluationskriterien beschrieben und die Begriffe wie Schutzprofile, Sicherheitsanforderungen, Sicherheitsziele und Target of Evaluation (TOE, Evaluationsgegenstand) eingeführt.

Funktionale Sicherheitsanforderungen (engl. Security functional Requirements) – hier werden die Sicherheitsanforderungen an die vordefinierten Sicherheitsfunktionalitäten beschrieben. Diese Anforderungen sind nach Klassen strukturiert. Jede Klasse besteht aus mehreren Familien, und jede Familie hat mindestens eine Komponente in der die Sicherheitsanforderungen beschrieben werden. Es können auch eigene Sicherheitsvorgaben als die Grundlage für die Evaluierung definiert werden.
Anforderungen an die Vertrauenswürdigkeit (engl. Security assurance requirements) – In diesem Teil werden die Kriterien für die Evaluierung von Sicherheitsvorgaben und Schutzprofilen spezifiziert. Die Sicherheitsanforderungen sind wie die funktionalen Anforderungen mittels Klassen, Familien und Komponenten strukturiert. Jede Komponente besteht aus Anforderungen an den Entwickler, Anforderungen an Inhalt und Form der Prüfnachweise und Anforderungen an den Evaluator.[106]

Common Criteria sind nur für die Anwendungen geeignet, deren Umfeld und deren Funktionalität möglichst stabil bleibt (Smartcards, Firewalls…).[107]

---

[104] Sachar, *Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering*., Seite 54f
[105] BITKOM Bundesverband Informationswirtschaft, Telekommunikation und neue Medien, „Kompass der IT-Sicherheitsstandards: Auszüge zum Thema Elektronische Identitäten", 2014., Seite 30
[106] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien., Seite 30f
[107] Sachar, *Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering*., Seite 56

ISO 27002 Entwicklung von Informationssystemen – enthält „Best Practices“ wie man die Kontrollvorgaben für den Informationssicherheitsmanagement-Prozess umsetzen kann. Teile von ISO 27002 sind je nach Version und Landesausgabe unterschiedlich angeordnet, bearbeiten aber folgende Themen[108]:

- Sicherheitsanforderungen von Informationssysteme: Die Erstellung eines Anforderungskatalogs für Sicherheitsaspekte.
- Korrekter Ablauf von Prozessen in der Software: Input-Validierung, Output Validierung, Nachrichtenintegrität und Kontrollmöglichkeiten für den fehlerfreien Ablauf von Softwareprozessen.
- Kryptographische Kontrollen: Vorgaben für den kontrollierten Einsatz von Verschlüsselungstechnologien und für die Verwendung von kryptografischen Schlüsseln.
- Sicherheit der Systemdateien: Vorgaben für den Schutz der Informationen und Programme, die während der Entwicklung, oder während des Betriebs manipuliert werden können.
- Sicherheit im Entwicklungsprozess und Support: Vorgaben zum Schutz von Entwicklungsprozessen, sowie Vorgaben zum Schutz von nachträglichen Änderungen der Software nach Auslieferung.
- Technische Schwachstellenmanagement: Beschreibung des Umgangs mit den Schwachstellen.

System Security Engineering Capability Maturity Modell (SSE CMM) – ISO/IEC 21827 – Dieses Modell ist eine Weiterentwicklung aus dem allgemeinen Capability maturity modell (CMM), das besonders in der Softwareentwicklung verbreitet ist. Das allgemeine CMM wurde an die speziellen Anforderungen des Sicherheitsmanagements angepasst.[109] Dieses Modell managt die Entwicklung von sicheren Software mit Hilfe von einer Reihe von Kontrollmaßnahmen, die jeweils unterschiedliche Reifegrade haben können. So kann eingeschätzt werden, ob und in welchem Maße die Voraussetzungen zur Entwicklung sicherer Software gegeben sind, und eigene Skills, bzw. die eines Lieferanten (als Kunde) können bewertet

[108] Sachar., Seite 57

[109] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien, „Kompass der IT-Sicherheitsstandards: Auszüge zum Thema Elektronische Identitäten“., Seite 33

werden.[110] SSE CMM unterteilt die sichere Softwareentwicklung in drei Hauptprozesse die voneinander abhängig sind: Risiko, Vertrauenswürdigkeit und System-Lebenszyklus. Es beschreibt eine Menge von Basis-Aktivitäten, die in 22 Prozessbereiche aufgeteilt sind. Elf Prozessbereiche beziehen sich auf die sichere Entwicklung und elf auf die Projektorganisation.[111]

Die Basis-Aktivitäten sind generisch und können einem der folgenden Reifegrad zugeordnet werden[112]:

- Reifegrad 0 – Nicht umgesetzt.
- Reifegrad 1 – Formlos umgesetzt – Einzelne Maßnahmen existieren, Prozess ist sehr instabil und kaum umgesetzt.
- Reifegrad 2 – Geplant und weiterverfolgt – Es existiert ein stabiler Prozess, der in Projekten mit einem Projektmanagement gelebt wird.
- Reifegrad 3 – Gut definiert – Ein Prozess ist definiert. Es gibt auch ein Prozessmodell, das eine Implementierung des Prozesses sicherstellt.
- Reifegrad 4 – Quantitativ kontrolliert – Es existieren Prozessdatenanalysen und Prozessmessungen. Diese werden zur Weiterentwicklung des Prozesses genutzt.
- Reifegrad 5 – Kontinuierlich verbessernd – Prozessbewertung und Prozessoptimierung wird vom Management regelmäßig durchgeführt.

Building Security In Maturity Modell (BSIMM) und OWASP Software Assurance Maturity Modell (OWASP SAMM) – Diese beiden Modelle haben eine hohe Überdeckung und die Herangehensweisen sind sehr ähnlich. Beide basieren auf den gleichen Anfängen.[113]
BSIMM besteht aus mehreren Aktivitäten die in vier Domänen und drei Reifegraden eingeteilt sind. Die Domänen sind[114]:

[110] Sachar, *Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering*., Seite 58 u. 60
[111] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien, „Kompass der IT-Sicherheitsstandards: Auszüge zum Thema Elektronische Identitäten“., Seite 33
[112] Bundesverband Informationswirtschaft, Telekommunikation und neue Medien., Seite 34
[113] Sachar, *Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering*., Seite 60f
[114] Sachar., Seite 60f

- Governance - beinhaltet die Aktivitäten zur Entwicklung von Strategien und Vorgabensystemen, zur Ausbildung von Mitarbeitern und zur Etablierung von Metriken.
- Intelligence - hier befinden sich die wissensorientierten Aktivitäten wie z.B. Anforderungen zur funktionalen Sicherheitsaspekten, Angriffsmodelle, Entwurfsmuster und Standards
- Secure Software Development Lifecycle (SSDL) Touch Points – beinhaltet Sicherheitsbezogene Aktivitäten die in den Entwicklungsprozess eingreifen wie z.B. die Architekturanalyse und Sicherheitstests
- Deployment – enthält Aktivitäten für Software, die für den Kunden vorbereitet wird, oder die schon beim Kunden ist (z.B. Penetrationstest, Konfigurationsmanagement, Umgang mit Schwachstellen, Sicherheit der Betriebsumgebung…)

Die Reifegrade sind[115]:

- Reifegrad 1 – Es existiert ein Verständnis für den Stand der Technik
- Reifegrad 2 – Die Aktivitäten werden mit dem vorhandenen Verständnis angeglichen.
- Reifegrad 3 – Eigene, auf die Entwicklungsorganisation und die Software abgestimmte Methoden zu den genannten Aktivitäten werden entwickelt.

OWASP SAMM[116] (früher OpenSAMM) ist ein Modell, das von Open Web Application Security Project (OWASP) aufgenommen wurde und jedem Interessierten frei zur Verfügung steht (eine kurze Beschreibung von OWASP weiter unten). OWASP SAMM ist nach vier „Business Functions“ aufgebaut. Jede diese Funktionen besteht aus jeweils drei Säulen mit thematisch gruppierten Aktivitäten.[117]

Die Funktionen sind:[118]

- Governance (Steuerung) mit Säulen: Strategy & Metrics, Policy & Compliance und Educcation & Guidance
- Construction (Entwicklung) mit Säulen: Threat Assesment, Security Requirements und Security Architecture

[115] Sachar., Seite 61
[116] https://www.owasp.org/index.php/OWASP_SAMM_Project
[117] Kai Jendrian, „Sicherheit als Qualitätsmerkmal mit OpenSAMM“, 2012.
[118] Jendrian.

- Verification (Prüfung) mit Säulen: Design Review, Code Review und Security Testing
- Deployment (Inbetriebnahme und Betrieb) mit Säulen: Vulnerability Management, Environment Hardening und Operational Enablement

Jede Säule im Entwicklungsprozess kann durch folgende Reifegrade bewertet werden:[119]

- Reifegrad 0: Keine relevanten Aktivitäten zur Sicherheit
- Reifegrad 1: Es existiert wenig Verständnis und die Maßnahmen zur Sicherheit sind unkoordiniert
- Reifegrad 2: Die Wirksamkeit und Effizienz von Maßnahmen werden kontrolliert verbessert
- Reifegrad 3: Die Sicherheit wird umfassend beherrscht.

Microsoft SDL (Security Development Lifecycle) – Bei Microsoft wird der Entwicklungsprozess mit Sicherheitsprüfungen ergänzt. Ziel ist die Anzahl der Schwachstellen zu reduzieren und die Auswirkungen der verbleibenden Schwachstellen zu minimieren.[120]

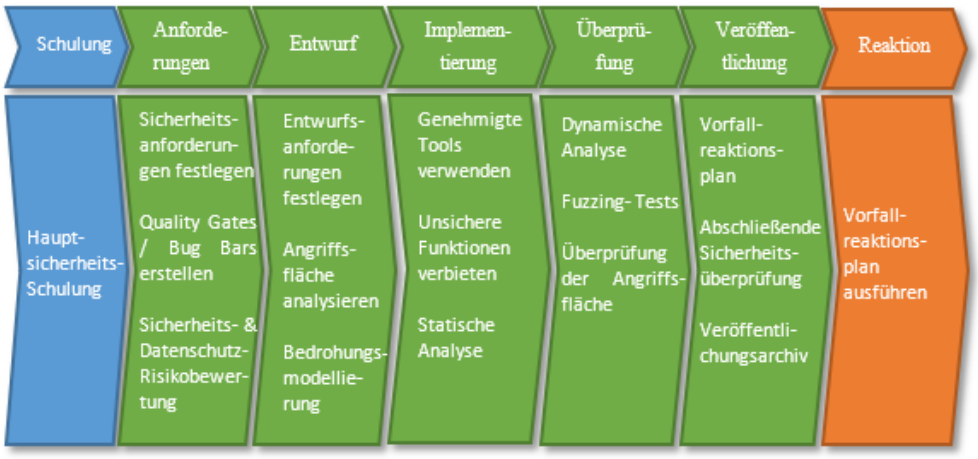


Abbildung 8. Microsoft SDL-Prozess mit seinen Methoden

[119] Jendrian.

[120] Sachar, *Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering*., Seite 62

Die Aktivitäten des Microsoft SDL[121]:
Es existiert ein zentrales Team, der dafür sorgt, dass alle an der Software arbeitenden Mitarbeiter die richtigen Entscheidungen treffen. Um die Angriffsfläche der Software zu verkleinern, wird in der Designphase unbedingt eine Bedrohungsmodellierung durchgeführt. In der Entwicklungsphase werden bestmögliche, anerkannte Techniken eingesetzt. Wo immer es auch möglich ist, werden Werkzeuge und Checklisten als Unterstützung verwendet. Im Rahmen der Tests werden auf jedem Fall die Sicherheitsanforderungen getestet, Angriffsszenarien werden unter Einsatz von Fuzzing durchgespielt und Penetrationstests werden durchgeführt. Mittels „Security Pushes" werden die kritischen Softwareteile auf Sicherheit begutachtet, und die Sicherheitsdokumentation auf den aktuellen Stand gebracht. In der „abschließenden Sicherheitsüberprüfung" hinterfragt ein Expertenteam alle Methoden und Ergebnisse noch einmal, und erst bei erfolgreichen Bestehen kann da Produkt freigegeben werden. Bei der Sicherheits-Reaktion wird nicht nur auf Schwachstellen reagiert, sondern der Input wird genutzt um davon zu lernen und um dieses Wissen für die Entwicklung neuer Produkte zu verwenden.

Open Web Application Security Project (OWASP)[122] – ist eine Informationsplattform für sichere Software mit vielen freiwilligen Mitarbeitern und auch vielen Nutzern, die dort die Informationen nutzen. OWASP ist mit OWASP Top Ten, einer Liste mit häufigsten Schwachstellen bekannt geworden. Neben Top Ten gibt es da eine Sammlung von Tools, Modellen, Lernumgebungen und Literatursammlungen, wie z.B.:

- OWASP Development Guide für Webentwickler
- OWASP WebGoat, online Lernplattform für Hackertechniken
- OWASP WebScarab, Penetrationstest-Tool zum Scannen und Hacken von Webbasierten Anwendungen
- …

Build Security in Touch Points – Diese Methode versucht das Notwendige in wenigen sog. Berührungspunkten mit bestehenden Softwareentwicklungsprozessen zu verbinden. Touch Points sind: Security

[121] Sachar., Seite 63
[122] Sachar., Seite 64f

Requirements Elicitation, Code Review, Risk-Based Security Test, Abuse Cases, Architectural Risk analysis…[123]

Secologic – ist ein Projekt das „10 Goldene Regeln“ für sichere Software entworfen hat. Die Empfehlungen sind stark auf die jeweilige Zielgruppe (Analyst, Architekt, Entwickler, Tester, IT-Produktmanager) abgestimmt. Diese bekommen unterschiedliche Handlungsanweisungen.[124]

2.5.5 Security Building Blocks

Die sicherheitsbezogenen Designentscheidungen sollten in den Softwareentwicklungsprozessen sehr früh im Gesamtprozess getroffen werden. Die übliche Vorgehensweise in meisten Fällen ist aber eine Mischung aus der Analyse der Sicherheitsanforderungen, Ad-hoc-Designentscheidungen, einigen Best Practices, individueller Sicherheitsexpertise. Nachdem die Schwachstellen gefunden wurden, kommt es zu einer schrittweisen Verbesserung des Produktes. Um dieses Problem zu umgehen wurde im Rahmen des SecFuture Projekts[125] ein neuer Security Engineering Prozess entwickelt, der die sicherheitsrelevanten Komponenten, Security Building Blocks, verwendet. Security Building Blocks sind bereits definierte und getestete Softwarekomponenten, die in einer frühen Phase der Softwareentwicklung (schon während der Modellierung des Softwaresystems) genutzt werden können um Security-Design-Entscheidungen zu treffen. Security Building Block Modell wird hauptsächlich zur Entwicklung der mehr oder weniger komplexen eingebetteten Systeme verwendet. [126]

Die Entwicklung der eingebetteten Systeme ist aufgrund ihrer spezifischen Eigenschaften sehr komplex. Viele solche Systeme bestehen aus vielen verschiedenen Komponenten. Informationen werden verarbeitet und von einer Komponente auf die andere weitergeleitet. Eingebettete Systeme haben auch eine reaktive Natur, d.h. wenn sie Informationen verarbeiten, müssen sie oft auf eine spezifische Weise reagieren (z.B. ein anderes System

[123] Sachar., Seite 65f
[124] Sachar., Seite 66
[125] https://www.secfutur.eu/
[126] Andre Rein u. a., „Introducing Security Building Block Models“, 2012.

aktivieren, Informationen senden…). Außerdem haben eingebettete Systeme Echtzeiteigenschaft. Sie müssen in Echtzeit die Informationen, die sie erhalten haben, verarbeiten und auf diese reagieren. Diese Systeme arbeiten mit externen Komponenten wie Sendern, Videokameras, Sensoren oder Ressourcen wie Datenbanken, APIs, usw., d.h. sie verwenden viele Komponenten und Ressourcen als Hardware oder Software. [127]

Security Building Blocks (SBB) sind gekapselte Komponenten. Sie können mit anderen Komponenten, die auch domänenunabhängig sind interagieren. Ein SBB kann nur eine konkrete Implementierung einer Sicherheitslösung sein. Um eine SBB in den Engineering-Prozess zu integrieren, ist eine Beschreibung der SBB (z.B. in Form eines UML-Modells) erforderlich. Ein SBB-Modell repräsentiert eine oder mehrere Implementierungen der SBB. Security Building Blocks sind die Hauptkomponenten jedes Security Building Block Modells. Jedes Artefakt aus dem SBB Metamodell hat mindestens eine direkte Beziehung zu einer SBB. Ein SBB kann aus anderen SBBs bestehen, so kann der Detaillierungsgrad während des SBBM-Entwicklungsprozesses erhöht werden, wenn er benötigt wird. Andererseits können mehrere SBBs verwendet werden um eine komplexere SBB zu bilden.[128]

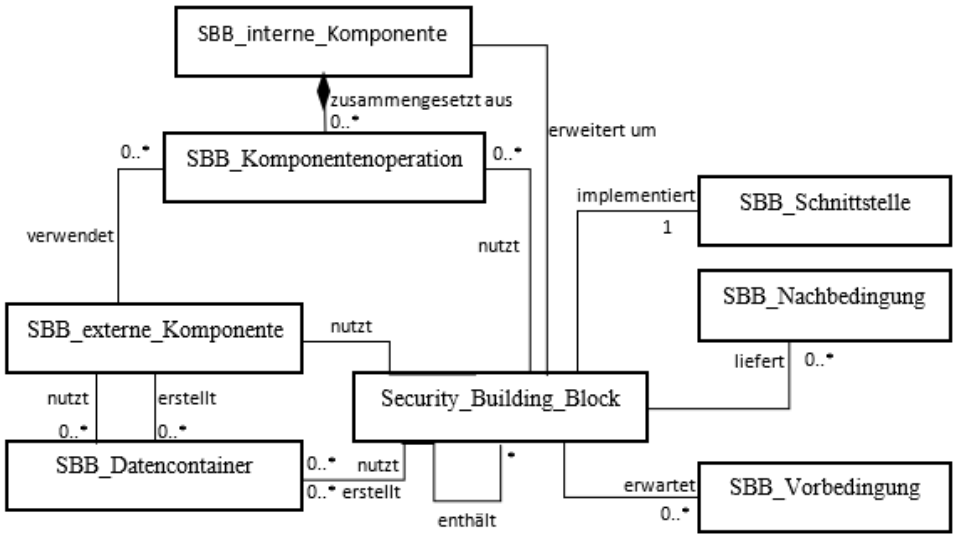


Abbildung 9. Security Building Block Metamodell

[127] Rein u. a.
[128] Rein u. a.

