Ein Vorgehensmodell für die Entwicklung sicherer Software

Armin Lunkeit[1] · Wolf Zimmer[2]

[1]OpenLimit SignCubes AG
armin.lunkeit@openlimit.com

[2]U | D | Z Unternehmensberatung
wolf.zimmer@udz.biz

Zusammenfassung

Sichere Software wird mit der Eigenschaft in Verbindung gebracht, dass der Code der Anwendung keine ausnutzbaren Sicherheitslücken aufweist. Diese Sichtweise schränkt den Blick auf die IT-Sicherheit jedoch zu sehr ein. Vielmehr setzt heute nahezu jede Software Sicherheitsfunktionalität um, beispielsweise durch die Nutzung verschlüsselter Kommunikation oder durch Verwendung anderer Sicherheitskomponenten. Sichere Software zeichnet sich demnach sowohl durch das Fehlen ausnutzbarer Sicherheitslücken als auch durch die korrekte Umsetzung angebotener Sicherheitsfunktionalität, die auf Grundlage bestehender Sicherheitsanforderungen definiert wird, aus. Dieser Beitrag beschreibt ein Vorgehensmodell zur Entwicklung sicherer Software und ordnet Methoden der IT-Sicherheit den vom Modell vorgesehenen Phasen zu.

1 Einleitung

Die Entwicklung neuer Anwendungen und Systeme sieht sich mit der Herausforderung konfrontiert, dass Anforderungen zu Beginn einer Produktentwicklung unvollständig oder gänzlich unbekannt sind und vereinfachte Annahmen an die geplante Nutzung und die spätere Einsatzumgebung getroffen werden. Nicht selten werden Anforderungen im Verlauf des Entwicklungsprozesses modifiziert, hinzugefügt oder fallengelassen. Oftmals unbehandelt bleiben dabei die möglichen Auswirkungen auf die zugrunde liegenden Annahmen. Die Komplexität steigt, wenn Sicherheitsanforderungen an die zu entwickelnde Software gestellt werden. Diese sind nichtfunktionaler Natur und bieten aufgrund der oftmals fehlenden Formalisierung viel größeren Interpretationsspielraum als funktionale Anforderungen. Um besser mit unklaren oder sich entwickelnden Anforderungen umzugehen, wurden iterative und adaptive Vorgehensmodelle entwickelt, die in der Softwareentwicklung vielfach eingesetzt werden. Diese sehen jedoch keine Ansätze vor, um auf Veränderungen und Präzisierungen von Sicherheitsanforderungen während der Entwicklungsphase zu reagieren. Dies ist für die Entwicklung von Anwendungen mit Sicherheitsfunktionalität wie im Gesundheits- und Energiesektor eine wenig komfortable Situation. Aus dieser Ausgangslage ergibt sich die Notwendigkeit eines neuen Paradigmas in der Softwareentwicklung – der Approximation. Die Autoren stellen ein rekursiv approximatives Vorgehensmodell für die Entwicklung sicherer Anwendungen (RAVENA) vor, das die Anpassung funktionaler und nicht-funktionaler Anforderungen zulässt.

P. Schartner (Hrsg.) · D•A•CH Security 2016 · syssec (2016) 339-349.

2 Kontext und verwandte Arbeiten

Die vorgestellte Arbeit steht sowohl im Kontext mit den Vorgehensmodellen und Methoden der Softwareentwicklung als auch mit denen der IT-Sicherheit. Beide Hintergründe werden besprochen.

Die Definition der Eckpunkte des sequentiellen Wasserfallmodells erfolgte 1970 durch Winston Royce in dem Artikel 'Managing the Development of Large Scale Software Systems' (vgl. [Royc70]). Aus dem Wasserfallmodell wurden V-Modell und Spiralmodell abgeleitet. Das V-Modell XT ist das Standardvorgehensmodell bei der Entwicklungsprojekten der öffentlichen Hand in Deutschland [Bart15]. Dieses bietet ein Framework für ein sequentielles Vorgehen an, dass auf die individuellen Bedürfnisse der nutzenden Organisation und/oder des Unternehmens anpassbar ist.

Das Spiralmodell [Boeh88] ist eines der ersten dokumentierten iterativen Vorgehensmodelle mit mehrfacher Wiederholung eines definierten Entwicklungszykluses als elementarem Prinzip.