Die Abbildung 8 zeigt Security Building Block Metamodell. Dieser definiert alle Artefakte und Beziehungen eines Security Building Block Modells und wird im Folgenden kurz beschrieben[129]:

SBB_Datencontainer: SBB Datencontainer dient als Container für alle Daten die von einer SBB bearbeitet werden müssen. Er ist der allgemeinste Typ im SBB Metamodell. Ein SBB Datencontainer kann als Ausgabewert eines SBBs oder einer externen Komponente entstehen. Ein SBB kann diesen auch als Eingabewert verwenden.

Security_Building Block: SBB ist die zentrale Komponente und hat Beziehungen zu allen anderen Artefakten:

- erstellt und verwendet die SBB Datencontainer-Werte
- erwartet eine SBB-Vorbedingung
- stellt eine SBB Nachbedingung bereit
- implementiert eine SBB-Schnittstelle, die von einem Systemmodellierer oder anderen SBBs verwendet werden kann
- schließt andere SBBs ein
- verwendet eine SBB externe Komponente
- verwendet eine SBB-Komponentenoperation
- erweitert eine interne SBB-Komponente, um ihren Zweck zu verbessern oder zu ändern

SBB_Vorbedingung: SBB-Vorbedingung stellt eine Anforderung dar, die formell oder informell sein kann. Sie gibt an, unter welchen Bedingungen ein SBB erfolgreich angewendet werden kann.

SBB_Nachbedingung: SBB-Nachbedingung ist eine Aussage, die gültig ist, wenn eine SBB erfolgreich angewendet wird. Vorher muss jede SBB-Vorbedingung eingehalten werden. SBB-Nachbedingung kann auch formell oder informell sein.

SBB_externe_Komponente: Die externe Komponente ist eine Systemkomponente, die außerhalb der Reichweite des aktuellen SBB, manchmal sogar außerhalb der Reichweite des SBB Modells liegt. Sie muss für einen ordnungsgemäßen Betrieb eines SBB verfügbar sein. Ein SBB kann die Funktionalität der externen Komponente direkt oder indirekt (Verwendung der, von ihr erzeugten, Datenstrukturen) verwenden. Eine externe SBB Komponente kann einen SBB-Datencontainer als Eingabewert verwenden. Zusätzlich kann eine externe SBB-Komponente Funktionen

[129] Rein u. a.

einer internen Komponente anwenden, indem sie ihre SBB-Komponentenoperationen verwendet.

SBB_interne_Komponente: Die interne Komponente ist über ihre SBB-Komponentenoperation direkt einem SBB zugeordnet. Jede interne Komponente besteht aus SBB-Komponentenoperationen, die eine konkrete Funktionalität der Komponente darstellen. Ein SBB kann die interne Komponente durch die nicht spezifizierte Verwendung der Komponentenoperationen innerhalb des SBB modifizieren. Zusätzlich kann ein SBB eine interne Komponente erweitern, so dass die Funktionalität der Komponente verbessert wird.

SBB_Komponentenoperation: Die SBB-Komponentenoperation ist eine Operation einer internen Komponente, die von einer SBB ausgeführt werden kann. Dies erfolgt intern in einer SBB und ist meist ein Aufruf einer Funktion oder Methode, die von einer Bibliothek der internen Komponente zur Verfügung gestellt wird.

SBB_Schnittstelle: Die SBB-Schnittstelle kann von einem Systemingenieur, oder anderen SBBs, die den aktuellen SBB integrieren müssen, verwendet werden. Die SBB Schnittstelle dokumentiert die Eingabe- und Ausgabewerte und beschreibt die Funktionalität. Diese Informationen sind für einen Systemingenieur, der SBB in einem konkreten Systemmodell verwenden möchte, notwendig. Die SBB-Schnittstelle wird auch verwendet, wenn SBBs miteinander kombiniert werden.[130]

2.6 Agile Security Requirements Engineering

In der agilen Softwareentwicklung werden User Stories eingesetzt um Anforderungen (auch Sicherheitsanforderungen) zu spezifizieren. Diese werden oft mit Abuser Stories ergänzt, für die Arten von Angriffen, die mit den User Stories nicht beschrieben werden können. Folgender Kapitel beschreibt User und Abuser Stories, ihre Bedeutung für agile Security Engineering, sowie einige existierende Frameworks für Risiko- und Security Management, die in der agilen Softwareentwicklung verwendet werden können.

[130] Rein u. a.

2.6.1 User Stories und Abuser Stories

Eine User Story ist eine kurze Beschreibung einer Anforderung an ein Softwaresystem. User Stories werden häufig in den agilen Softwareprojekten verwendet um die Anforderungen zu beschreiben und sie zu managen. Sie sind unscharf formuliert und lassen viel Raum für Änderungen.[131]

Eine User Story besteht aus folgenden drei Teilen:

- Karte – beschreibt kurz die umzusetzende Anforderung, ohne ins Detail zu gehen. Die Anforderungen werden normalerweise auf A5 Karteikarten geschrieben. Diese lassen sich einfach umsortieren, neu schreiben, oder, wenn sie nicht mehr relevant sind, durchreißen. Für die Anforderungsbeschreibung hat sich folgendes Muster bewährt:
  Als *<Benutzerrolle>* will ich *<das Ziel>* [, so dass *<Grund für das Ziel>*][132]
- Konversation – ist der Schlüssel zu einer guten User Story. Im Rahmen der Konversation bespricht das Team die Details der Story mit dem Product Owner. Auf der Vorderseite der Karte werden diese Details oder die Fragen zu einer User Story notiert. Die Kommunikation zwischen dem Product Owner und dem Team kann so intensiv werden, dass keine Notizen mehr gemacht werden, sondern, dass die besprochenen Details im Sprint unmittelbar umsetzt werden.[133]
- Akzeptanzkriterien – konkretisieren das Ziel der User Story und geben vor, wann die Story fertig ist. Diese werden von Product Owner und dem Team gemeinsam bestimmt und auf die Rückseite der Karte geschrieben.[134] Jede User Story hat eine Reihe von Akzeptanzkriterien, die erfüllt sein müssen, damit die User Story als abgeschlossen oder "erledigt" betrachtet werden kann.[135]

[131] Ralf Wirdemann, *SCRUM mit User Storys* (Carl Hanser Verlag München, 2011)., Seite 49
[132] Wirdemann., Seite 51, 57
[133] Wirdemann., Seite 52
[134] Wirdemann.. Seite 52
[135] Ahola u. a., *Handbook of The Secure Agile Software Development Life Cycle*., Seite 8

Um unterschiedliche Granularitätsebenen auszudrücken, wird zwischen Epics und User Stories unterschieden. Epics sind große User Stories, die wiederum aus Epics, oder aus einer ganzen Reihe von User Stories bestehen. Beim Aufteilen von Epics in User Stories entstehen Themen. Ein Thema gruppiert eine Menge zusammenhängender User Stories nach einem Thema.[136]

Die User Stories werden von den Product Ownern verwendet, um entweder bei einem Projektstart ihr Produkt-Backlog zu füllen, oder sie werden zu einem bestehenden Backlog hinzugefügt.
Die User Stories können auch Sicherheitsanforderungen beschreiben. Viele kleinere Unternehmen haben keinen Zugriff auf die Dienste eines Sicherheitsexperten. So ist es, aufgrund des fehlenden Fachwissens, für den Kunden schwierig die Sicherheitsanforderungen zu beschreiben. Dafür gibt es einige vorgefertigten Listen von User Stories, mit denen diese Sicherheitsanforderungen abgedeckt werden können.[137] Eine solche Liste wurde von Siiskonen, Särs und Vähä-Sipilä als einer der Ergebnisse der Sicherheitsforschung während des Cloud Software Programms in Finnland erstellt. Die Liste beschreibt User Stories, wie z.B. Verfügbarkeit, Malwareschutz, Backup, kryptographische Schlüsselverwaltung usw.[138] Diese Liste bezieht sich auf ein Cloud-basiertes Softwaresystem, und eingebettete Systeme z.B. könnten weitere User Stories erfordern. Auch nicht jede User Story aus der Liste muss auf ein System angewendet werden.[139]

Einige Arten von Angriffen können mit Hilfe von User Stories nicht ausgedrückt werden. Deshalb ist eine Erweiterung von User Stories erforderlich um die Bedrohung zu beschreiben – Abuser Stories.
Abuser Stories beschreiben wie die Angreifer das System missbrauchen und das Vermögen der Beteiligten gefährden könnten. So werden neue Sicherheitsanforderungen identifiziert.
Abuser Stories sollten kurz und informell geschrieben werden. Im Gegensatz zu User Stories sollten die Abuser Stories nicht ausschließlich von Kunden

[136] Wirdemann, *SCRUM mit User Storys*., Seite 57ff
[137] Ahola u. a., *Handbook of The Secure Agile Software Development Life Cycle*., Seite 9
[138] Ahola u. a., Seite 6 u. 72
[139] Ahola u. a., Seite 9

geschrieben werden, sondern gemeinsam mit dem Entwicklungsteam, da einige Bedrohungen von den Entwicklern besser erkannt werden können.[140]

2.6.2 Existierende Frameworks für Risiko- und Security Management in der agilen Softwareentwicklung

Im Folgenden werden einige Modelle beschrieben, die Risiko- und Security Management in der agilen Softwareentwicklung unterstützen.

MSDL (Microsoft Security Development Lifecycle) agile – MSDL agile ist eine agile Version von im Kapitel 2.5.4 beschriebenen Microsoft SDL. Im Gegensatz zu MSDL müssen bei MSDL agile nicht alle Anforderungen für jede Iteration erfüllt sein. Da es bei der agilen Softwareentwicklung keine Entwicklungsphasen (Entwurf, Implementierung, Überprüfung, Veröffentlichung) wie in der klassischen Wasserfall-basierten Entwicklung gibt, wurde MSDL auf agile Entwicklung angepasst.[141]

MSDL agile unterteilt die Anforderungen in folgende Kategorien[142]:

- „One-Time“ Anforderungen – sind abgesehen von der Erstellung eines Basis-Bedrohungsmodells einfache Anforderungen, die nur einmal während der Projektlaufzeit ausgeführt werden. Es müssen nicht alle einmalige Anforderungen im ersten Sprint abgeschlossen werden.

  Die Bedrohungsmodellierung ist die Grundlage eines SDL. Zu Beginn des Projekts wird ein Basis-Bedrohungsmodell erstellt. Bei jedem Sprint wird es dann mit neuen Risiken aus den Teilen des Produkts, die sich in der Entwicklung befinden aktualisiert. So werden potentielle Designprobleme, die die Sicherheit beeinträchtigen könnten identifiziert.
- „Bucket“ Anforderungen – müssen während der Produktlebensdauer mehrmals durchgeführt werden, sie müssen aber nicht bei jedem Sprint vollendet werden. Es gibt drei verschiedene „Bucket“-Kategorien: Überprüfungsaufgaben, Designprüfung und Planung. Jede Kategorie enthält mehrere

[140] Johan Peeters, „Agile Security Requirements Engineering“, 2005.
[141] Ahola u. a., *Handbook of The Secure Agile Software Development Life Cycle*., Seite 33
[142] Ahola u. a., Seite 33f

verschiedene Aufgaben. Bei jedem Sprint muss das Team je eine Aufgabe jedes „Buckets“ ausführen. Es gibt keine bestimmte Reihenfolge in der die Bucket-Anforderungen abgeschlossen sein müssen, es darf aber nicht länger als sechs Monate dauern, ohne dass eine Anforderung angesprochen wird.

- „Every-Sprint“ Anforderungen – sind wichtige Anforderungen, die in jedem Sprint ausgeführt werden müssen, und egal wie lang der Sprint dauert, kann keine Software ausgeliefert werden, ohne dass alle „Every-Sprint“ Anforderungen abgeschlossen worden sind.

Open Web Application Security Project (OWASP) Konferenz – Auf der OWASP-Konferenz 2010 wurde das Produktsicherheitsrisikomanagement in der agilen Umgebung von Vähä-Sipilä eingeführt.[143] Scrum wurde als Beispiel für eine agile Methode verwendet und in drei verschieden Phasen, die gleichzeitig, und nicht in chronologischer Reihenfolge erfolgen unterteilt[144]:

- Requirements Engineering Phase– kann auf zwei Arten umgesetzt werden: als „security stories“, oder als „abuse cases“. „Security stories“ sind Szenarien zu Sicherheitsaspekten, die sehr allgemein sein sollen. „Abuse cases“, oder Missbrauchsfälle sind Szenarien, die nicht auftreten dürfen. Sie werden verwendet, wenn es technisch keine Möglichkeit gibt positive Sicherheitsanforderungen für bestehende Probleme zu erstellen.
- Entwicklungsphase – hier wurden drei verschiedene Ideen vorgeschlagen. Als erstes sollten am Ende jedes Sprints in der Besprechungsphase (Review) geprüft werden ob alle Sicherheitskriterien erfüllt sind. Als zweites Werkzeug sollte in jedem Sprint eine Bedrohungsmodellierung verwendet werden. Das heißt, für jede potenzielle riskante Aufgabe im Product-Backlog, soll eine zusätzliche Aufgabe vorhanden sein, eine Bedrohungsanalyse durchzuführen. Die dritte Idee war, dass das Team eine Checkliste, die typische Probleme in der Vergangenheit oder erwartete Probleme in der Zukunft widerspiegelt, haben sollte um riskante Aufgaben zu identifizieren.

[143] Ahola u. a., Seite 34
[144] Ahola u. a., Seite 34f

- Integration- und Freigabephase - Während dieser Phase trägt der Product Owner die Verantwortung für die Restrisiken. Wenn die Restrisiken zu hoch sind und nicht akzeptiert werden können, muss Product Owner eine neue Bedrohungsmodellierung für sie planen.

Forschung an der Universität Stockholm – Eine Studie der Universität Stockholm beschreibt ein Modell für die Zusammenführung von Risikomanagement mit agilen Praktiken.[145]

Das vorgeschlagene Risikomanagement-Framework hat vier verschiedene organisatorische Ebenen[146]:

- Risikomanagement-Forum ist eine Möglichkeit Risiken zu kommunizieren
- Business Manager ist verantwortlich für die Planung und das Management von Risiken auf Unternehmensebene
- Produktmanager ist ähnlich wie Product Owner im Scrum
- Das Team ist dem Scrum-Team ähnlich.

Die Risikoidentifikation, Bewertung und Managementplanung sollten durch den Product Owner und das Scrum-Team vorgenommen werden, Risikoüberwachung, das Akzeptieren von Restrisiken und nachträgliche Analyse sollte nur vom Team durchgeführt werden.
Der Product Owner ist für die Risiken in der Requirement-Engineering-Phase verantwortlich, er kann das Risikomanagement an eine andere Person (z.B. an jemanden in Scrum-Team) delegieren. Während der Sprintphase ist das Team dafür verantwortlich, neue Risiken zu erkennen und zu managen. In der Studie heißt es, dass der Teamleiter das Risikomanagement in der Implementierungsphase (bei Scrum Sprintphase) überwachen sollte. Da es in Scrum keinen Teamleiter gibt, könnte diese Rolle vom Scrum-Master oder jemanden, der daran interessiert ist, für das Risikomanagement im Projekt verantwortlich zu sein, übernommen werden.
Die Abbildung 10 zeigt das Modell, seine Prozessphasen und die Kommunikationskanäle.

[145] Ahola u. a., Seite 35
[146] Ahola u. a., Seite 35

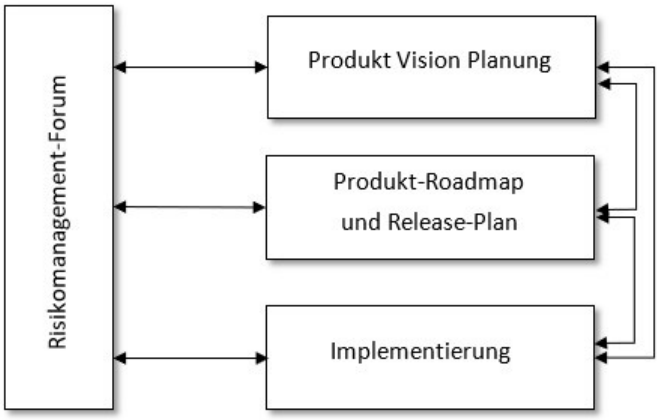


Abbildung 10. Prozessphasen und die Kommunikationskanäle

Dieses Modell wurde wegen mangelnder Agilität, und weil es keine Vorschläge zur Risikoerkennung und Risikobewertung vorlegte kritisiert.[147]

Risk board – Risikoboard ist ein einfaches Whiteboard unterteilt in Kategorien für verschiedene Risikotypen.
Die Kategorien könnten sein: neue Risiken, in Arbeit, abgewendet und ignoriert. Jedes Risiko sollte auf eine farbige Haftnotiz geschrieben werden. Die Farben stellen die Größe des Risikos dar. Die rote Farbe stellt das höchste Risiko dar, gefolgt von gelb und grün. Mit Hilfe von Risikoboard sollen die Risiken überwacht werden und für alle Beteiligten sichtbar sein. [148]

[147] Ahola u. a., Seite 35
[148] Ahola u. a., Seite 36

3 VIEWPOINT INTEGRATION

An der Entwicklung der meisten Großen und komplexen Systemen sind viele Menschen, die verschiedene Sichten haben, beteiligt. Diese Sichten sind durch die Fähigkeiten, Kenntnisse und Verantwortlichkeiten definiert und überschneiden sich, so dass eine Koordinierung erforderlich ist. Da das Wissen innerhalb einer Sicht auf verschiedene Arten dargestellt werden kann, sind diese Überschneidungen nicht offensichtlich. All das wird noch verstärkt, wenn das System verschiedene Technologien einsetzt (Software, Hardware, Mechanik usw.). [149]

Um dieses „multiple perspective problem" zu lösen wurden in den neunziger Jahren in einer Reihe von Forschungsprojekten am Imperial College in London einige Modelle entwickelt, die die Sichten der Beteiligten und somit ihre Anforderungen spezifizieren, analysieren und verwalten.[150]

Im Folgenden werden drei dieser Modelle behandelt: TARA, ViewPoints Framework und FOREST.

3.1 TARA

Das Ziel des TARA-Projekts (Tool Assisted Requirements Analysis) war es, drei wichtige Erweiterungen der CASE-Technologie im Bereich der Anforderungsanalyse zu untersuchen: Methodenunterstützung, Animation und Wiederverwendung.

[149] A Finkelsetin u. a., „Viewpoints: A Framework for Integrating Multiple Perspectives in System Development", *International Journal of Software Engineering and Knowledge Engineering* 2, Nr. 1 (1992): 31–58.
[150] Finkelstein u. a., „Requirements Engineering Through Viewpoints".