Die Weiterentwicklung der iterativen Modelle ist das adaptive Vorgehen, welches einen völligen Paradigmenwechsel in der Softwareentwicklung vornimmt (vgl. [AGM]). B. Boehm greift diesen Aspekt in [Boeh02] auf und untersucht die Randbedingungen und Auswirkungen des adaptiven Vorgehens im Vergleich zu planungsgetriebenen Modellen. Der Artikel geht auf die Risiken adaptiver Vorgehensmodelle in der Softwareentwicklung ein und schlägt eine Kombination aus planungsgetriebenem und adaptivem Vorgehen vor.

Eine Erhebung zu den in Deutschland genutzten Entwicklungsmodellen erfolgt in [KuLi14] mit dem Ergebnis, dass viele verschiedene Prozessmodelle genutzt werden, kein bestimmter Ansatz in der Praxis dominiert und adaptive bzw. agile Vorgehensweisen nicht so stark dominieren, wie dies in der IT-Presse dargestellt wird.

Zur Entwicklung sicherer Anwendungen stehen Methoden der IT-Sicherheit zur Verfügung, die in den Prozess der Softwareentwicklung integriert werden müssen. In der Anforderungsphase stehen dabei die Identifikation der zu schützenden Werte, die Charakterisierung der Sicherheitsziele und Bedrohungen sowie des Angreifers im Mittelpunkt. Relevante Methoden sind in diesem Zusammenhang das Threat Modelling ([OWA], ([PoVL13]), Risikobewertungen [Prit14] und Ansätze zur Risikomodellierung (siehe dazu CORAS [LuSS11]).

Die Architektur- und Designphase muss auf der Grundlage der vorliegenden Anforderungen eine den Zielen gerecht werdende Software- und Komponentenarchitektur hervorbringen. In der Softwareentwicklung erfolgt dies mithilfe der Modellierung, beispielsweise auf UML basierend. In diesem Zusammenhang ist der Ansatz des UML-Profils UMLsec [Jür13] hervorzuheben.

Um in der Phase der Implementierung die Anforderungen an sichere Anwendungen umzusetzen, sind Metriken zur Bewertung (Komplexitätsanalyse, siehe [McCa76], [Hals77]) als auch Methoden zur Untersuchung des Codes (statische Codeanalyse, siehe [GMGM09]) verfügbar. Die Validierung der Anwendung erfolgt mithilfe geeigneter Testmethoden, wobei bereits im Entwurf des Testvorgehens und der Testfälle der Aspekt der IT-Sicherheit Berücksichtigung finden muss. Dazu ist die Unified Modeling Language (UML) in Verbindung mit dem UML Testing Profile (siehe dazu [BDGH$^+$08]) besonders geeignet. Die Verwendung von Testgeneratoren für den funktionalen Test als auch heuristische Teststrategien (siehe dazu [UtLe07]) sind in diesem Zusammenhang hervorzuheben.

Eine besondere Rolle nimmt die ISO 15408 (Common Criteria for Information Technology Security Evaluation) ([CCP12a], [CCP12b], [CCP12c]) ein. Die Common Criteria sind ein Framework zur Sicherheitsevaluierung, definieren jedoch weder ein Vorgehensmodell noch Methoden zur Entwicklung sicherer Anwendungen. Sie sind jedoch das umfangreichste Kriterienwerk zur Bewertung der IT-Sicherheit eines Produktes und unter diesem Aspekt zu berücksichtigen.

Die Entwicklung sicherer Anwendungen benötigt ein Vorgehensmodell, das die Methoden der IT-Sicherheit integriert. Es ist keine Publikation bekannt, die die Integration dieser Methoden in einem rekursiv approximativen Verfahren beschreibt. Dieser Artikel stellt einen solchen Ansatz vor.

3 Notwendigkeit eines approximativen Vorgehens

Die Notwendigkeit eines approximativen Vorgehens in der Entwicklung ergibt sich im Wesentlichen aus drei zu lösenden Themenkomplexen.

1. Es besteht ein Konflikt zwischen nicht abgeschlossener Anforderungsermittlung und frühem Implementierungsbeginn.
2. Informationen und daraus ableitbare Anforderungen des Domänenkontexts sind zu Beginn eines Entwicklungsprojekts unvollständig.
3. Nicht-funktionale Anforderungen, zu denen auch die Sicherheitsanforderungen gehören, bieten Interpretationsspielräume und weisen semantische Unschärfen auf.