Als Basis für dieses Projekt wurde die weit verbreitete Anforderungsanalyse-Methode (CORE) und ein CASE-Tool zur Diagrammkonstruktion und Konsistenzprüfung (The Analyst) verwendet. [151]

3.1.1 CORE und The Analyst

CORE ist eine präskriptive Methode, die aus einer Reihe von Schritten besteht. Diese Schritte stellen die Sicht des Benutzers auf die Leistungen die das geplante System erbringen soll und auf die durch seine Betriebsumgebung auferlegten Beschränkungen dar. CORE bietet Techniken und Notationen für alle Phasen der Anforderungsermittlung, Spezifikation und Analyse von Anforderungen. Die einzelnen Schritte in CORE sollten in einer genau definierten Reihenfolge ausgeführt werden.[152]

Die Schritte in sind:

- Viewpoint-Identifikation – Die möglichen Viewpoints werden identifiziert und in funktionale und nicht-funktionale unterteilt. Die funktionalen ViewPoints werden dann in eine Menge von begrenzten („bounded“) ViewPoints und definierenden („defining“) ViewPoints unterteilt.[153] Die definierende ViewPoints sind Subprozesse des Systems, die begrenzten interagieren indirekt mit dem System.[154]
- ViewPoint-Strukturierung – Die definierenden ViewPoints werden zerlegt und hierarchisch strukturiert; die begrenzten ViewPoints werden auf derselben Ebene wie das Zielsystem platziert.[155]
- Tabellarische Sammlung – Die Informationen über ViewPoints werden in einer Tabelle gesammelt (Quelle, Input, Aktion, Output, Zielort). Tabellarische Sammlungen helfen, Lücken und Konflikte beim Informationsfluss zwischen verschiedenen ViewPoints zu beheben.[156]

---

[151] Anthony Finkelstein und Jeff Kramer, „TARA: Tool Assisted Requirements Analysis“, 1991.
[152] Finkelstein und Kramer.
[153] Gerald Kotonya und Ian Sommerville, „Viewpoints for requirements definition“, *Software Engineering Journal*, 1992.
[154] Yang Zhishang, Zerui Sun, und Shenluo Li, „Viewpoint Based Problem Modelling Technique and Its Application“, *Journal of Software* 9, Nr. 5 (2014).
[155] Kotonya und Sommerville, „Viewpoints for requirements definition“.
[156] Kotonya und Sommerville.

- Datenstrukturierung – Die Ausgabe jedes ViewPoints wird analysiert und zerlegt.[157]
- Einzel-ViewPoint Modellierung und Kombinierte ViewPoint Modellierung – Modelllierung von ViewPoint-Aktionen
- Beschränkungsanalyse wird durchgeführt[158]

Der Analyst ist ein interaktives Softwaretool, das die CORE-Methode unterstützt. Es bietet die Werkzeuge zum Speichern und Präsentieren grafischer CORE-Spezifikationen und enthält eine regelbasierte Konsistenzprüfung. Die syntaktischen Überprüfungen für schlecht geformte Diagramme werden vor dem Speichern durchgeführt, die semantischen (z. B. Datenflüsse ohne spezifizierten Ziel) können entweder auf Aufforderung oder kontinuierlich durchgeführt werden.[159]

3.1.2 Methodenunterstützung

Da es nur wenige echte Experten in einer bestimmten Anforderungsanalyse-Methode gibt, sollte CASE-Technologie um die Methodenunterstützung erweitert werden. Die meisten Methoden sind nicht standardisiert, sondern existieren in unterschiedlichen Versionen. Deshalb ist es wichtig, dass das Methodenmodell für Änderungen zugänglich ist, ohne dass eine Neucodierung erforderlich ist. Methoden existieren nicht nur in verschiedenen Versionen, sondern auch in einer einzelnen Organisation können sie sich im Laufe der Zeit ändern (aufgrund der Erfahrung oder der Anforderungen neuer Anwendungen). Ein wesentlicher Bestandteil der Entwicklung eines Methodenunterstützungstools ist das Konstruieren eines Methodenmodells. Jedes Methodenmodell hat zwei Komponenten: eine normative Komponente (enthält Regeln darüber, was in verschiedenen Situationen zu tun ist) und eine beschreibende Komponente (kodiert das Wissen über die Konzepte, die der Methode zugrunde liegen). Das Tool sollte in der Lage sein, den aktuellen Stand der Anforderungsanalyse aufzuzeigen

[157] Finkelstein und Kramer, „TARA: Tool Assisted Requirements Analysis“.
[158] Finkelstein und Kramer.
[159] Finkelstein und Kramer.

und erklären welche Maßnahmen erforderlich sind, um diese zu vervollständigen.[160]

3.1.3 Animation

Die meisten Anforderungsanalyse-Ansätze sind schwach, wenn es um Spezifikation von Prozessen geht, weil sie das beabsichtigte Verhalten nicht in einer dynamischen Prozessform widerspiegeln können. So wurde die CASE-Technologie um Spezifikationsanimation erweitert um weitere Prozessinformationen bereitzustellen. Die Animation wird hauptsächlich verwendet um Transaktionen, insbesondere die, die kritische Leistungs- oder Zuverlässigkeitsaspekte des vorgeschlagenen Systems beinhalten zu validieren. Die Animation einer Transaktion spielt im Wesentlichen ein Szenario ab, das im späteren System stattfinden kann. So werden oft die Lücken in der Spezifikation gezeigt. Deshalb sollte der Animator in der Lage sein, mit dem Analysetool (dem Analyst) zu interagieren. So kann die Spezifikation korrigiert und modifiziert werden.[161]

Es gibt mehrere Möglichkeiten, wie ein Prozess animiert werden könnte. Finkelstein und Kramer unterstützen zwei Ansätze[162]:

- Transaktionsanimation, wo es keine separaten Aufbau- und Ausführungsphasen gibt. Hier wird jede Aktion sofort ausgeführt, nachdem sie ausgewählt wurde. Die Animation beinhaltet das interaktive Auswählen und Ausführen jeder der Aktionen. Der Benutzer wählt alternative Entscheidungspfade basierend auf dem aktuellen Animationszustand.
- Strategie, bei der die Aufbau- und Ausführungsphase separat ausgeführt werden. Eine Transaktion wird aufgebaut, indem einzeln durch die Spezifikation hindurchgegangen wird, und die Aktionen, die ausgeführt werden, sollen ausgewählt werden. Die Aktionen werden dann der Reihe nach ausgeführt. Diese Strategie wird

[160] Finkelstein und Kramer.
[161] Finkelstein und Kramer.
[162] Finkelstein und Kramer.

verwendet, wenn der Benutzer vorher genau weiß, welche Aktionen in einem bestimmten Szenario involviert sind.

3.1.4 Wiederverwendung

Ziel der Wiederverwendung war Spezifikationsfragmente zu identifizieren und zu charakterisieren, damit die gute Kandidaten zukünftig wiederverwendet werden können. Anschließend sollten die wiederverwendbaren Spezifikationen an die neue Umgebung angepasst werden.[163]

3.1.5 Ergebnisse des Projekts

Zu den Ergebnissen des TARA- Projekts gehören[164] :

- ein Methodenunterstützungssystem für CORE, das mit dem Analysten integriert ist und den Methodenbenutzern unterstützende Hinweise gibt
- der Animator, ein Tool, der die Animation von CORE-Transaktionen unterstützt. Der Animator hat volle Grafikunterstützung für die Erzeugung und Manipulation von Transaktionsdiagrammen
- TRUE, ein Prototyptool – der die Wiederverwendung von Transaktionen unterstützt.

3.2 VIEWPOINTS FRAMEWORK

ViewPoints Framework ist ein Framework, dass die Sichten der Beteiligten strukturiert, organisiert und verwaltet.

[163] Finkelstein und Kramer.
[164] Finkelstein u. a., „Requirements Engineering Through Viewpoints“.

3.2.1 Aufbau

Mit jedem ViewPoint wird eine bestimmte Sicht dargestellt. ViewPoints sind lose verbunden und können räumlich verteilt sein. ViewPoints kapseln Repräsentationswissen, Entwicklungsprozesswissen und Spezifikationswissen über ein System und seine Domäne ein. Dieses Wissen wird in fünf sogenannten Slots beschrieben (siehe Abbildung 7).[165]

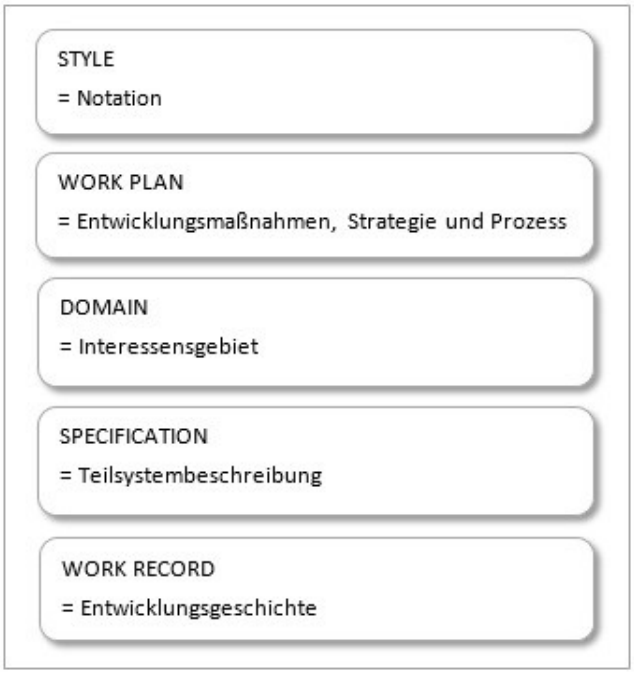


Abbildung 11. Die fünf Slots des ViewPoints

Style Slot beschreibt das vom ViewPoint verwendete Repräsentationsschema.[166] Er besteht aus zwei Teilen: Objekten (Elemente des Style Slots) und Relationen (Beziehungen die zwischen den Objekten bestehen).[167] Z.B. verwendet ein Datenmodellierer bei der Entwicklung einer Datenbank das Entity-Relationsship-Modell. In dem Fall enthält der Style Slot die Beschreibung der graphischen Notation (z.B. die Objekte

[165] Bashar Nuseibeh, Jeff Kramer, und Anthony Finkelstein, „A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification“, 1994.
[166] Nuseibeh, Kramer, und Finkelstein.
[167] Finkelsetin u. a., „Viewpoints: A Framework for Integrating Multiple Perspectives in System Development“.

(Entitäten) werden durch Rechtecke und die Relationen (Beziehungen) durch Rauten beschrieben).[168]

Domain Slot identifiziert den Interessensbereich des ViewPoint in Bezug auf das gesamte System in der Entwicklung. Das ist ein Teilidentifikator oder eine Kennung (Name) eines ViewPoint.[169]

Specification Slot beschreibt die ViewPoint Domäne in der im Style Slot beschriebenen Notation. [170]

Work Plan Slot beschreibt die Vorgehensweise bei der Entwicklung der Beschreibung im Spezifikation-Slot. Er besteht aus vier Teilbereichen[171]:

- ***Assembly-Actions*** sind die grundlegenden Aktionen (wie das Anlegen, Verändern und Löschen), die erforderlich sind um eine Beschreibung in der im Style-Slot angegebenen Notation zu erstellen.
- ***Check-Actions*** prüfen den ViewPoint auf Konsistenz (ob korrektes Wissen dargestellt wird). Das beinhaltet sowohl die interne Korrektheit (***In-ViewPoint-Check-Actions***) als auch die Korrektheit der Abhängigkeiten zwischen den ViewPoints (***Inter-ViewPoint-Check-Actions).***
- ***ViewPoint-Trigger-Actions*** sind Aktionen die im Laufe des Entwicklungsprozesses dynamisch, bei bestimmten Ereignissen ausgeführt werden (wie z.B. das Anlegen neuer ViewPoints, oder das Ausführen von Aktionen, wenn die Inkonsistenzen auftreten.
- ***Guide-Actions*** ist ein Hilfesystem, dass die Anwender bei der Benutzung des ViewPoints unterstützt. Er gibt Hinweise wann etwas zu tun ist (z.B. Konsistenzüberprüfungen), oder wie etwas zu tun ist (z.B. Vorgehen bei Inkonsistenzen).

Work Record Slot – hier werden der Entwicklungsstatus und der Verlauf der ViewPoint -Spezifikation in Bezug auf die Durchgeführten Work Plan Aktionen dokumentiert und verwaltet. Mit Work Record Slot kann die

---

[168] Chrisstian Piwetz, *Anforderungsbeschreibung für Groupware-Systeme - ein sichtenorientierter Ansatz* (Berlin: Logos verlag, 2001)., Seite 72

[169] Nuseibeh, Kramer, und Finkelstein, „A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification".

[170] Nuseibeh, Kramer, und Finkelstein.

[171] Piwetz, *Anforderungsbeschreibung für Groupware-Systeme - ein sichtenorientierter Ansatz*., Seite 73

Rückverfolgbarkeit zu und von Anforderungen erreicht und eine gewisse Form von Entwicklungslogik aufgezeichnet werden.[172]

3.2.2 ViewPoint Templates

ViewPoint Templates unterstützen die Wiederverwendung von ViewPoints indem die fünf Slots in zwei Gruppen unterteilt werden. Eine Gruppe speichert allgemeines Wissen, das vom eigentlichen Inhalt des ViewPoints unabhängig ist. Dazu gehören Style- und Work Plan Slot. D.h., wenn mehrere ViewPoints dieselbe Notation verwenden, sind Style- und Work-Plan Slot bei diesen ViewPoints gleich. Die restlichen drei Slots (Domain-, Spezifikation- und Work Record Slot) werden auf jeden Fall unterschiedlich sein. Bei dem ViewPoint Template sind nur die Slots ausgefüllt, die das anwendungsunabhängige Wissen speichern. So können die ViewPoinnt-Anwender auf das unabhängige Wissen zugreifen, ohne dass sie jeden ViewPoint immer neu vollständig beschreiben. Der ViewPoint-Anwender ergänzt nach der Instanziierung des ViewPoint Templates die restlichen Slots mit dem anwendungsabhängigen Wissen. Durch die Einführung von ViewPoint Templates ergibt sich auch die Aufteilung der ViewPoint-Anwender in Methodenentwickler (erstellt die neue, und verändert die schon bestehende ViewPoint Templates), und Methodenbenutzer (verwendet das Wissen von ViewPoint Templates für die eigentliche Beschreibung eines Systems. Ein Mensch kann sowohl Methodenendwickler, als auch Methodenbenutzer sein.[173]

3.2.3 ViewPoint Integration

Die Integration der ViewPoints hat die Aufgabe sicherzustellen, dass eine Reihe von ViewPoints konsistent zusammenpasst. Dabei sollen sie ihre

[172] Nuseibeh, Kramer, und Finkelstein, „A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification“.
[173] Piwetz, *Anforderungsbeschreibung für Groupware-Systeme - ein sichtenorientierter Ansatz*., Seite 76f

ursprüngliche Struktur beibehalten. Deshalb wird ViewPoint Integration häufig mit dem Inkonsistenzmanagement gleichgesetzt.[174]

Die Beziehungen zwischen den ViewPoints werden verwendet um die Informationen zwischen den ViewPoint Spezifikationen umzuwandeln und zu übertragen und die Konsistenz zu überprüfen. Deshalb ist es notwendig, die Inter-ViewPoint Regeln, die diese Beziehungen beschreiben zu definieren. Dabei soll spezifiziert werden, wie sie angewendet werden sollen und wann sie aufgerufen werden können.[175]

Die Abbildung 8. Zeigt die ViewPoint-Integration Aktivitäten. Ein beschrifteter Pfeil zeigt eine Vorbedingung für den nächsten auszuführenden Schritt an.[176]

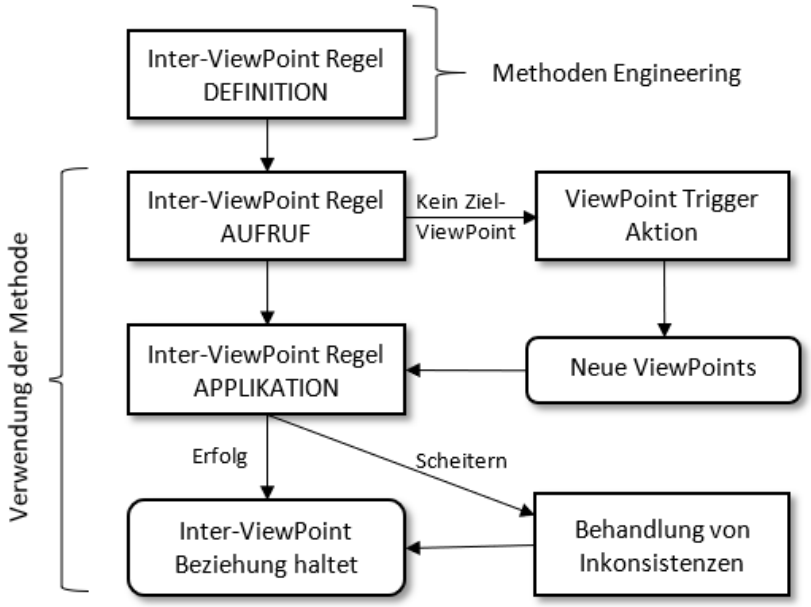


Abbildung 12. ViewPoint Integration Aktivitäten

Schritt 1: Inter-ViewPoint Regel – Definition

Die Inter-ViewPoint Regeln werden in dem Template Workplan-Slot definiert. Sie beschreiben die Beziehungen zwischen den ViewPoints die

[174] Mehrdad Sabetzadeh, Anthony Finkelstein, und Michael Goedicke, „Viewpoints“, 2009., Seite 21
[175] Nuseibeh, Kramer, und Finkelstein, „A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification“., Seite 8
[176] Nuseibeh, Kramer, und Finkelstein., Seite 9

noch nicht erstellt wurden und beziehen sich nicht auf tatsächliche ViewPoints, sondern auf ViewPoint „Typen“.
Sie haben folgende Form:

$$\forall VP_{S,}\ \exists VP_{D,}\ so\ dass\ VP_S\ \Re\ VP_D$$

Wobei VP_S der Quell-ViewPoint ist, in dem die Regel liegt, und VP_D Ziel-ViewPoint mit dem die Beziehung $\Re$ gilt und $VP_S \neq VP_D$. Die Regel gilt für jeden ViewPoint, das von dem Template abgeleitet wird, in dem die Regel definiert ist.[177]

Schritt 2: Inter-ViewPoint Regel – Aufruf

Der Besitzer des ViewPoints ruft die Inter-ViewPoint Regel auf, die sich in dem ViewPoint befindet. Zum Zeitpunkt des Aufrufs soll für jede VP_S mindestens eine VP_D existieren, so dass $VP_S\ \Re\ VP_D$.
Wenn VP_D nicht vorhanden ist, wird eine ViewPoint-Triggeraktion ausgeführt um ein VP_D zu erstellen. Erst dann könnte der Schritt 3 ausgeführt werden.
Inter-ViewPoint Regel – Aufruf identifiziert die ViewPoints, die auf Konsistenz überprüft werden sollen, oder zwischen denen die Informationen übertragen werden sollen.[178]

Es gibt drei Ansätze um den Aufruf der Inter-ViewPoint Regeln zu steuern[179]:

- „constrained“ – die Regeln werden ständig aufgerufen
- „pragmatic“ – Regelaufruf wird vom Benutzer ein- und ausgeschaltet
- „process-oriented“ - ein ViewPoint-Prozessmodell definiert, unter welchen Bedingungen die Regeln aufgerufen werden sollen.

Schritt 3: Inter-ViewPoint Regel – Applikation

Die Inter-ViewPoint Regel – Applikation überprüft die Konsistenz zwischen den zwei ViewPoints. Dabei müssen die interagierenden ViewPoints ein Kommunikationsprotokoll verwenden, in dem die Informationen ausgetauscht und verglichen werden. Dies umfasst in einer verteilten Umgebung die physische Übertragung der Informationen von einem ViewPoint zu einem anderen. Bei der Übertragung sollen die Informationen,

[177] Nuseibeh, Kramer, und Finkelstein., Seite 9
[178] Nuseibeh, Kramer, und Finkelstein., Seite 9
[179] Nuseibeh, Kramer, und Finkelstein., Seite 11

in eine Form, die von dem anderen ViewPoint verstanden wird, umgewandelt werden. Der Mechanismus für eine solche Interaktion muss spezifiziert werden.

Bei der Applikation einer Inter-ViewPoint-Regel existieren die beiden ViewPoints VP_S und VP_D, aber es ist noch nicht bekannt, ob die Beziehung $\Re$ zwischen ihnen besteht. Wenn die Regel (direkt oder nach einer Konfliktlösung) erfolgreich angewendet wurde, führt das zu einer gültigen Beziehung, die zwischen diesen beiden ViewPoints besteht. Die Bestätigung, dass eine Regel zwischen zwei ViewPoints besteht, ist ein zusätzlicher Schritt, der zu einer größeren ViewPoint-Integration führt.[180]

3.2.4 Behandlung der Inkonsistenzen

Die Aufrechterhaltung der Konsistenz in der multiperspektivischen Softwareentwicklung ist nicht immer möglich. Manchmal ist das nicht einmal wünschenswert, weil es zu den unnötigen Einschränkungen im Entwicklungsprozess und zum Verlust wichtiger Informationen führen kann. Deshalb sollten die Wege gefunden werden mit Widersprüchen zu arbeiten. Das kann durch die Behandlung von Inkonsistenzen, in der Regeln explizit spezifiziert werden, die angeben, wie bei Inkonsistenzen vorgegangen werden soll, erreicht werden. Für zwei Teilspezifikationen kann eine gemeinsame Repräsentation gefunden und verwendet werden, um Inkonsistenzen zu identifizieren[181].

Eine mögliche Vorgehensweise wäre[182]:

- Teilspezifisches Wissen in jedem ViewPoint wird in klassische Logik erster Stufe übersetzt
- Logische Inkonsistenzen werden identifiziert
- Regeln für die temporale Logik (Meta-Ebene) werden mit den Inkonsistenzen, die ermittelt wurden, kombiniert, um Aktionen zur Behandlung von Inkonsistenzen zu spezifizieren.

[180] Nuseibeh, Kramer, und Finkelstein., Seite 12
[181] Nuseibeh, Kramer, und Finkelstein., Seite 13
[182] Nuseibeh, Kramer, und Finkelstein., Seite 13

3.2.5 Weitere Integrationsaktivitäten

Obwohl es ideal wäre, ViewPoints als unabhängige Objekte behandeln zu können, gibt es oft Situationen, in denen eine Sammlung von ViewPoints zu einer größeren Spezifikation kombiniert werden muss (z.B. um die Interaktionen zwischen ViewPoints besser zu erkunden, oder um eine einheitliche Perspektive zu erhalten). So kann die Integration von ViewPoints außer Inkonsistenz Management noch drei weitere Aktivitäten umfassen: das Zusammenführen (merging), parallele Komposition (parallel composition) und das Weben (weaving). Im Folgenden werden diese kurz erläutert.[183]

Merging bzw. das Zusammenführen - Das Hauptziel von Merging ist die Vereinheitlichung von Überschneidungen zwischen ViewPoints. Merge kombiniert die ViewPoints so, dass im Ergebnis nur eine Kopie der Teile, die sich überschneiden, enthalten ist. Diese Eigenschaft wird als Nicht-Redundanz bezeichnet und ist eine Grundvoraussetzung für das Zusammenführen. Das Zusammenführen von ViewPoints unterliegt häufig weiteren Kriterien für die Korrektheit.[184]

Die wichtigsten dieser Kriterien sind[185]:

- Vollständigkeit - Merge sollte keine Informationen verlieren, d. H. Es sollte alle Quell-ViewPoints vollständig darstellen.
- Minimalität - Merge sollte keine Informationen einführen, die nicht in den Quell-ViewPoints enthalten sind.
- Semantische Erhaltung - Merge sollte die Expression und Erhaltung von semantischen Eigenschaften unterstützen. Wenn z.B ViewPoints das Verhalten der Zustandsautomaten ausdrücken, ist es von Vorteil ihr zeitliches Verhalten beizubehalten, um sicherzustellen, dass das Zusammenführen die beabsichtigte Bedeutung der Quell-ViewPoints korrekt erfasst.
- Totalität - Merge sollte für jede Gruppe von ViewPoints genau definiert sein, unabhängig davon, ob sie konsistent sind oder nicht.

[183] Sabetzadeh, Finkelstein, und Goedicke, „Viewpoints“., Seite 21f
[184] Sabetzadeh, Finkelstein, und Goedicke., Seite 24
[185] Sabetzadeh, Finkelstein, und Goedicke., Seite 25

Das ist besonders wichtig, wenn man Inkonsistenzen zwischen den Quell-ViewPoints tolerieren möchte.

Diese zusätzlichen Kriterien gelten nicht für alle Probleme beim Zusammenführen von Modellen.
Wenn z.B. Merging eine Konfliktauflösung beinhaltet, können die Kriterien Vollständigkeit und Minimalität unerwünscht sein, da möglicherweise die Informationen hinzugefügt oder gelöscht werden sollen.[186]

Parallel composition bzw. Parallele Komposition – betrifft den Prozess der Kombination einer Reihe von Verhaltensperspektiven, die Verschiedene Komponenten eines Systems haben können. Im Gegensatz zu Merging liegen hier die Beziehungen zwischen den Schnittstellen der ViewPoints und nicht zwischen ihren internen Inhalten. Ein weiterer Unterschied zu Merging ist, dass bei Parallelen Komposition mehrere Kopien desselben ViewPoints beteiligt sein können, weil das System mehrere Komponenten des gleichen Typs aufweisen kann.[187]

Weaving bzw. das Weben – betrifft die Integration von Anforderungen in die aspektorientierte Softwareentwicklung. Die aspektorientierte Softwareentwicklung ist ein Versuch, in komplexen Systemen eine bessere funktionelle Zerlegung zu erreichen. So werden die unterschiedlichen Anliegen in unterschiedliche Systemkomponenten eingekapselt. Es gibt aber einige Anforderungen, die viele Teile des Systems durchdringen und sich nicht auf eine Komponente beschränken können (ein Beispiel dafür ist die Protokollierung, die alle protokollierten Aktivitäten in einem System betrifft). Diese „cross-cutting concerns“ bzw. Querschnittsanforderungen werden dann mittels Weben in das Basissystem einbezogen. Abhängig von der Art des Basissystems kann das Weben statisch (zur Kompilierungszeit) oder dynamisch (zur Laufzeit) durchgeführt werden. Das Weben kann zu unerwünschten Nebenwirkungen führen. So kann es dann erforderlich sein, die automatisierte Analysetechniken zu verwenden, um

[186] Sabetzadeh, Finkelstein, und Goedicke., Seite 25f
[187] Sabetzadeh, Finkelstein, und Goedicke., Seite 26ff

sicherzustellen, dass das Webergebnis Korrektheitseigenschaften, die notwendig sind, erfüllt.[188]

3.3 FOREST

Das Forest Projekt befasste sich mit der Anforderungsspezifikation von großen eingebetteten Echtzeitsystemen. Eine Anforderungsspezifikation ist ein sehr komplexes Dokument, der viele Rollen und viele verschiedene Arten von Informationen beinhaltet. Der Kern dieses Dokuments ist ein Systemmodell, das die Anforderungen der User identifiziert. [189]

Bei der Identifikation der Anforderungen der Echtzeitsysteme müssen zeitliche Aspekte berücksichtigt werden[190]:

- Es gibt Ereignisse die zu festen Zeiten in Echtzeit innerhalb einer Toleranz stattfinden
- Bei manchen Sicherheitsproblemen müssen dringende Korrekturmaßnahmen innerhalb enger Zeitgrenzen stattfinden
- Es gibt Zeiten in denen keine Ereignisse stattfinden dürfen
- Viele Komponenten im Echtzeitsystem treten gleichzeitig auf

Bei den Echtzeitsystemen muss nicht nur das funktionale Verhalten jeder Komponente (Was tut die Komponente?), sondern auch kausale Aspekte (Welche Umstände verursachen dieses Verhalten?), identifiziert werden.
Der FOREST-Ansatz stellt eine Methode zur Verfügung, die Bestimmung der oben genannten Aspekte bei der Anforderungserhebung unterstützt und bietet ein formales Repräsentationsschema um diese Aspekte auszudrücken.[191]

Die Grundlage des FOREST Ansatzes bildet die formale Spezifikationssprache Modell Action Logic M[A]L, und die Methode Structured Common Sense (SCS), die den Prozess, mit dem eine M[A]L-Spezifikation konstruiert wird, leitet und organisiert.[192]

---

[188] Sabetzadeh, Finkelstein, und Goedicke., Seite 29
[189] S J Goldsack und A C W Finkelstein, „Requirements engineering for real-time systems“, 1991.
[190] Goldsack und Finkelstein.
[191] Goldsack und Finkelstein.
[192] Finkelstein u. a., „Requirements Engineering Through Viewpoints“.

3.3.1 Modell Action Logic M[A]L

M[A]L nutzt in erster Linie die Prädikatenlogik erster Stufe um die Eigenschaften eines Systems zu beschreiben. Um alle Aspekte der oben beschriebenen Anforderungen abzudecken, wird die Grundlogik wie folgt erweitert:[193]

- Agenten/Aktionen
- Deontische Ausdrücke
- Zeitoperatoren
- Kombination von Aktionen

Im Folgenden, jeweils eine kurze Beschreibung.

Agenten /Aktionen[194] – Das System besteht aus interagierenden Agenten, die Aktionen unter Verwendung von Vor- und Nachbedingungen ausführen. Die Konsequenzen der Aktionen werden als Systemstatus definiert. Angenommen ag ist der Agent, ac die Aktion und β ist eine Formel die gilt nachdem ag ac ausgeführt hat. Das kann durch folgende Formel ausgedrückt werden:

$$[ag, ac]\beta$$

Eine Vorbedingung α, die erfüllt ist, bevor eine Aktion stattfindet, wird wie folgt ausgedrückt:

$$\alpha \rightarrow [ag, ac]\beta$$

Deontische Ausdrücke[195] – Um ein Systemverhalten vollständiger zu beschreiben, verwendet M[A]L zwei Operatoren, die die Zustände definieren, in denen Aktionen stattfinden können oder müssen. So werden auch die kausalen Beziehungen zwischen den Aktionen ausgedrückt.

Wenn folgende Formel in einem Zustand gilt, ***kann*** Agent seine Aktion durchführen:

[193] Goldsack und Finkelstein, „Requirements engineering for real-time systems“.
[194] Goldsack und Finkelstein.
[195] Goldsack und Finkelstein.

$$per(ag, ac)$$

Wenn folgende Formel in einem Zustand gilt, ***muss*** Agent seine Aktion durchführen:

$$obl(ag, ac)$$

Zeitoperatoren[196] – Da die Aktionen in Zeitintervallen stattfinden und Bedingungen während Perioden gelten, wurde die Prädikatenlogik durch das Merkmal einer zeitlichen Intervalllogik erweitert.

Angenommen $t1$ und $t2$ stellen Zeitintervalle dar. Folgende Beziehungen zwischen den Zeitintervallen sind definiert:

$t1 < t2$	$t1$ geht $t2$ vollständig voraus.
$t1\ [t2]$	$t1$ überlappt $t2$, d.h. $t1$ beginnt zuerst, endet aber vor Ende von $t2$.
$t1 \supset t2$	$t2$ ist vollständig in $t1$ enthalten.
$t1 = t2$	$t1$ und $t2$ sind das gleiche Intervall.

Da die Aktionen auch in Zeitintervallen stattfinden, wurden weitere Operatoren eingeführt.

Folgende Formel gilt für jedes Intervall, das genau einem Intervall entspricht in dem der Agent seine Aktion ausführt:

$$occurrence(ag, ac)$$

Eine weitere Formel gilt zu allen Zeitpunkten, die innerhalb eines der Intervalle liegen die durch die Formel $occurrence(ag, ac)$ definiert sind:

$$happening(ag, ac)$$

Zeitoperatoren können um deontische Ausdrücke erweitert werden. Sicherheitsbedingungen werden oft durch die Kombination von zeitlichen und deontischen Aspekten ausgedrückt.

Wenn ein Agent ag seine Aktion ac vor dem Ende eines Intervalls in dem α wahr ist, abschließen muss, kann das wie folgt ausgedrückt werden:

[196] Goldsack und Finkelstein.

$$obl(ag, ac, \alpha)$$

Kombination von Aktionen[197] – Aktionen werden auf verschiedene Arten kombiniert um komplexere Aktionen zu bilden. Die Kombination von Aktionen kann wie folgt ausgedrückt werden:

$a; b$ Kombination von a und b, wobei zuerst a und dann b ausgeführt wird.

$a + b$ Nichtdeterministische Wahl von a oder b.

$a||b$ Beide Aktionen werden ausgeführt, egal welche Reihenfolge. Sie können auch parallel ausgeführt werden.

3.3.2 Structured Common Sense SCS

Um eine Spezifikation zu erstellen müssen einige Fragen über Agenten (Welche Agenten gibt es? Warum diese Agenten?), Aktionen (Welche Aktionen werden durchgeführt? Wann? Welche Vorbedingungen und Nachbedingungen gibt es? Warum?) und Klassen (Welche Klassen gibt es? Welche Attribute und Beziehungen?) beantwortet werden.
Structured Common Sense (SCS) versucht den Prozess mit dem diese Fragen beantwortet werden systematisch zu steuern und zu organisieren.[198]

SCS besteht aus einer Anzahl von verschiedenen Schritten, von denen einige parallel und einige sequentiell durchgeführt werden. Ein Arbeitsplan bestimmt den Fortschritt durch SCS. Mit jedem Schritt sind grafische Darstellungen und Heuristiken verknüpft.
Die Schritte in SCS sind:

- Identifikation der Agenten im System
- Physische Datenfluss-Analyse – Bestimmen der Verbindungen und Interaktionen zwischen den Agenten und Aufrufen der Aktionstabellen
- Aktionstabelle und Aktionsbeschreibung – Identifikation von Aktionen und eine vorläufige Beschreibung in der natürlichen

[197] Goldsack und Finkelstein.
[198] Goldsack und Finkelstein.

Sprache. Dieser Schritt entspricht dem CORE Schritt Tabellarische Sammlung (siehe Kapitel 3.1.1).

- Datenzerlegung und Analyse der Entity-Relationship Attributen - Identifizierung von Arten, Prädikaten und Funktionen
- Berechtigungs- und Verpflichtungsanalyse – Bestimmung der Kausalitäten zwischen den Aktionen, und Konstruktion von Axiomen.
- Temporale Analyse – Bestimmung der Beziehungen zwischen Aktionen, die zeitliche Intervalle nutzen.[199]

[199] Goldsack und Finkelstein.

4 KOOPERATIVE TECHNOLOGIEN

Die Softwareentwicklung ist wegen des zunehmenden Umfangs der zu erstellenden Systeme sowie der Komplexität immer mit vielen Projektmitarbeitern mit unterschiedlichen Aufgaben verbunden. Der Entwicklungsprozess findet häufig sowohl innerhalb eines Projektteams als auch zwischen mehreren spezialisierten Arbeitsgruppen statt, und ist ein zeitlich und räumlich verteilter kooperativer Arbeitsprozess. Während des gesamten Entwicklungsprozesses besteht Kommunikations- und Koordinationsbedarf.[200]

Im folgenden Abschnitt werden wichtige Begriffe der kooperativen Technologien definiert und einige Arten der Klassifikation von Groupware beschrieben.