Die Abklärung aller funktionalen und nicht-funktionalen Anforderungen ist ein essentieller Bestandteil eines jeden Entwicklungsprojekts. Da die Entwicklung der neuen Software in der Regel sehr früh beginnen soll, sind zu diesem Zeitpunkt häufig nicht alle Anforderungen bekannt. Das Ziel der raschen Verfügbarkeit (time-to-market) bestimmt oftmals maßgeblich den Entwicklungsprozess. In dieser Situation muss auf der Grundlage unvollständiger Information und vereinfachter Annahmen die Umsetzung wesentlicher Produkteigenschaften erfolgen und nicht-funktionale Anforderungen wie Sicherheit und Zuverlässigkeit sind vorausschauend berücksichtigen. Die Lösung eines solchen Problems ist grundsätzlich nur approximativ möglich.

Die Erfassung und Definition von Sicherheitsanforderungen erfolgt immer vor dem Hintergrund einer konkreten fachlichen Domäne. Evans [Evan04] und Beckers [Beck15] haben darauf ausführlich aufmerksam gemacht. Dies ist für die Entwicklung von Anwendungen von grundlegender Bedeutung, da die Sicherheitsanforderungen bspw. im eHealth-Sektor naturgemäß von denen, die sich aus der Modernisierung von Energienetzen und der in diesem Kontext geplanten intelligenten Messsysteme oder einer Maschine-zu-Maschine Kommunikation ergeben, verschieden sind. Der Domänenkontext ist zu Beginn eines Entwicklungsprojekts häufig durch unvollständige Informationen sowie durch rasche technologische Umbrüche gekennzeichnet, so dass auch aus dieser Perspektive ein approximatives Vorgehen geboten erscheint.

Darüber hinaus wird ein rekursiv approximatives Verfahren wegen unscharfer nicht-funktionaler Anforderungen benötigt. Die Zurechnung der Sicherheitsanforderungen zu den nicht-funktionalen Anforderungen erfolgt auf Grund der oftmals hohen Abstraktion. Diese müssen jedoch im Verlauf des Softwareentwicklungsprozesses in funktionale Anforderungen überführt werden, die ein oder mehrere von dem zu entwickelnden Produkt verfolgten Sicherheitsziele umsetzen. Die Ermittlung zu schützender Werte, Bedrohungen und Sicherheitsziele muss daher permanent in das Entwicklungsvorgehen integriert und insbesondere bei frühzeitig startender

Implementierungsphase schrittweise präzisiert werden. Die schrittweise Schärfung wird durch das approximative Vorgehen ausdrücklich vorgesehen und unterstützt die daraus resultierenden notwendigen Schritte wie Design-Überarbeitungen und Code-Refactoring.

4 Etablierte Vorgehensmodelle

Agile Softwareentwicklungsprozesse haben das Ziel, in kurzen Abständen nutzbare Prototypen der zu entwickelnden Software bereitzustellen, während sequentielle Modelle wie das Wasserfallmodell diese erst zu einem späten Zeitpunkt des Entwicklungsverlaufs zur Verfügung stellen. Im Kern nutzt das agile Paradigma der Softwareentwicklung häufige Rückkopplungen und ein zyklisch iteratives Vorgehen in allen Schritten. Dabei nehmen agile Vorgehensmodelle selbst in einem fortgeschrittenen Entwicklungsstadium neue Anforderungen auf, was äußerst stabiler Prozesse im Anforderungs- und Änderungsmanagement bedarf. In den sequentiellen Vorgehensmodellen ist die Reflexion und flexible Handhabung dagegen nicht ohne weiteres möglich. Nachfolgend werden exemplarisch fünf Vorgehensmodelle hinsichtlich ihres Vermögens, mit zunächst unklaren und unvollständigen Informationen umzugehen, beschrieben.

4.1 Das Wasserfallmodell

Das Wasserfallmodell ist ein sequentielles Vorgehensmodell mit den Phasen Anforderungsanalyse, Entwurf, Implementierung, Überprüfung (Test) und Wartung. Es ist geeignet, wenn die Anforderungen zu Beginn eines Softwareprojektes hinreichend präzise und vollständig beschrieben werden können, und in den nachfolgenden Phasen keine Änderungen zu erwarten sind.

4.2 Das V-Modell

Das V-Modell ist aus dem Wasserfallmodell entstanden, verfeinert dies in den Phasen Design, Implementierung und Test, bleibt aber sequentiell. Das daraus entwickelte V-Modell XT erlaubt die Anpassung an spezifische Projekterfordernisse. Das V-Modell ist eingeschränkt imstande, mit geänderten Anforderungen umzugehen, kann aber um ein restriktives Änderungsmanagement angereichert werden.