4.1 BEGRIFFE

Computerunterstütztes kooperatives Arbeiten wird wie folgt definiert:

> >> Computer Supported Cooperative Work (CSCW) ist die Bezeichnung des Forschungsgebietes, welches auf interdisziplinärer Basis untersucht, wie Individuen in Arbeitsgruppen oder Teams zusammenarbeiten und wie sie dabei durch Informations- und Kommunikationstechnologie unterstützt werden können.
>
> Ziel aller Bemühungen im Gebiet CSCW ist es, unter Verwendung aller zur Verfügung stehenden Mittel der Informations- und Kommunikationstechnologie, Gruppenprozesse zu unterstützen und

[200] Josef Altmann, *Kooperative Softwareentwicklung: Rechnerunterstützte Koordination und Kooperation in Softwareprojekten* (Linz: Universitätsverlag Rudolf Trauner, 1999)., Seite 1

dabei die Effektivität und Effizienz der Gruppenarbeit zu erhöhen. <<[201]

Groupware, ein weiterer Begriff im Gebiet der CSCW wird wie folgt definiert:

>> Groupware are computer-based systems that support groups of people engaged in a common task (or goal) and that provide an interface to a shared environment. <<[202]

>> Groupware bzw. CSCW-Applikationen sind aus Software und eventuell spezifischer Hardware bestehende Systeme, durch die Gruppenarbeit unterstützt oder ermöglicht wird. <<[203]

Workflow Management wird von Workflow Management Coalition als eine Ganz-, oder Teilautomatisierung eines Geschäftsprozesses bezeichnet. Dabei werden die Dokumente, Informationen oder Aufgaben nach festgelegten Verfahrensregeln, von einem Teilnehmer an einen anderen übergeben.[204] Die Prozesse, die durch Workflow Management unterstützt werden haben eine hohe Wiederholungfrequenz und sind sehr strukturiert.[205]

Workgroup Computing – im Gegensatz zu Workflow Management unterstützt Workgroup Computing die Kooperation von Personen, die in Gruppen arbeiten und meist unstrukturierte Aufgaben zu erfüllen haben, die eine niedrige Wiederholungsfrequenz haben.[206]

Kommunikation wird als die Verständigung mehrerer Personen untereinander definiert.
Unter **Koordination** wird jene Kommunikation verstanden, die notwendig ist, um die aufgabenbezogenen Tätigkeiten im Rahmen der Gruppenarbeit abzustimmen.
Die Kommunikation, die zur Koordination und zur Festlegung gemeinsamer Ziele notwendig ist, wird als **Kooperation** bezeichnet.[207]

---

[201] Stephanie Teufel u. a., *Computerunterstützung für die Gruppenarbeit* (Bonn: Addison-Wesley, 1995)., Seite 17
[202] C.A. Ellis, S.J. Gibbs, und G.L. Rein, „Groupware: Some Issues and Experiences", *Communication of the ACM* 34, Nr. l (1991).
[203] Teufel u. a., *Computerunterstützung für die Gruppenarbeit*., Seite 22
[204] Workflow Management Coalition, „Terminology & Glossary", 1996.
[205] Piwetz, *Anforderungsbeschreibung für Groupware-Systeme - ein sichtenorientierter Ansatz*., Seite 17
[206] Teufel u. a., *Computerunterstützung für die Gruppenarbeit*., Seite 209
[207] Teufel u. a., Seite 12

Während bei der Kooperation das gemeinsame Ziel in unterschiedliche Teilaufgaben aufgeteilt ist, für die jeweils eine Person, oder eine Gruppe von Personen verantwortlich ist, trägt bei der **Kollaboration** jeder gleichermaßen zur Verfolgung des gemeinsamen Ziels bei. D.h. die Aufgaben der Beteiligten werden nicht explizit definiert, sondern ergeben sich dynamisch während des Arbeitsprozesses.[208]

4.2 KLASSIFIKATION VON GROUPWARE

Raum-Zeit-Matrix ordnet die Groupware Lösungen nach Zeit und Ort. Hier ist es von Bedeutung, ob sich die Gruppenmitglieder, die durch das Kooperative System unterstützt werden sollen, am gleichen Ort oder an verschiedenen Orten befinden, und ob sie gleichzeitig, oder zeitlich versetzt interagieren. Einige umfangreiche Groupware Systeme können mehrere Quadranten gleichzeitig abdecken. Die Abbildung 5. Zeigt den Raum-Zeit-Matrix.[209]

Abbildung 13. Raum-Zeit-Matrix

[208] Jan Sebastian Schmalz, „Zwischen Kooperation und Kollaboration, zwischen Hierarchie und Heterarchie: Organisationsprinzipien und -strukturen von Wikis“, *kommunikation @ gesellschaft* 8 (2007): 1–21.
[209] Tom Gross und Michael Koch, *Computer-Supported Cooperative Work* (München: Oldenburg Verlag, 2009)., Seite 49f

Die Applikationen können unabhängig von der Technologie nach **Unterstützungsfunktionen** gliedert werden. Die Prozesse der Gruppenarbeit die unterstützt werden sind Kommunikations-, Koordinations- und Kooperationsprozesse.[210]

Auch bei der Klassifikation nach der Unterstützungsfunktionen können die meisten CSCW-Applikationen nicht eindeutig einem der oben genannten Bereiche zugeordnet werden. Die Abbildung 6. zeigt die Positionierung verschiedener Applikationen, je nach Schwergewicht der Betonung auf jeweilige Unterstützungsfunktion.[211]

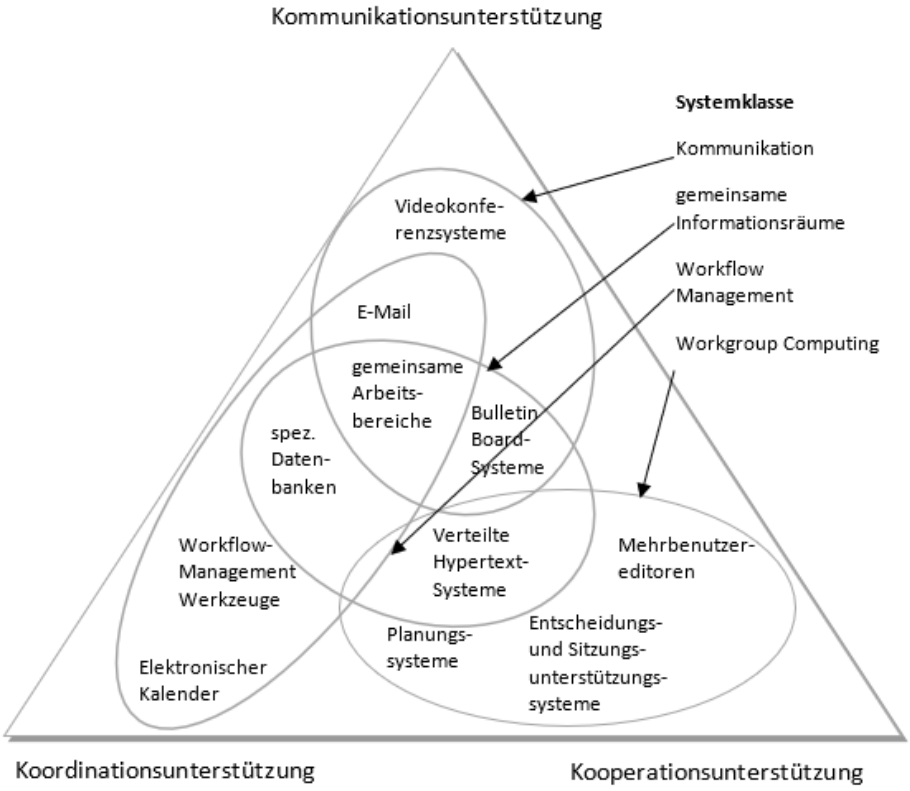


Abbildung 14. Klassifikation nach Unterstützungsfunktionen

[210] Teufel u. a., *Computerunterstützung für die Gruppenarbeit*. , Seite 26
[211] Teufel u. a., Seite 27

5 Entwicklung des ViewPoint Integration-basierten Modells

Für die Entwicklung der Modellierungsmethode wurde die von OMiLAB empfohlener OMI Lifecycle verwendet. OMI Lifecycle ist ein iterativer Prozess, der aus fünf Phasen besteht[212]:

- Creation – in dieser Phase werden die Ziele und die Anforderungen der Modellierungsmethode definiert.
- Design – diese Phase spezifiziert das Metamodell, die Sprachgrammatik, graphische Darstellung und Funktionalität.
- Formalization – hier wird das Ergebnis der vorherigen Phase formalisiert. Es wird sichergestellt, dass die Beschreibung nicht zweideutig ist. Ziel dieser Phase ist, die Ergebnisse innerhalb einer Community teilen zu können.
- Development – in dieser Phase wird das Modell-Prototyp entwickelt.
- Deployment / Validation – in dieser Phase wird das Modell-Prototyp den Benutzern zur Validierung bereitgestellt. Die Rückmeldung und die möglichen geänderten Anforderungen, die sich aus der Erfahrung der Benutzer ergeben können, sind der Ausgangspunkt für die nächste Iteration.

[212] Dimitris Karagiannis, Heinrich C. Mayr, und John Mylopoulos, *Domain-Specific Conceptual Modelling: Concepts, Methods and Tools* (Springer International Publishing Schwitzerland, 2016)., Seite 64f

5.1 CREATION

Ziel der zu entwickelnden Modellierungsmethode ist die ViewPoint-Integration basierte Security Requirements Spezifikation zu unterstützen. Da die Entwicklung von großen und komplexen Systemen mit vielen Rollen verbunden ist, sollen die Rollen und ihre Aktivitäten im kooperativen Prozess der Anforderungsspezifikation modelliert werden können. Die unterschiedlichen Sichten, die durch unterschiedliche Verantwortlichkeiten, Fähigkeiten und Wissen entstehen, und die sich eventuell überschneiden, sollen koordiniert werden können. Die komplexen Anforderungen sollen in einfache Anforderungen zerlegt werden können.

5.2 DESIGN

Als Basis für die Entwicklung des Modells wurde das Metamodell für kooperative Prozesse von Selmin Nurcan und Colette Rolland genommen. Die Abbildung 15. zeigt dieses Metamodell, das im Folgenden beschrieben wird.

5.2.1 Beschreibung des Metamodells

Das zentrale Konzept dieses Modells ist das Konzept des Kontextes. Ein Kontext besteht aus einer Situation und einer Entscheidung, die in dieser Situation getroffen wird. Eine Situation kann verschiedene Granularitätsstufen haben. Eine Entscheidung muss immer mit der Situation in Verbindung gebracht werden, in der sie gilt und drückt aus, was der Benutzer erreichen möchte (das Ziel). Eine Situation kann mit mehreren Entscheidungen verbunden sein. Ein Kontext ist an eine Rolle gebunden. Diese ist in einzelne Rolle und Gruppenrolle klassifiziert. Gruppenrolle enthält mehrere einzelne Rollen.
Die Entscheidungen haben unterschiedliche Konsequenzen, so werden die verschiedenen Kontexte entsprechend ihrer Konsequenzen im Metamodell in ausführbare Kontexte, Plankontexte und Auswahlkontexte klassifiziert.[213]

[213] Nurcan und Rolland, „Meta-modelling for cooperative processes“.

Ausführbarer Kontext – Beim ausführbaren Konzept wird das Ziel einer Entscheidung durch eine Aktion realisiert, so ist im Meta-Modell ein ausführbareres Konzept mit einer Aktion verknüpft. Aktionen werden in zwei Typen klassifiziert: einzelne Aktion und Konversationsaktion. Einzelne Aktion wird weiter in einfache Aktion und komplexe Aktion unterteilt. Eine Komplexe Aktion besteht aus mehreren Aktionen. Einfache Aktion führt die Transformation des Systems durch, so dass sie die neuen Komponenten erstellt, oder, die schon vorhandenen aktualisiert oder löscht. Konversationsaktion erzeugt die Nachrichten. Daher wird das Produkt in System und Nachricht klassifiziert. Eine Nachricht kann mehrere Systemkomponenten betreffen. Eine einzelne Aktion wird von einer einzelnen Rolle ausgeführt. Eine Konversationsaktion ist einer Gruppenrolle zugeordnet und erzeugt mehrere Nachrichten, wobei jede Nachricht von einer einzelnen Rolle erzeugt wird, die in dieser Gruppenrolle enthalten ist. Aus jeder Konversationsaktion können neue Kontexte entstehen, die entweder Plankontexte, Auswahlkontexte, oder die ausführbaren Kontexte sein können. Die ausführbaren Kontexte können wiederum neue Kontexte erzeugen.[214]

Plankontext – Mit Hilfe von Plankontext kann ein Kontext, der als ein komplexes Problem angesehen wird in eine Reihe von Teilproblemen zerlegt werden. Ein Plankontext ist ein Mittel um das Granularitätsproblem zu lösen. Jeder Teilkontext besteht aus einer Untersituation und einer Unterentscheidung, die über diese Untersituation getroffen wird. Die Teilkontexte können von jedem Typ sein (ausführbarer Kontext, Plankontext oder Auswahlkontext). Die Reihenfolge der Kontexte innerhalb eines Plans wird durch das Vorrangdiagramm definiert. Vorranglinks definieren entweder die möglichen geordneten Übergänge zwischen Kontexten oder ihre mögliche parallele Umsetzung. Einem Link kann ein Auswahlkriterium, ein Kriterium, dass definiert, wann der Übergang durchgeführt werden kann, zugewiesen werden.[215]

Auswahlkontext – Ein Auswahlkontext wird verwendet, wenn es mehrere alternative Möglichkeiten gibt, eine Entscheidung zu treffen. Die Situation erfordert hier die Untersuchung von Alternativen bei der

[214] Nurcan und Rolland.
[215] Nurcan und Rolland.

Entscheidungsfindung. Jede Alternative ist eine Strategie, die für die Lösung des Problems verfolgt werden könnte. Ein Auswahlkontext bietet eine Auswahl aus einer Reihe von Strategien. Die verschiedenen Alternativen eines Auswahlkontextes werden im Prozess- Metamodell in der alternativen Beziehung dargestellt. Sie sind mit Auswahlkriterien verknüpft. Ein Auswahlkriterium ist eine Kombination von Argumenten, die eine Alternative eines Auswahlkontexts unterstützen oder ablehnen. Ein Auswahlkriterium kann auch Prioritätsregeln bereitstellen, um abhängig von den Argumenten eine Alternative unter mehreren auszuwählen. Mittels Auswahlkontext kann auch das Granularitätsproblem behandelt werden. Die Alternativen eines Auswahlkontextes sind auch Kontexte, so können Kontexte eine alternative Beziehung teilen. Das führt zu alternativbasierten Hierarchien von Kontexten und erlaubt die Verfeinerung von grobkörnigen Entscheidungen in feinkörnigere.[216]

[216] Nurcan und Rolland.

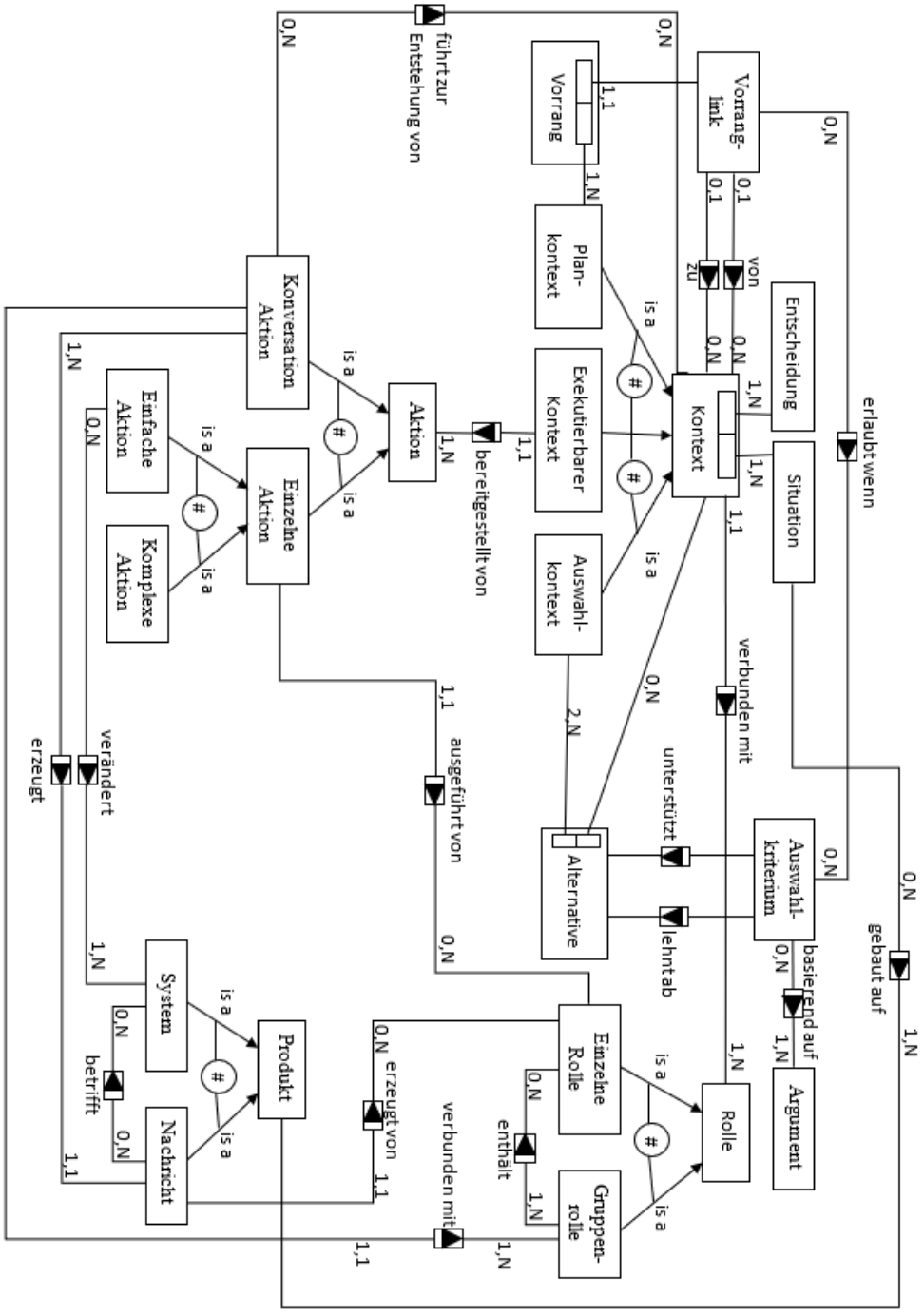
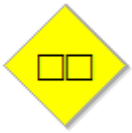


Abbildung 15. Metamodell für kooperative Prozesse[217]

[217] Nurcan und Rolland.

5.2.2 Graphische Darstellung

Die folgende Tabelle zeigt die graphische Darstellung der einzelnen Klassen mit jeweils kurzer Beschreibung und den dazugehörigen Attributen:

Klasse	Beschreibung und Attribute
 Kontext	Kontext besteht aus einer Situation und einer Entscheidung, die in dieser Situation getroffen wird. Es gibt drei Arten von Kontext: ausführbarer Kontext, Plankontext und Auswahlkontext. Attribute: Name, Situation_Decission
 Plankontext	Mittels Plankontext werden komplexe Kontexte in Teilkontexte zerlegt. Attribute: Name, Situation_Sub-issue Vererbte Attribute: Situation_Decision
 Ausführbarer Kontext	Ausführbarer Kontext realisiert das Ziel einer Entscheidung durch eine Aktion. Attribute: Name Vererbte Attribute: Situation_Decision
 Auswahlkontext	Auswahlkontext hilft bei mehreren möglichen Alternativen eine Entscheidung zu treffen. Attribute: Name Vererbte Attribute: Situation_Decision
 Vorrang	Vorrang definiert die Reihenfolge der Teilkontexte innerhalb eines Plankontextes. Attribute: Name, Situation_Decision
 Vorrang Link	Vorrang Link definiert entweder die möglichen geordneten Übergänge zwischen den Teilkontexten eines Plankontextes, oder ihre parallele Umsetzung. Attribute: Name, Transition, Description
 Alternative	Alternative beinhaltet alle möglichen Alternativen eines Auswahlkontextes. Attribute: Name, Description, Situation_Alternative
 Auswahlkriterium	Auswahlkriterium wählt eine Alternative unter mehreren mittels Argumenten, die eine Alternative eines Auswahlkontextes unterstützen oder ablehnen. Er enthält auch Prioritätsregeln und definiert wann beim Vorrang Link der Übergang durchgeführt werden kann. Attribute: Name, Description, Priority_Rules

 Argument	Argument hilft eine Alternative unter mehreren auszuwählen. Attribute: Name, Description
 Gruppenrolle	Gruppenrolle besteht aus mehreren einzelnen Rollen, und führt Konversationsaktionen durch. Attribute: Name Vererbte Attribute: Aufgabe, Absicht
 Einzelne Rolle	Einzelne Rolle führt einzelne Aktionen durch. Attribute: Name Vererbte Attribute: Aufgabe, Absicht
 Konversation Aktion	Konversation Aktion erzeugt mehrere Nachrichten, was zur Entstehung neuer Kontexte führt. Attribute: Name
 Einfache Aktion	Einfache Aktion führt die Transformation des Systems durch (d.h. erzeugt, ändert oder löscht die Anforderungen). Attribute: Name, Artefact_transformation, Description
 Komplexe Aktion	Komplexe Aktion besteht aus mehreren einfachen Aktionen. Attribute: Name
 Nachricht	Nachricht wird mittels Konversationsaktion erzeugt. Attribute: Name, Message
 Systemkomponente/ Artefakt	Systemkomponente/ Artefakt repräsentiert die statische Komponente des Informationssystems, in unserem Fall die Anforderung Attribute: Name, Description

Tabelle 1. Klassen und ihre Attribute

5.2.3 Vorgehensweise beim Modellieren

Nachdem die Grupperollen und die individuellen Rollen modelliert wurden und ihre Attribute (Aufgabe und Absicht) eingetragen wurden kann mit der Modellierung des Kontextes begonnen werden. Da nur die einfache Aktion eines ausführbaren Kontextes ein Artefakt (eine Sicherheitsanforderung) erstellen kann, müssen alle anderen Kontextarten so lange verfeinert bzw. behandelt werden bis diese auch entsteht. Plankontext muss z.B. in Teilkontexte aufgeteilt und in einer durch Auswahlkriterium bestimmter Reihenfolge abgearbeitet werden. Sind die Teilkontexte ausführbare Kontexte, die mit einer einfachen Aktion ausgeführt werden, so entstehen die Artefakte. Gibt es ein oder mehrere Teilkontexte, die eine andere Art von Kontexten darstellen, so müssen sie weiterbearbeitet werden. So auch bei Auswahlkontext – ist die Alternative kein ausführbarer Kontext, der mit einer einfachen Aktion ausgeführt wird, muss diese dementsprechend weiterbearbeitet werden. Handelt es sich bei einer Aktion um eine Konversationsaktion führt das zur Entstehung neuer Kontexte, die auch entsprechend ihrer Art bearbeitet werden müssen.