4.3 Spiralmodell

Das Spiralmodell ist ein iteratives Vorgehensmodell und erlaubt den Eingriff in die laufende Entwicklung. Werden Änderungen an einer Anforderung vorgenommen, so wird die Entwicklungsiteration wiederholt, bis die gewünschte Funktionalität zur Verfügung steht.

4.4 Feature-Driven Development

Das Vorgehensmodell des Feature-Driven Developments [PaFe] wurde entwickelt, um ein Projekt mit einer großen Anzahl von Entwicklern zu realisieren. Ausgehend von einer Feature-Liste wird jedes umzusetzende Feature eigenständig geplant, entworfen und umgesetzt. Der Entwurf und die Umsetzung des Features bilden ein iteratives Vorgehen ab. Die iterative Näherung an eine akzeptable Lösung ist ein Kernkonzept dieses Vorgehens. Es ist jedoch auf die Iteration aus Entwurf und Konstruktion je Feature beschränkt und geht von einer festen Liste der umzusetzenden Software-Features aus.

4.5 Scrum

Scrum ist ein adaptiver Ansatz zur agilen Softwareentwicklung. Es ist ein offenes Framework, wobei innerhalb des Vorgehensmodells drei Hauptrollen besetzt sein müssen: Product Owner, Entwicklungsteam und Scrum Master. Der Product Owner ist für den wirtschaftlichen Erfolg des entwickelten Produkts und in der Hauptsache für die Definition der Anforderungen verantwortlich. Das Entwicklungsteam verantwortet die zeitgerechte Umsetzung der definierten Anforderungen bei Einhaltung vorgegebener Qualitätsstandards. Der Scrum-Master ist für die Umsetzung des Scrum-Frameworks verantwortlich, hat jedoch keinen fachlichen Einfluss auf die Entwicklung. Anforderungsmanagement und Dokumentation werden über User Stories abgebildet, mit denen Anforderungen in der Sprache des Kunden formuliert werden sollen. Ein Scrum Projekt besteht aus mindestens einer Iteration, die im Scrum-Kontext als Sprint bezeichnet wird. Scrum versteht sich als Framework zur Implementierung eines Entwicklungsprozesses. Es geht jedoch nicht auf Methoden ein und nimmt den Aspekt der Iteration aus einer anderen Perspektive auf. Der Schwerpunkt liegt auf der Möglichkeit zur Verbesserung und Anpassung funktionaler Anforderungen. Die schrittweise Präzisierung einer Anforderung steht nicht im Mittelpunkt.

4.6 Vorläufiges Fazit zu den etablierten Modellen

Sequentielle Vorgehensmodelle sehen die Aufnahme neuer, geänderter oder präzisierter Anforderungen bei bereits laufender Implementierungsphase nicht vor. Iterative und adaptive Vorgehensmodelle können mit Modifikationen der Anforderungen umgehen, benötigen in den Iterationen jedoch ebenfalls präzise Zieldefinitionen.

User Stories in der Sprache des Managements oder Kunden sind dafür jedoch kein Äquivalent, nicht zuletzt, weil die gewählten Formulierungen häufig vage und auf dem Hintergrund vorausgesetzter und kaum falsifizierbarer Annahmen formuliert werden. Dies ist insbesondere für die Entwicklung sicherheitskritischer Anwendungen problematisch, da mit diesem Detaillierungsgrad akzeptierte und nicht-akzeptierte Sicherheitsrisiken nicht exakt beschrieben werden können.

Die Integration von Methoden zur schrittweisen Präzisierung vorliegender Sicherheitsanforderungen und die Abbildung der Konsequenzen dieser Präzisierung auf den Prozess der Softwareentwicklung wird von keinem der Modelle betrachtet.

5 Eckpunkte des Vorgehensmodells RAVENA

Der Umstand, dass Softwareentwicklung zu jedem Zeitpunkt ein approximativer Prozess ist, gehört zu den Grundeinsichten eines Vorgehensmodells für die Entwicklung sicherer Anwendungen. Der im Folgenden skizzierte Ansatz kombiniert sequentielle sowie rekursiv approximative Phasen und aggregiert bestehende Methoden zur Entwicklung sicherheitskritischer Anwendungen in einem Vorgehensmodell. Ein Kernelement ist dabei, dass Sicherheitsanforderungen in funktionale Anforderungen überführt werden, so dass eine schrittweise Konkretisierung sowohl der Anforderung als auch der Umsetzung ermöglicht wird. Die Struktur des Vorgehensmodells ist sequentiell, wobei jede der beinhalteten Phasen iterativ ist. Durch Rückkopplungen zwischen diesen entsteht die rekursiv-approximative Natur des Ansatzes.

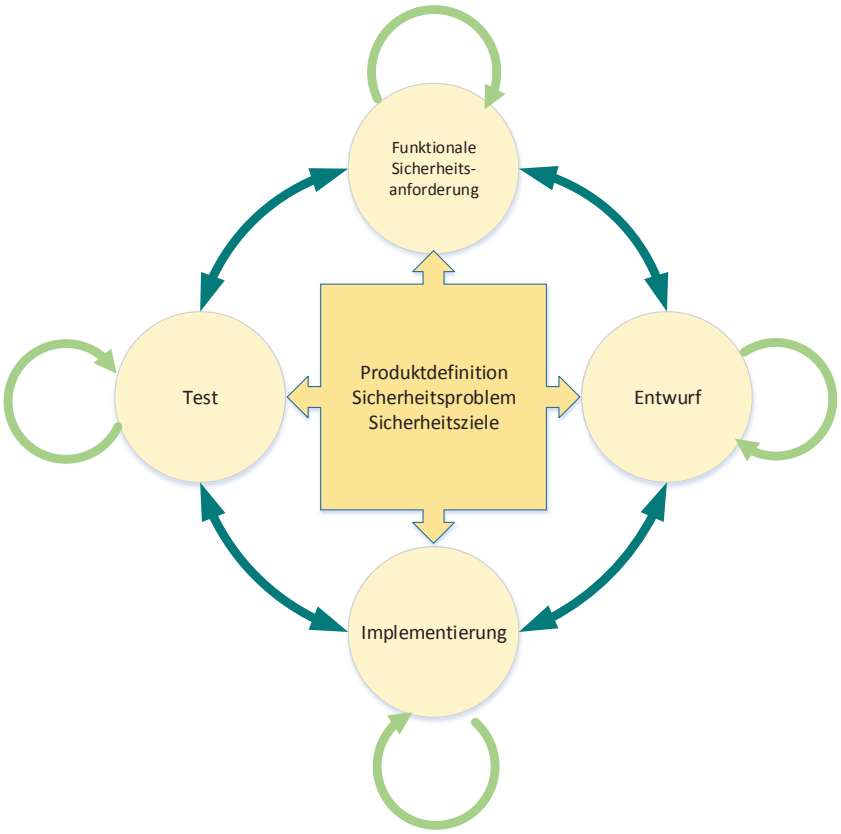

Abb. 1: Vorgehensmodell RAVENA

5.1 Definition des Produktes und Umgebung

Die Voraussetzung für die Entwicklung eines Produktes ist dessen Anforderungsspezifikation sowie die Formulierung der angenommenen Umgebung. Auf die Natur dieses Aspekts geht das Modell nicht vertiefend ein, jedoch bilden die Ergebnisse dieser Phase die Grundlage für den weiteren Verlauf. Die Definition des Produktes und der Umgebung kann sich auch bei bereits laufender Entwicklung ändern. In diesem Falle müssen die neu hinzugekommenen oder geänderten Anforderungen die Phasen des Vorgehensmodells erneut durchlaufen, wobei die Auswirkungen der Änderungsanforderungen zu berücksichtigen sind.

5.2 Definition des Sicherheitsproblems

Die Definition des Sicherheitsproblems ist eine wesentliche Tätigkeit in dem Modell. Auf Grundlage eines klar definierten Sicherheitsproblems ist es später überhaupt erst möglich, konkrete Sicherheitsfunktionalität zu definieren. Im Ergebnis dieser Phase sind die zu schützenden Werte, die Annahmen an die Systemumgebung, etwaige organisatorische Sicherheitsanforderungen, die Charakterisierung des Angreifers und die auf das neu zu entwickelnde Produkt einwirkenden Bedrohungen bekannt. Nutzbare Arbeitstechniken sind dabei das Threat-Modelling in Kombination mit STRIDE [Micr] zur Threat-Klassifikation. Der risikobasierte CORAS-Ansatz [LuSS11] aggregiert diese Ansätze und bietet neben der Threat-Ermittlung Techniken zur Definition eigenständiger Metriken zur Risikobewertung an. Diese sind in der nachfolgenden Phase von Bedeutung.