Wenn es keine Kontexte mehr gibt, die behandelt werden sollen, ist die Anforderungsspezifikation fertig.

5.3 DEVELOPMENT

Für die Entwicklung der Modellierungsmethode wurde Metamodellierungsplattform ADOxx verwendet. Unter der Verwendung der ADOxx kann eine Modellierungsmethode realisiert werden, die neben einer Modellierungssprache, auch aus einem Modellierungsverfahren in Form von Mechanismen und Algorithmen besteht.[218]

Folgende Konzepte aus einer implementierungsperspektive sind in ADOxx von Bedeutung[219]:

[218] „The ADOxx Metamodelling Platform - ADOxx.org“, zugegriffen 10. Juni 2018, https://www.adoxx.org/live/home.
[219] „The ADOxx Metamodelling Platform - ADOxx.org“.

- Klassen und Beziehungen – es gibt sowohl vordefinierte (Metamodell-) Klassen, als auch selbstdefinierte Klassen, die Realisierung eines konkreten Metamodells erlauben. Vordefinierte Klassen sind abstrakte Klassen, die verwendet werden können, um die vordefinierte Syntax und die Attribute für selbstdefinierte Klassen zu erben.

  Beziehungen bzw. relationale Klassen werden durch ihre Quell- und Zielklasse, ihre Kardinalität und ihre Attribute definiert.
- Attribute – definieren bestimmte Eigenschaften von Klassen und relationalen Klassen. Jedes Attribut hat einen Namen, einen Typ und einen Wert.
- Spezielle Attribute – dazu gehören z.B. GRAPHREP (enthält die graphische Notation einer Klasse oder einer relationalen Klasse), ATTREP (steuert die Verfügbarkeit und Struktur eines ADOxx Notebooks), Kardinalitäten…
- Attributfacetten – helfen die Attribute weiter zu erläutern. Aussehen und Verhalten von Attributen werden konfiguriert.
- Modelltypen und Modelltypattribute - stellen sinnvolle Kombinationen von Klassen und Beziehungsklassen dar, die vom Modellierer zum Erstellen von Modellen verwendet werden.

Im ersten Schritt wurden Klassen (dabei wurde auf Klassenhierarchien geachtet) und relationale Klassen erstellt. Die graphische Notation der Klassen und der relationalen Klassen wurde mittels GRAPHREP definiert. Danach wurden Attribute, Attributfacetten und mittels ATTREP, die Struktur der jeweiligen Notebooks der erstellten Klassen definiert.
Da es zu erwarten war, dass das Modell aufgrund von vielen Rollen, die beteiligt sind, und vielen Kontexten und Aktionen, die benötigt werden um Anforderungen zu spezifizieren, unübersichtlich werden könnte, wurden vier Modelltypen definiert (Context, Role, Action und Product).

Die folgenden Abbildungen zeigen Klassenhierarchie (Abb. 16), ein Beispiel vom GRAPHREP (Abb. 17) und ein Beispiel eines Notebooks (Abb. 18).

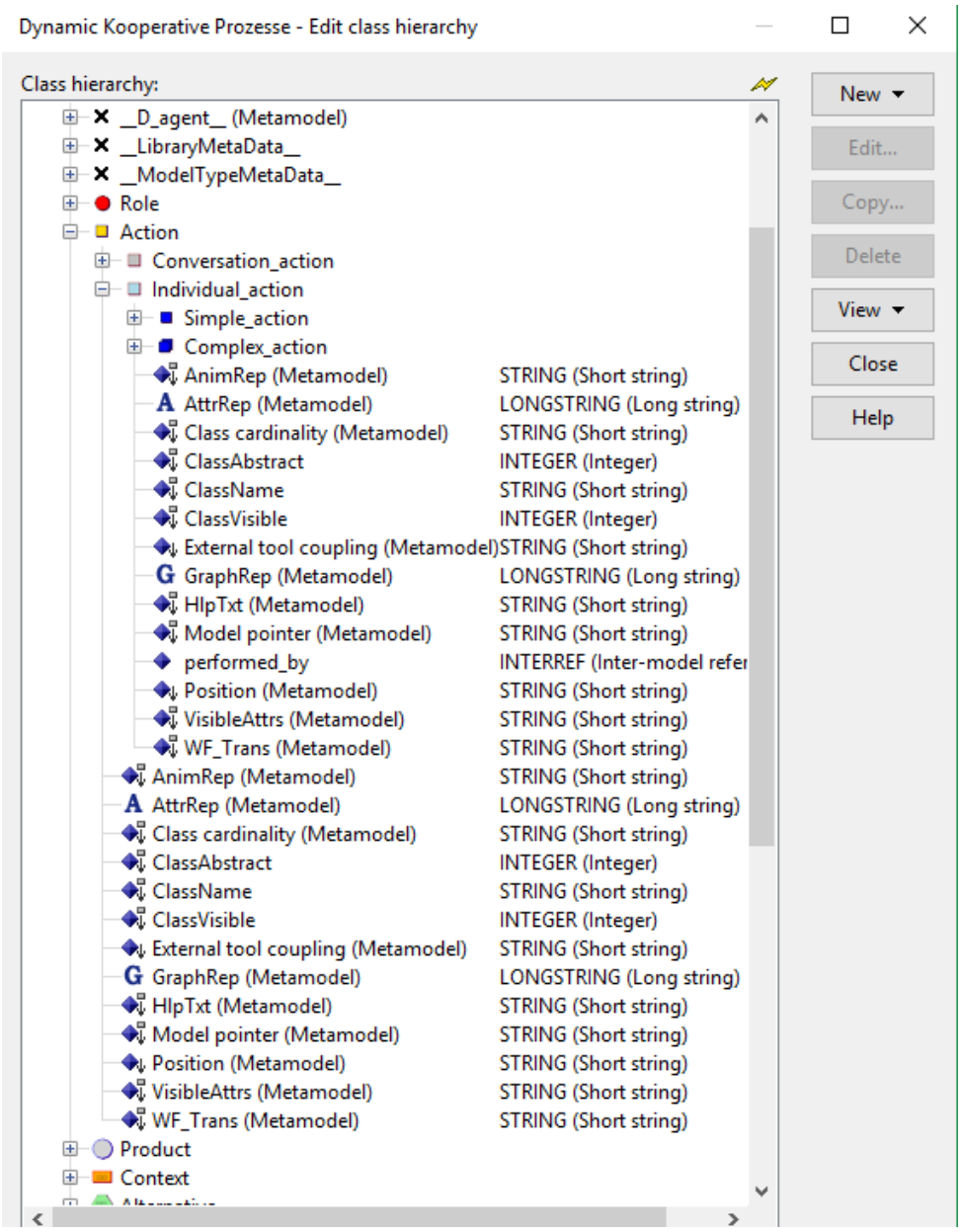


Abbildung 16. Klassenhierarchie in Adoxx

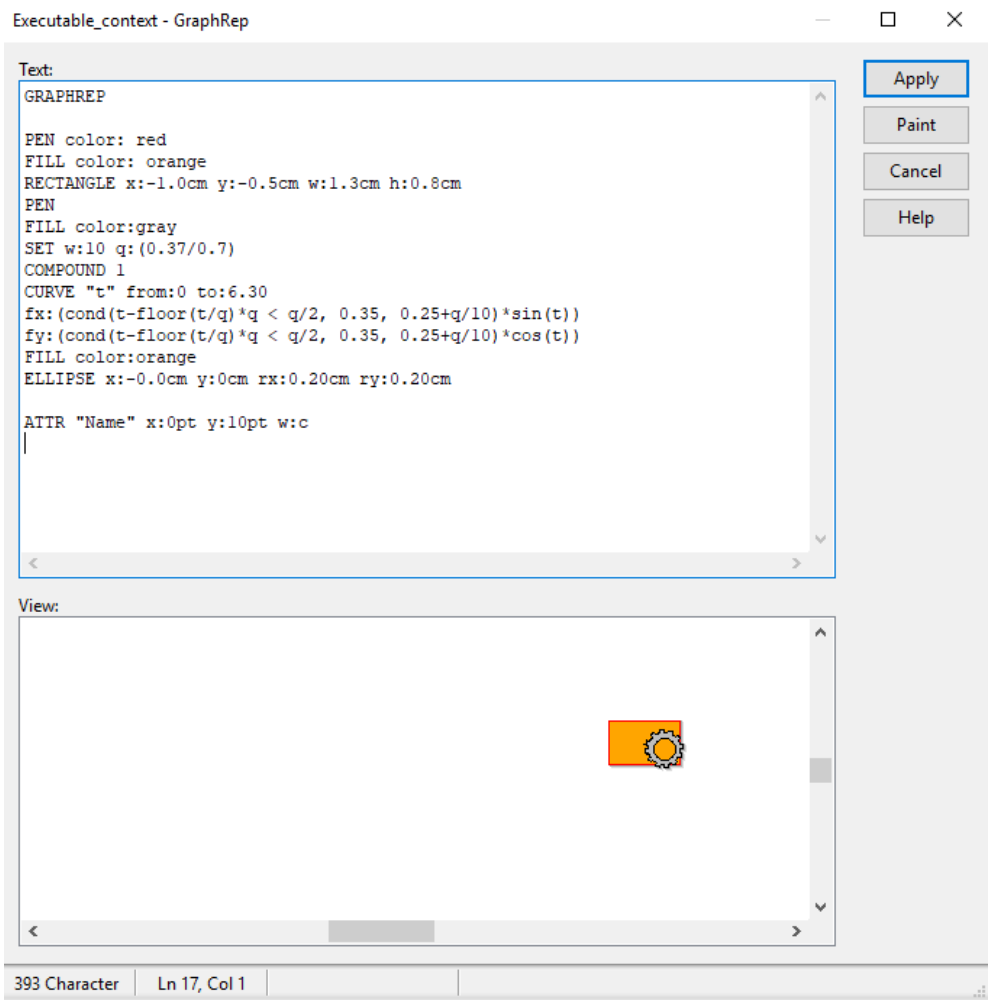


Abbildung 17. Beispiel einer GRAPHREP

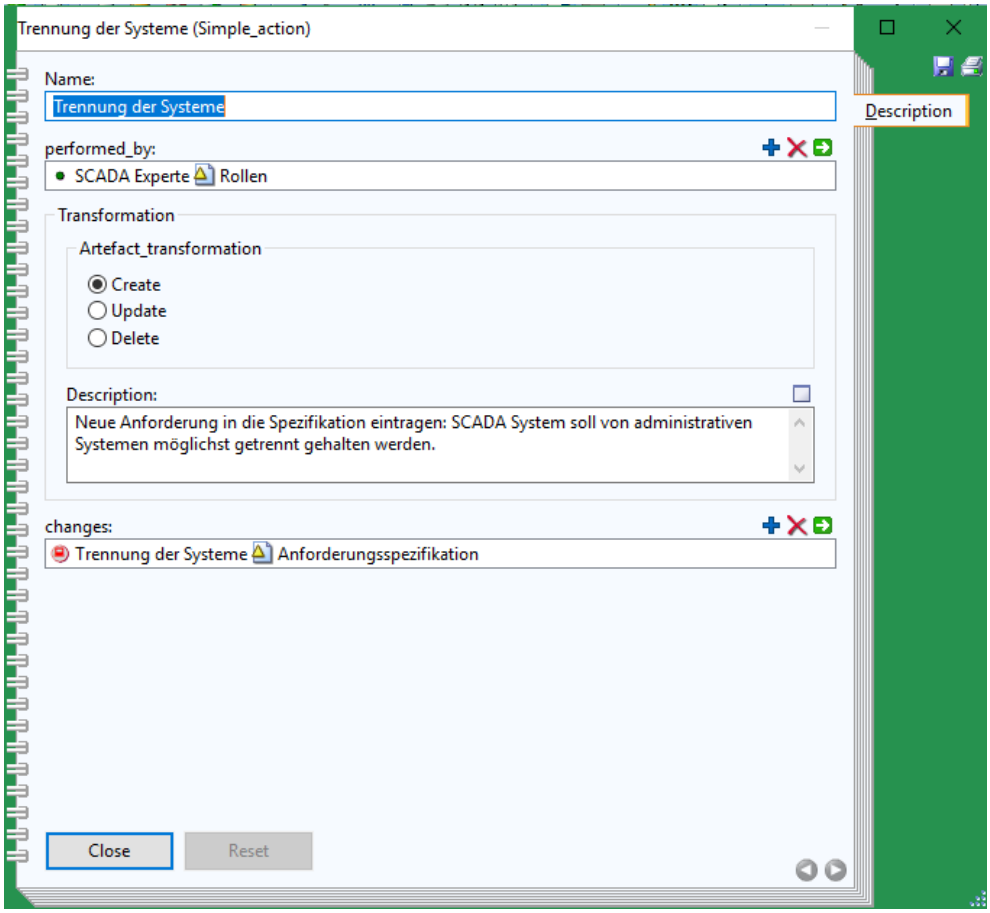


Abbildung 18. Beispiel eins Notebooks

Anschließend wurde ein Algorithmus implementiert, der in einer Viewbox alle identifizierten Anforderungen und deren Beschreibungen anzeigt. Dadurch wird erleichtert, die identifizierte Anforderungen zu lesen. Dieser Algorithmus funktioniert nur in dem Modelltyp Product. Versucht man diesen in einem anderen Modelltyp zu starten, gibt es einen entsprechenden Hinweis.

6 FALLSTUDIE: SPEZIFIKATION DER SICHERHEITSANFORDERUNGEN UND DIE ERSTELLUNG EINER SICHERHEITSSTRATEGIE

In diesem Kapitel wird am Beispiel eines fiktiven Unternehmens, das ein SCADA-System implementieren möchte, gezeigt, wie man das im Kapitel 5 beschriebene Modell nützen könnte, um Sicherheitsanforderungen zu spezifizieren. Zuerst werden die Eigenschaften und die Bestandteile eines SCADA-Systems beschrieben und das Unternehmen, ein Stromlieferant, vorgestellt. Anschließend wird eine Strategie zur Erstellung der Cyber-Security in dem Unternehmen ausgearbeitet und der Prozess der Anforderungsspezifikation mit Hilfe des Modells beschrieben.

6.1 SCADA

SCADA steht für Supervisory Control and Data Acquisition, also Überwachung Steuerung und Datenerfassung. SCADA-Systeme werden verwendet um komplexe industrielle Prozesse zu automatisieren. Die Prozesse, die an verschiedenen Standorten verteilt sind, werden überwacht und gesteuert. Anwendungsgebiete der SCADA-Systeme sind: Stromerzeugung und Stromverteilung, Wasser und Abwasser, Fertigung, Massentransit, Verkehrssignale…[220]

Ein SCADA-System hat vier Funktionen[221]:

- Datenerfassung
- Vernetzte Datenkommunikation

[220] Rajeev Kumar Chauhan, Kalpana Chauhan, und Mohan Lal Dewal, „Intelligent SCADA System“, 2010.
[221] Chauhan, Chauhan, und Dewal.

- Datenpräsentation
- Kontrolle

Ein typisches SCADA System (Abb. 19.) besteht aus mehreren Subsystemen:[222]

- Human-Machine-Interface (HMI) - eine Mensch-Maschine Schnittstelle, auf der die Informationen dargestellt werden. Die Benutzer verwenden diese zur Überwachung und Steuerung der mit SCADA verbundenen Prozesse.
- Ein Computer - führt die Überwachung (Erfassung von Daten) sowie die Kontrolle der verknüpften Prozesse durch.
- Remote Terminal Units (RTUs) - sammeln Daten aus dem Feld. Diese werden dann von eingesetzten Sensoren an das Überwachungs- und Kontrollsystem übertragen und die notwendigen Anpassungen werden vorgenommen.
- Programmable Logic Controllers (PLCs) – werden als Alternative zu RTUs verwendet. Sie haben mehrere Vorteile gegenüber der RTUs (z.B. die Fähigkeit, Steuerlogik bereitzustellen und auszuführen).
- Eine Kommunikationsinfrastruktur – verbindet alle Komponenten der SCADA-Systeme.

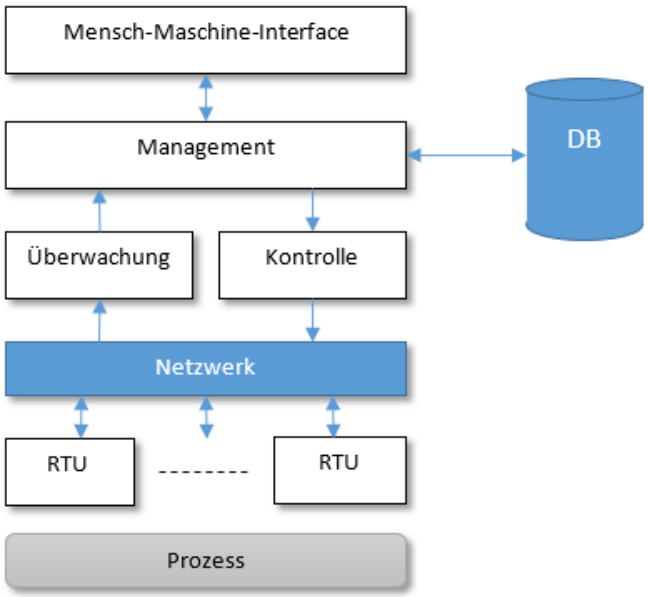


Abbildung 19. Typische Softwarearchitektur eines SCADA-Systems

[222] Stamatias Karnouskos u. a., „State of the Art in Industrial Automation“, 2014.

6.2 „STROM GMBH“

Wie man im Kapitel 5 beschriebenen Modell nützen könnte um Anforderungsspezifikation zu unterstützen, wird anhand von Sicherheits-Anforderungsspezifikation bei der Implementierung eines SCADA-Systems gezeigt.

Zur Illustration dient hier ein fiktives Unternehmen, entstanden als Fusion zweier Unternehmen „Strom GmbH“ und „Wasserkraft GmbH“. Das neue Unternehmen hat den Namen „Strom GmbH“ behalten und möchte sich am Markt als sauberer Stromlieferant ganz neu positionieren. Dabei sollen die Sicherheitsstandards eingehalten werden. Strom GmbH verfügt derzeit über 3 Wasserkraftwerke, betreibt einen Windpark mit 13 Windrädern und erzeugt Strom für 220000 Haushalte.

Beide Unternehmen, sowohl „Strom GmbH“ als auch „Wasserkraft GmbH“ verfügten jeweils über ein veraltetes SCADA System. Zur Visualisierung und Überwachung der Prozesse soll ein neues, modernes SCADA System implementiert werden. Um sicherzustellen, dass das neue SCADA-System allen Erwartungen entspricht, hat „Strom GmbH“ beschlossen, alle Sicherheitsanforderungen zu analysieren und eine Anforderungsspezifikation zu erstellen. Die Umstellung soll dann allmählich erfolgen, d.h. der neue und die alten Systeme sollen gleichzeitig ausgeführt werden, und die Anzahl der Transaktionen, die mit dem neuen System abgewickelt werden, sollte schrittweise erhöht werden, bis das alte System verschwindet.

6.3 ERSTELLUNG EINER STRATEGIE

Um die Prioritäten bezüglich Cybersicherheit im Unternehmen festzulegen, sollte eine Strategie aufgebaut werden.

Gregory J. und C.Joseph Touhill[223] empfehlen die Cybersicherheit in die Gesamtstrategie des Unternehmens einzubauen. Die Cybersicherheit sollte in den Unternehmensplänen, -richtlinien und -verfahren berücksichtigt werden. Die Risiken sollten identifiziert werden und im Einklang mit der Unternehmensstrategie und der damit verbundenen Risikobereitschaft, die die Strategie bestimmt, verwaltet werden.
Erstellung der Strategie ist Teamarbeit, daher sollten zusätzlich zu Führungskräften leistungsstarke Mitarbeiter und technische Experten hinzugezogen werden. [224]
Bei der Erstellung einer Sicherheitsstrategie sollte man sich auf die folgenden vier Fragen konzentrieren[225]:

- Wo sind wir jetzt?
- Womit sollten wir arbeiten?
- Wo wollen wir sein?
- Wie kommen wir dorthin?

6.3.1 Wo sind wir jetzt?

Diese Frage wird von vielen Unternehmen durch die SWOT Analyse beantwortet. Die SWOT Analyse untersucht die Stärken, Schwächen, Chancen und Risiken des Unternehmens. Stärken und Schwächen sind intern ausgerichtet, während Chancen und Risiken extern ausgerichtet sind.[226]

Die SWOT Analyse soll untersuchen wo das Unternehmen in Bezug auf seine Cybersicherheitsposition steht.

Das Ergebnis der SWOT Analyse für den Energielieferanten „Strom GmbH“ könnte z.B. wie in der Tabelle 2. aussehen.

[223] Gregory J. Touhill und C. Joseph Touhill, *Cybersecurity for Executives A Practical Guide* (Hoboken, New Jersey: John Wiley & Sons, Inc., 2014)., Seite 98ff
[224] Touhill und Touhill., Seite 98ff
[225] Touhill und Touhill.
[226] Touhill und Touhill.

Strenghts	**Weakness**
• Gute Internetpräsenz • Hoher Bekanntheitsgrad • Viele Prozesse sind effizient, effektiv und gut dokumentiert • Gute, qualifizierte und erfahrene Mitarbeiter • Ressourcen zur Implementierung neuer Technologien vorhanden	• Webseite könnte verbessert werden • Schutz der kritischen Informationen sollte geprüft und gegebenenfalls verbessert werden. • Die strategischen Pläne, die Cybersicherheit planen, sollten aktualisiert werden • Change-Management Prozess, sollte neu durchdacht werden • Kein einheitliches SCADA-System • Die Liste der Bedrohungsquellen sollte aktualisiert werden
Opportunities	**Threats**
• Neue Technologien einsetzen um Wettbewerbsfähigkeit zu erhöhen (Photovoltaik-Anlagen) • Webseite verbessern, indem Videos über saubere Stromerzeugung integriert werden	• Cyberangriffe • Cyber-Spionage • Höhere Gewalt • Unabsichtliche Schaden durch Mitarbeiter

Tabelle 2. SWOT Analyse

6.3.2 Womit sollten wir arbeiten?

Nachdem mittels SWOT-Analyse festgestellt wurde, wo das Unternehmen steht, kann mit der Beurteilung, womit es arbeiten soll, begonnen werden. In diesem Schritt werden technischen, menschlichen und finanziellen Ressourcen überprüft. Die Cyber-Themen, die hier behandelt werden sollen, sind: Information, Pläne, Technologie, Personal und Finanzen.[227]

Das Thema **Information** behandelt geistiges Eigentum und Geschäftsgeheimnisse, die einen Wettbewerbsvorteil gegenüber den Mitbewerbern verschaffen. Es wird geprüft, wie diese zum Erfolg des Unternehmens beitragen. Weiters wird hier geprüft, ob die Informationen ihren Wert im Laufe der Zeit behalten, oder unter welchen Bedingungen sie ihren Wert verlieren würden.
Das Thema **Pläne** behandelt Pläne und kritische Informationen. Hier wird untersucht ob Pläne ausreichend und effektiv sind und ob sie verfolgt werden. Außerdem werden hier die kritischen Informationen identifiziert und es wird geprüft, wie wertvoll sie sind und wie man sie schützen könnte.
Das Thema **Technologie** untersucht wie aktuell die Technologie ist, ob sie die Bedürfnisse der Zukunft erfüllt, oder bald die Upgrades benötigt werden. Es wird untersucht welche Software verwendet wird um allgemeine Bedrohungen zu verhindern (Antivirenprogramme, Firewalls, Intrusion Detection-Systeme) und ob die Verschlüsselung verwendet wird um die Informationen zu schützen.
Das Thema **Personal** behandelt Personal und Finanzen. Sie untersucht wie gut die Mitarbeiter sind, ob sie qualifiziert sind und ob es Universitäten, oder andere Möglichkeiten gibt um sie zu schulen und auszubilden.
Das Thema Finanzen untersucht die sowohl die aktuelle finanzielle Situation, als auch die Prognosen für die nächsten Jahre.[228]

[227] Touhill und Touhill., Seite 103
[228] Touhill und Touhill., Seite 103f

6.3.3 Wo wollen wir sein?

In jeder Strategie sollte auch eine Cybersicherheitsvision eingefügt werden. Die Vision hängt von vielen Faktoren ab, einschließlich der Branche, in der man sich befindet, den Ressourcen, die man hat, den Möglichkeiten, die man erforschen möchte, und dem Risiko, das man eingehen möchte. Die Vision sollte einfach, gut dokumentiert und leicht zu verstehen und zu merken sein, denn wenn die Vision des Unternehmens leicht zu merken ist, konzentrieren sich die Mitarbeiter eher auf die Vision und integrieren sie in alles, was sie tun. [229]

Die Vision von „Strom GmbH“ ist: Vertrauenswürdigster und Kundenorientiertester Stromlieferant im Land, der Umwelt zu Liebe.

6.3.4 Wie kommen wir dorthin?

Nachdem die Vision definiert ist, soll herausgefunden werden, wie man diese Vision erreichen kann. Vision Statement gibt den "strategischen" Blick auf das, was ein Unternehmen erreichen will, Mission Statement ist eine kurze und prägnante Aussage darüber, wie man diese Vision erreichen könnte.[230]

Das Unternehmen „Strom GmbH“ hat folgende Mission: Sicher und zuverlässig sauberen Strom zu erzeugen und zu verteilen, und auf Kundenwünsche einzugehen.

Wenn man sagt „sicher und zuverlässig“, wird dabei auch an Cybersicherheit gedacht.

[229] Touhill und Touhill., Seite 104ff
[230] Touhill und Touhill., Seite 107

6.3.5 Das Erreichen der Ziele

Sobald die Vision, die Mission und die Grundwerte definiert wurden, sollen diese als Grundlage für die Erstellung der Ziele verwendet werden, um die Vision zu verwirklichen. Hier soll zwischen Zielen und Vorgaben unterschieden werden. Die Ziele sind die Aussagen darüber was man erreichen möchte, die Vorgaben sind spezifische, messbare und zeitrelevante Aussagen darüber, was wann erreicht werden soll.
Die erstellten Ziele sind am effektivsten, wenn sie machbar, akzeptabel (dürften nicht in Konflikt mit der Vision oder den Werten des Unternehmens stehen), geeignet (müssen zur Vision und Mission des Unternehmens passen) und erschwinglich sind.
Die Vorgaben sind am stärksten, wenn sie spezifisch, messbar, erreichbar, realistisch und zeitnah sind.[231]
Der Unterschied zwischen Zielen und Vorgaben wird von Touhill und Touhill anhand folgenden Beispiels gezeigt:
Ziel: Angenommen wir haben das Ziel die Cyber-Risiken zu reduzieren.
Vorgaben: Auf der Grundlage der Risikoanalyse und Geschäftsziele können die folgenden drei Vorgaben vorgeschlagen werden[232]:

- Intrusion Detection System bis zum zweiten Quartal dieses Geschäftsjahres implementieren
- Sicherheits-Patches innerhalb von 24 Stunden nach der Veröffentlichung von vertrauenswürdigen Quellen installieren
- erste Cybersicherheitstrainings für 100% der neuen Mitarbeiter innerhalb von drei Tagen nach der Einstellung und vor dem Zugang zum Informationssystem durchführen

Um die erstellten Ziele zu erreichen, braucht man eine gute Planung. Laut Touhill und Touhill gibt es vier Gründe warum gute Strategien scheitern[233]:

- Schlechte Planung und schlechte Ausführung – sowohl gute Ausführung von schlechten Plänen als auch schlechte Ausführung von guten Plänen führt zum Scheitern.
- Mangel an Kommunikation – Kommunikation in einem Unternehmen soll nach oben und nach unten vollständig, einfach und zeitnah sein.

[231] Touhill und Touhill., Seite 108ff
[232] Touhill und Touhill., Seite 110
[233] Touhill und Touhill., Seite 111ff

Führungskräfte benötigen qualitativ hochwertige, zeitnahe und vollständige Informationen, um Entscheidungen zu treffen, und müssen diese auf untergeordneten Ebenen kommunizieren.

- Wiederstand gegen Veränderungen – es gibt immer wieder Mitarbeiter, die sich den Änderungen widersetzen und somit bewusst oder unbewusst die Umsetzung der Strategie sabotieren. Um dieses Problem zu bekämpfen, sollten die Mitarbeiter informiert werden und die Hintergrunde der Entscheidungen sollten kommuniziert werden.
- Mangel an Führung und Kontrolle - das Top-Management hat einen großen Einfluss darauf, wie gut die Mitarbeiter Strategien unterstützen. Führung und Kontrolle über die Umsetzung der Strategie ist von großer Bedeutung für Erfolg im Bereich Cybersicherheit.

Beim Aufbau der Strategie sollten folgende Grundwerte berücksichtigt werden:[234]

- Die für das Unternehmen wichtige Informationen identifizieren und schützen
- Cybersecurity als Teil der Unternehmenskultur machen
- Die Cybersecurity-Auswirkungen in den Entscheidungen berücksichtigen
- Fortschritt messen, d.h. Cybersicherheitsattribute zur Unterstützung der Strategie messen
- Gute Planung für den Erfolg

[234] Touhill und Touhill., Seite 119ff

7 EVALUATION UND DISKUSSION DER ERGEBNISSE

Wir gehen davon aus, dass das Unternehmen „Strom GmbH“ die komplette Sicherheitsstrategie (mit der internen und externen Analyse, Vision, Mission, Grundwerten, Zielen und Plänen) erstellt hat. Auf Grundlage dieser Strategie sollen jetzt die Anforderungen für das zu implementierende SCADA System identifiziert werden. Eine wichtige Frage, wie das neue SCADA- System die Mission des Unternehmens erfüllen könnte, soll dabei beantwortet werden. Der Prozess der Identifizierung der Anforderungen soll mit Hilfe der erstellten Modellierungsmethode abgebildet und unterstützt werden. Schon im Kapitel 5.3 wurde erwähnt, dass vier Modelltypen definiert wurden: Context, Role, Action und Product. Wir beginnen mit dem Modelltyp „Role“.

7.1 DIE ROLLEN

Obwohl ein Prozess mit einem Kontext beginnt, wurden der Einfachheit halber zuerst die Rollen identifiziert. Diese sind in fünf Gruppenrollen (Teams) aufgeteilt, wobei jede Rolle mehreren Gruppenrollen gehören kann:

- Expertenteam, Team des SCADA-System Anbieters
- Strom GmbH Lenkungsausschuss, Mitarbeiter für die strategische Nachverfolgung des Projekts
- Strom GmbH Engineering, Mitarbeiter für Engineering
- Strom GmbH Team, bestehend aus allen Mitgliedern des Lenkungsausschusses und des Engineering Teams und
- Projektteam, bestehend aus den ausgewählten Mitarbeitern des Expertenteams und des Strom GmbH Teams

Nur drei Teams (Strom GmbH Lenkungsausschuss, Strom GmbH Team und Projektteam nehmen an Konversationsaktionen teil.

Abbildung 20. Zeigt einen Ausschnitt aus dem Modell vom Modelltyp „Role“.

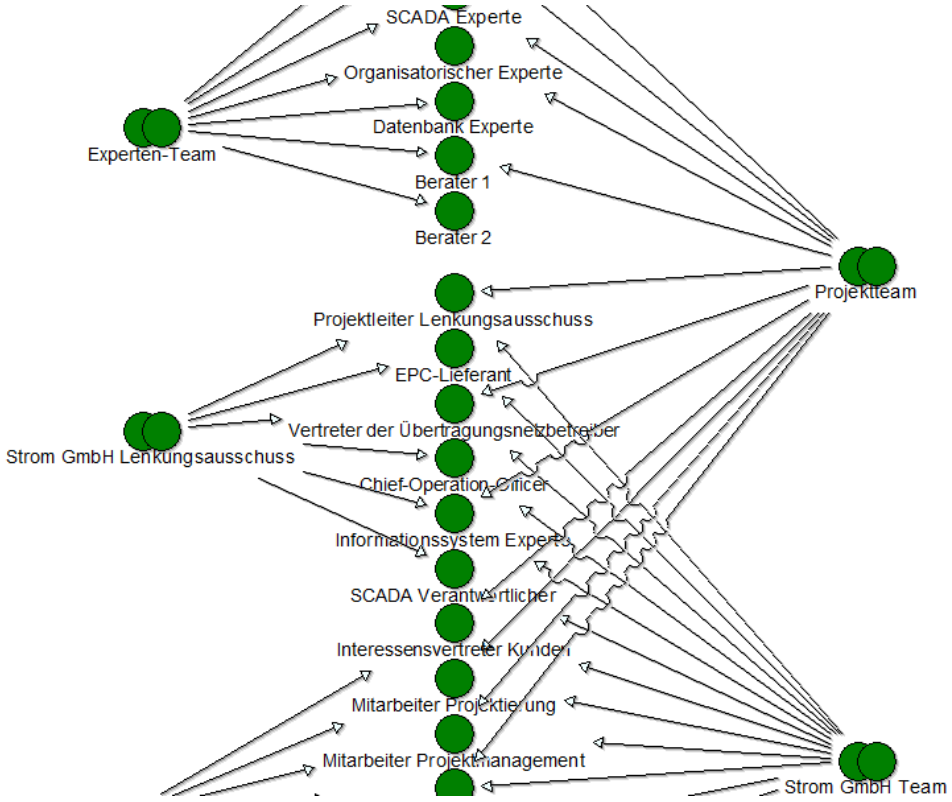


Abbildung 20. Ausschnitt aus dem Modell von Modelltyp „Role“

7.2 DIE ANFORDERUNGSERHEBUNG

Die Anforderungserhebung findet mittels Brainstorming in drei Besprechungen statt. Erste Besprechung, zwischen den Teammitgliedern des Strom GmbH Lenkungsausschusses, liefert gleich einige Anforderungen, die für die Managementebene von Bedeutung sind. Bei der zweiten Besprechung werden die Mitarbeiter nur informiert, dass sie sich Gedanken über Sicherheitsanforderungen machen sollen. Sie werden aufgefordert zu recherchieren, Berichte über die Angriffe auf Energieversorger die schon stattgefunden haben zu suchen und sich zu überlegen, wie man diese hätte verhindern können und die Notizen zu machen. Die Mitarbeiter, die an der

dritten Besprechung nicht teilnehmen, sollen Ihre Überlegungen auf einen Zettel genau schreiben, damit diese auch verwendet werden können.

Die dritte Besprechung liefert die meisten Sicherheitsanforderungen als Ergebnis. Manche davon sind grob gehalten und werden von einem Mitarbeiter (Rolle: SCADA-Verantwortlicher) verfeinert. Die Notizen, die Mitarbeiter, die an der dritten Besprechung nicht teilnehmen, gemacht haben, werden bei der Erhebung berücksichtigt. Die Rolle SCADA-Verantwortlicher ist auch für die Eintragung der Anforderungen in die Anforderungsspezifikation verantwortlich.

Im Folge werden einige Ausschnitte aus dem Modell gezeigt und kurz erläutert.

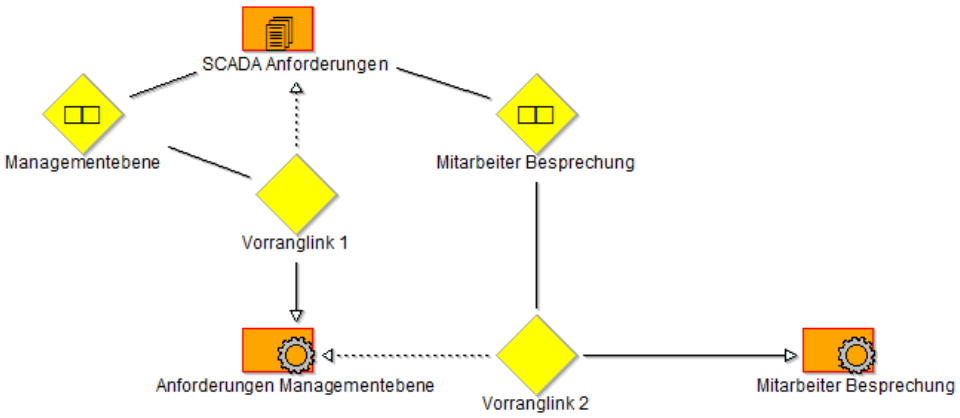


Abbildung 21. Plankontext SCADA Anforderungen (Modelltyp: Context)

Die Abbildung 21. zeigt einen Plankontext, der aus zwei Kontexten besteht, die nacheinander ausgeführt werden. Das wird mit dem Vorranglink ermöglicht. Dieser Ausschnitt bildet nur die Reihenfolge der Besprechungen ab. Zuerst findet die Anforderungserhebung in der Managementebene, und dann die Besprechung mit den Mitarbeitern statt.

Abbildung 22. Einige exekutierbare Kontexte (Modelltyp: Context)

Die Abbildung 22. bildet einige exekutierbare Kontexte ab, die als Ergebnis einer Konversationsaktion entstanden sind. Die verschiedenen Modelltypen sind miteinander mittels Inter-modell Referenz INTERREF verbunden.

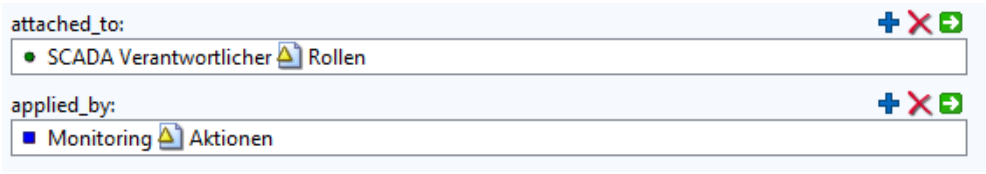


Abbildung 23. Inter-modell Referenz INTERREF vom Kontext Monitoring

Die Inter-modell Referenz in der Abbildung 23. zeigt, dass der Kontext Monitoring (vom Modelltyp Context) mit der Rolle SCADA Verantwortlicher (vom Modelltyp Role) und mit der einfachen Aktion Monitoring (vom Modelltyp Action) verbunden ist.

Die nächsten zwei Abbildungen (Abbildung 24. und Abbildung 25) zeigen die Ausschnitte aus den Modellen vom Modelltyp Action und Product.

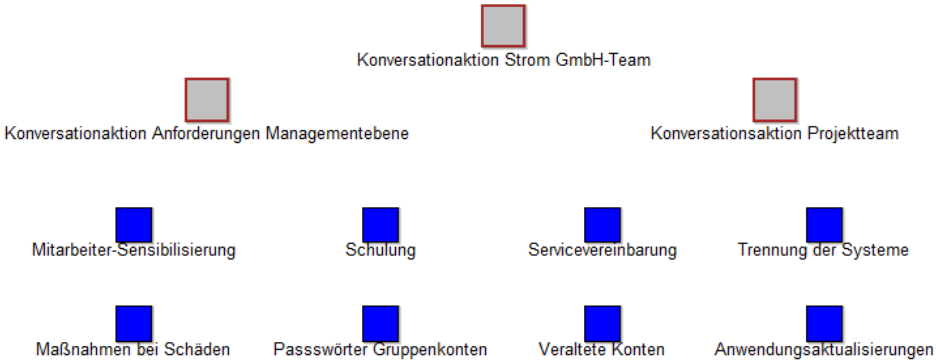


Abbildung 24. Einige Aktionen (Modelltyp:Action)

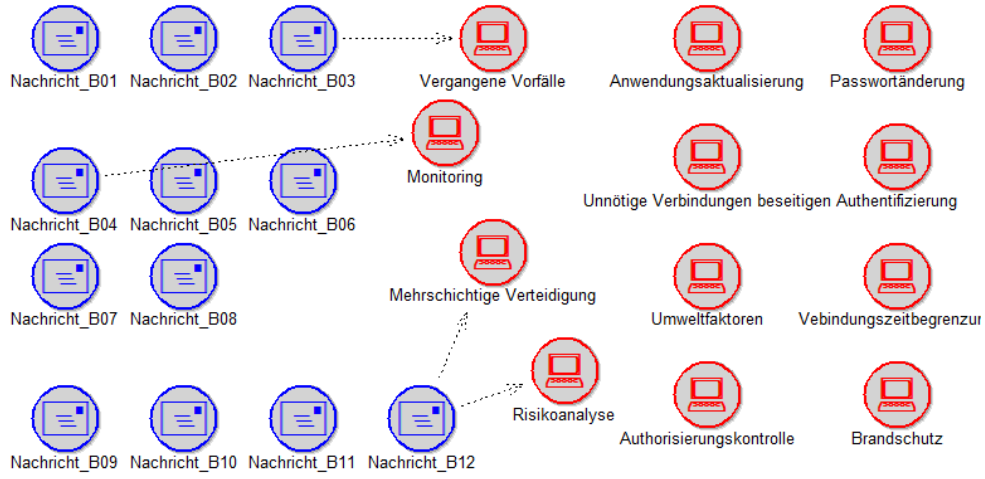


Abbildung 25. Nachrichten und Anforderungen (Modelltyp: Product)

Die identifizierten Anforderungen können zur besseren Lesbarkeit aufgerufen und in einer Viewbox angezeigt werden.

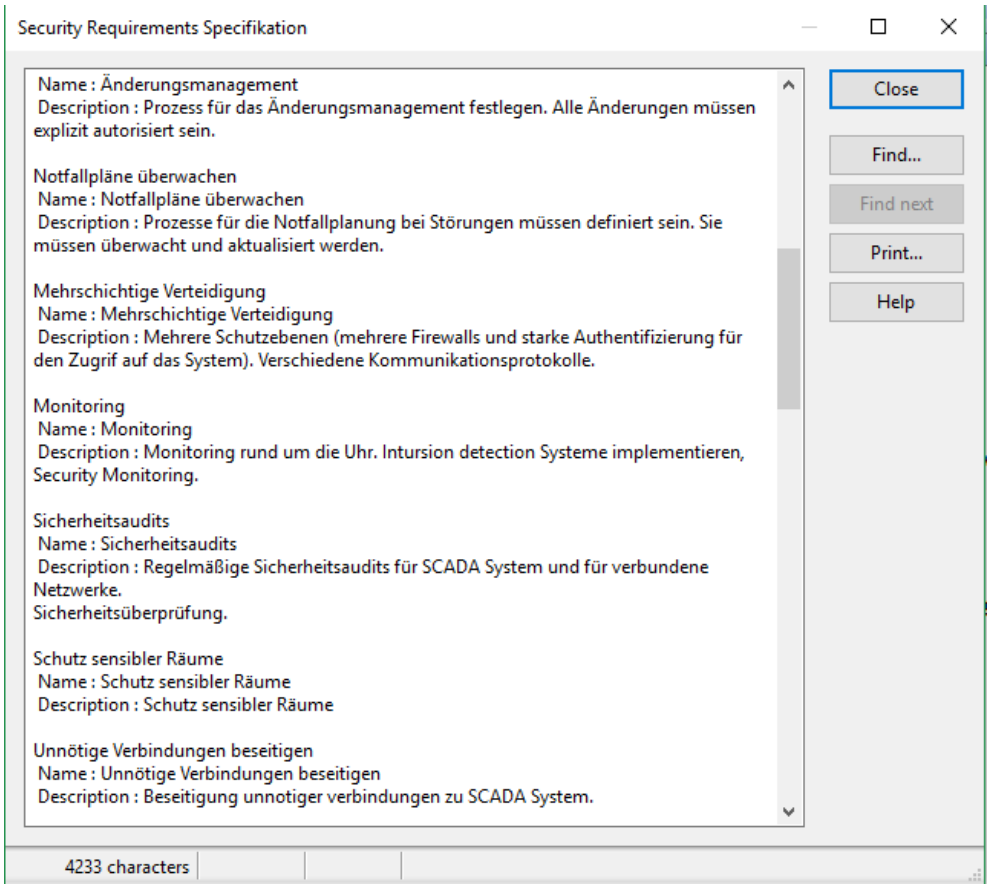


Abbildung 26. Die identifizierten Sicherheitsanforderungen in einer Viewbox

Die in dem Projekt „erhobenen“ Anforderungen stammen aus der „Guide to Increased Security in Industrial Control Systems“.[235]

Mit Hilfe dieser Modellierungsmethode konnte der Prozess der Anforderungserhebung gut abgebildet werden. Es konnte alles, was in Brainstorming-Sitzungen passiert ist, modelliert werden. Dabei sind neue Kontexte entstanden, die verbunden mit den Aktionen letztendlich zum Ziel, die Sicherheitsanforderungen zu identifizieren, geführt haben.

[235] Guide to Increased Security in Industrial Control Systems, Published by the Swedish Civil Contingencies Agency (MSB), 2010

8 ZUSAMMENFASSUNG

Die Entwicklung von großen und komplexen Systemen ist mit vielen Menschen, die unterschiedliche Sichten und Vorstellungen haben, verbunden und erfordert kooperative Arbeitsprozesse. Diese Prozesse sind sehr komplex und beinhalten viele Aktivitäten, Rollen, sowie Produkte als Ergebnis der Prozessaktivität. Die Vorgehensmodelle (Wasserfallmodell, V-Modell, Spiralmodell…) stellen diese Prozesse vereinfacht dar.

Anforderungsanalyse ist ein Teil jedes Softwareentwicklungsprozesses, und die Qualität der Anforderungsspezifikation ist von großer Bedeutung für dessen erfolgreiche Abwicklung. Da ein IT System vielen Bedrohungen ausgesetzt ist und viele Schwachstellen besitzen kann, ist es auch wichtig die Sicherheitsanforderungen zu definieren und Sicherheitsmechanismen mit denen die Sicherheitsanforderungen erfüllt werden können zu identifizieren.

Das Ziel dieser Arbeit war es, die zwei Aspekte: die kooperativen Arbeitsprozesse und die Spezifikation der Sicherheitsanforderungen zu verbinden und eine Modellierungsmethode zu erstellen, die beides unterstützt. Die Modellierungsmethode wurde auf Basis des Prozess-Metamodells von Selmin Nurcan und Colette Rolland erstellt und mit ADOxx-Plattform umgesetzt.

Mittels einer Fallstudie wurde anschließend gezeigt, wie diese Modellierungsmethode einen Prozess der Spezifikation der Sicherheitsanforderungen abbilden und unterstützen könnte. Als Beispiel wurde ein fiktives Unternehmen, ein Energielieferant, dass ein SCADA-System implementieren möchte, verwendet.

9 LITERATURVERZEICHNIS

Ahola, Jouko, Christian Frühwirth, Marko Helenius, Lea Kutvonen, Juho Myllylahti, Timo Nyberg, Ari Pietikäinen, u. a. *Handbook of The Secure Agile Software Development Life Cycle*. University of Oulu, 2014.

Altmann, Josef. *Kooperative Softwareentwicklung: Rechnerunterstützte Koordination und Kooperation in Softwareprojekten*. Linz: Universitätsverlag Rudolf Trauner, 1999.

Beck, Kent, James Grenning, Robert C. Martin, Mike Beedle, Jim Highsmith, Steve Mellor, Jeff Sutherland, u. a. „Manifesto for Agile Software Development“, 2001. http://agilemanifesto.org/.

Beng Leau, Yu, Wooi Khong Loo, Wai Yip Tham, und Soo Fun Tan. „Software Development Life Cycle AGILE vs Traditional Approaches“, 2012.

Broy, Manfred, und Andreas Rausch. „Das neue V - Modell® XT – Ein anpassbares Vorgehensmodell für Software und System Engineering“, 2005. https://doi.org/10.1007/s00287-005-0488-z.

Bundesamt für Sicherheit in der Informationstechnik. *Leitfaden Informationssicherheit*, 2012.

Bundesverband Informationswirtschaft, Telekommunikation und neue Medien, BITKOM. „Kompass der IT-Sicherheitsstandards: Auszüge zum Thema Elektronische Identitäten“, 2014.

Chauhan, Rajeev Kumar, Kalpana Chauhan, und Mohan Lal Dewal. „Intelligent SCADA System“, 2010.

Eckert Claudia. *IT-Sicherheit : Konzepte - Verfahren - Protokolle*. 8. Aufl. Oldenbourg Verlag München, 2013.

Ellis, C.A., S.J. Gibbs, und G.L. Rein. „Groupware: Some Issues and Experiences“. *Communication of the ACM* 34, Nr. l (1991).

Fabian, Benjamin, Seda Gürses, Maritta Heisel, Thomas Santen, und Holger schmidt. „A comparison of security requirements engineering methods“, 2009.

Finkelsetin, A, J Kramer, B Nuseibeh, L Finkelstein, und M Goedicke. „Viewpoints: A Framework for Integrating Multiple Perspectives in

System Development“. *International Journal of Software Engineering and Knowledge Engineering* 2, Nr. 1 (1992): 31–58.

Finkelstein, Anthony, Steve Easterbrook, Jeff Kramer, und Bashar Nuseibeh. „Requirements Engineering Through Viewpoints“, 1993.

Finkelstein, Anthony, und Jeff Kramer. „TARA: Tool Assisted Requirements Analysis“, 1991.

Giesecke, Rosemaria, und Stefan Fünfröcken. „Einfach zu mehr Software-Sicherheit: Unterstützung zur sicheren Softwareentwicklung“, 2007.

Goldsack, S J, und A C W Finkelstein. „Requirements engineering for real-time systems“, 1991.

Grechenig, Thomas, Mario Bernhart, Roland Breiteneder, und Karin Kappel. *Softwaretechnik mit Fallbeispielen aus realen Entwicklungsprojekten*. Pearson Studium, 2010.

Gross, Tom, und Michael Koch. *Computer-Supported Cooperative Work*. München: Oldenburg Verlag, 2009.

Hanser, Eckhart. *Agile Prozesse: Von XP über Scrum bis MAP*. Springer Verlag Berlin, 2010.

IEEE. *IEEE Standard Glossary of Software Engineering Terminology*. *Office*. Bd. 121990, 1990. https://doi.org/10.1109/IEEESTD.1990.101064.

International Organization for Standardization., International Electrotechnical Commission., Institute of Electrical and Electronics Engineers., und IEEE-SA Standards Board. *Systems and software engineering - life cycle processes - requirements engineering = Ingénierie des systèmes et du logiciel - processus de cycle de vie - ingénierie des exigences.* 1. Aufl. ISO, 2011.

Jendrian, Kai. „Sicherheit als Qualitätsmerkmal mit OpenSAMM“, 2012.

Karagiannis, Dimitris, Heinrich C. Mayr, und John Mylopoulos. *Domain-Specific Conceptual Modelling: Concepts, Methods and Tools*. Springer International Publishing Schwitzerland, 2016.

Karnouskos, Stamatias, Oscar Carlsson, Roberto Camp, und Matthias Riedl. „State of the Art in Industrial Automation“, 2014.

Kotonya, Gerald, und Ian Sommerville. „Viewpoints for requirements definition“. *Software Engineering Journal*, 1992.

Mead, Nancy R, Eric D Hough, und Theodore R Stehney Ii. „Security Quality Requirements Engineering (SQUARE) Methodology“, 2005.

Nurcan, Selmin, und Colette Rolland. „Meta-modelling for cooperative processes“, 1997.

Nuseibeh, Bashar, Jeff Kramer, und Anthony Finkelstein. „A Framework for Expressing the Relationships Between Multiple Views in Requirements Specification“, 1994.

Peeters, Johan. „Agile Security Requirements Engineering“, 2005.

Piwetz, Chrisstian. *Anforderungsbeschreibung für Groupware-Systeme - ein sichtenorientierter Ansatz*. Berlin: Logos verlag, 2001.

Pohl, Klaus, und Chris Rupp. *Basiswissen Requirements Engineering*. 4. Aufl. dpunkt.verlag GmbH Heidelberg, 2015.

Rein, Andre, Carsten Rudolph, Jose Fran. Ruis, und Marcos Arjona. „Introducing Security Building Block Modells“, 2012.

Rupp, Chris, und die SOPHISTen. *Requirements-Engineering und Management; Aus der Praxis von klassisch bis Agil*. 6. Aufl. Carl Hanser Verlag München, 2014.

Sabetzadeh, Mehrdad, Anthony Finkelstein, und Michael Goedicke. „Viewpoints“, 2009.

Sachar, Paulus. *Basiswissen Sichere Software: Aus- und Weiterbildung zum ISSECO Certified Professional for Secure Software Engineering*. Heidelberg: dpunkt.verlag GmbH, 2011.

———. „Geeignete Anforderungen für sichere Software“, 2012.

Sachdeva, Sakshi. „Agile Methodologies“. *(IJCSIT) International Journal of Computer Science and Information Technologies* 7(1), 2016 (2016).

Schmalz, Jan Sebastian. „Zwischen Kooperation und Kollaboration, zwischen Hierarchie und Heterarchie: Organisationsprinzipien und -strukturen von Wikis“. *kommunikation @ gesellschaft* 8 (2007): 1–21.

Sommerville, Ian. *Software Engineering*. 9. Aufl. Pearson Deutschland, 2012.

Teufel, Stephanie, Christian Sauter, Thomas Mühlherr, und Kurt Baucknecht. *Computerunterstützung für die Gruppenarbeit*. Bonn: Addison-Wesley, 1995.

„The ADOxx Metamodelling Platform - ADOxx.org“. Zugegriffen 10. Juni 2018. https://www.adoxx.org/live/home.

Touhill, Gregory J., und C. Joseph Touhill. *Cybersecurity for Executives A Practical Guide*. Hoboken, New Jersey: John Wiley & Sons, Inc., 2014.

Vogel-Heuser, Birgit. *Systems Software Engineering*. München: Oldenburg Industrieverlag, 2003.

Wirdemann, Ralf. *SCRUM mit User Storys*. Carl Hanser Verlag München, 2011.

Wohlmacher, Petra. „Sicherheitsanforderungen und

Sicherheitsmechanismen bei IT-Systemen“, 2000.

Wolf, Henning, und Stefan Roock. *Agile Softwareentwicklung*. 4. Aufl. Heidelberg: dpunkt.verlag, 2015.

Workflow Management Coalition. „Terminology & Glossary“, 1996.

Zhishang, Yang, Zerui Sun, und Shenluo Li. „Viewpoint Based Problem Modeling Technique and Its Application“. *Journal of Software* 9, Nr. 5 (2014).