5.3 Definition der Sicherheitsziele

Um Sicherheitsziele formulieren zu können, müssen auf Grundlage des zuvor formulierten Sicherheitsproblems die Auswirkungen einer jeder einzelnen Bedrohung betrachtet werden. Während es bei Techniken wie STRIDE darum geht, die Bedrohung zu klassifizieren, stellt die Auswirkungsanalyse die Konsequenzen des erfolgreichen Angriffs in den Mittelpunkt. Zur Bewertung stehen die Techniken STRIDE [Lebl] oder auch der vom BSI entwickelte Standard 100-4 zur Verfügung. Das wesentliche Kernelement der Auswirkungsanalyse ist die Bewertung des aus der Bedrohung resultierenden Risikos für zu schützende Werte des neu zu entwickelnden Produktes und im Gegenzug die Festlegung akzeptabler und nicht-akzeptabler Risiken. Die Risikoabschätzung und Behandlung wird ebenfalls durch das risikoorientierte Verfahren CORAS unterstützt und bietet eine bessere Ausgangsbasis als das vergleichsweise einfache Vorgehen nach STRIDE.

Das Ergebnis dieser Phase ist eine Liste mit Sicherheitszielen für das neue Produkt. Jedes Sicherheitsziel muss wenigstens gegen eine identifizierte Bedrohung wirken. Die Risiken werden ebenfalls aufgeführt, da sich in späteren Phasen sowohl die Risikobewertung als auch der Schwellwert für ein akzeptiertes oder nicht-akzeptiertes Risiko ändern können. Die Annahme, dass diese Werte über den gesamten Entwicklungs- oder späteren Lebenszyklus konstant sind, vernachlässigt den Umstand, dass durch ausgewählte Technologie, Architektur und Design die Bewertung eines Risikos geändert werden muss. Die Approximation ist dann beendet, wenn keine nicht-akzeptablen Risiken mehr vorhanden sind, denen kein Sicherheitsziel entgegenwirkt.

5.4 Definition der funktionalen Sicherheitsanforderungen

Während die vorangegangenen Phasen sequentieller Natur sind, beginnt mit der Definition der funktionalen Sicherheitsanforderungen die erste approximative Phase. Aus den umzusetzenden Sicherheitszielen werden Sicherheitsanforderungen ermittelt, die nun möglichst präzise zu beschreiben sind, um die Unschärfen, die nicht-funktionalen Anforderungen zu eigen sind, zu behandeln. Zur Präzisierung der Sicherheitsanforderungen muss deren Abbildung auf die Sicherheitsziele untersucht werden. Ziel dieser Untersuchung ist es festzustellen, ob die jeweiligen Sicherheitsziele vollständig erfüllt werden oder eine weitere Präzisierung der Sicherheitsanforderungen erforderlich ist.

Approximativ ist dieser Vorgang deshalb, als das insbesondere bei komplexeren Sicherheitszielen eine schrittweise Annäherung an die Erfüllung der Sicherheitsziele erfolgt und ein Sicherheitsziel durch mehrere funktionale Sicherheitsanforderungen adressiert werden kann. Ebenso können mehrere unterschiedliche Lösungen zur Umsetzung eines Sicherheitsziels vorhanden sein. Die Approximation in dieser Phase endet, wenn die Sicherheitsziele durch die formulierten Sicherheitsanforderungen erreicht werden.

Es sind nur wenige Techniken für eine Ableitung funktionaler Sicherheitsanforderungen aus Sicherheitszielen vorhanden. Einen Leitfaden bieten die Common Criteria, insbesondere in Teil 2 wird das Paradigma der Security Functional Requirements erläutert und mit vordefinierten Komponenten unterstützt. Dieses Vorgehen ist sehr formal und vor allem dann geeignet, wenn eine Sicherheitsevaluierung gemäß ISO 15408 des neu entstehenden Produktes geplant ist. Besser geeignet und wesentlich näher an den Arbeitstechniken der Softwareentwicklung ist die Arbeit mit Diagrammen wie sie im Threat Modelling genutzt werden und auch von CORAS mit dem Risk Treatment Diagram vorgeschlagen werden. In dieser Methode werden die Maß-

nahmen gegen eine Bedrohung eingezeichnet, wobei die Maßnahme selbst das Sicherheitsziel formuliert. Setzt man diese Diagramme in einen Kontext mit den bereits bekannten funktionalen Anforderungen, können die funktionalen Sicherheitsanforderungen abgeleitet werden. Im Ergebnis dieser Phase ist ein Anforderungskatalog aus Sicherheitsanforderungen entstanden, der in die nachfolgende Phase des Entwurfs einfließt.

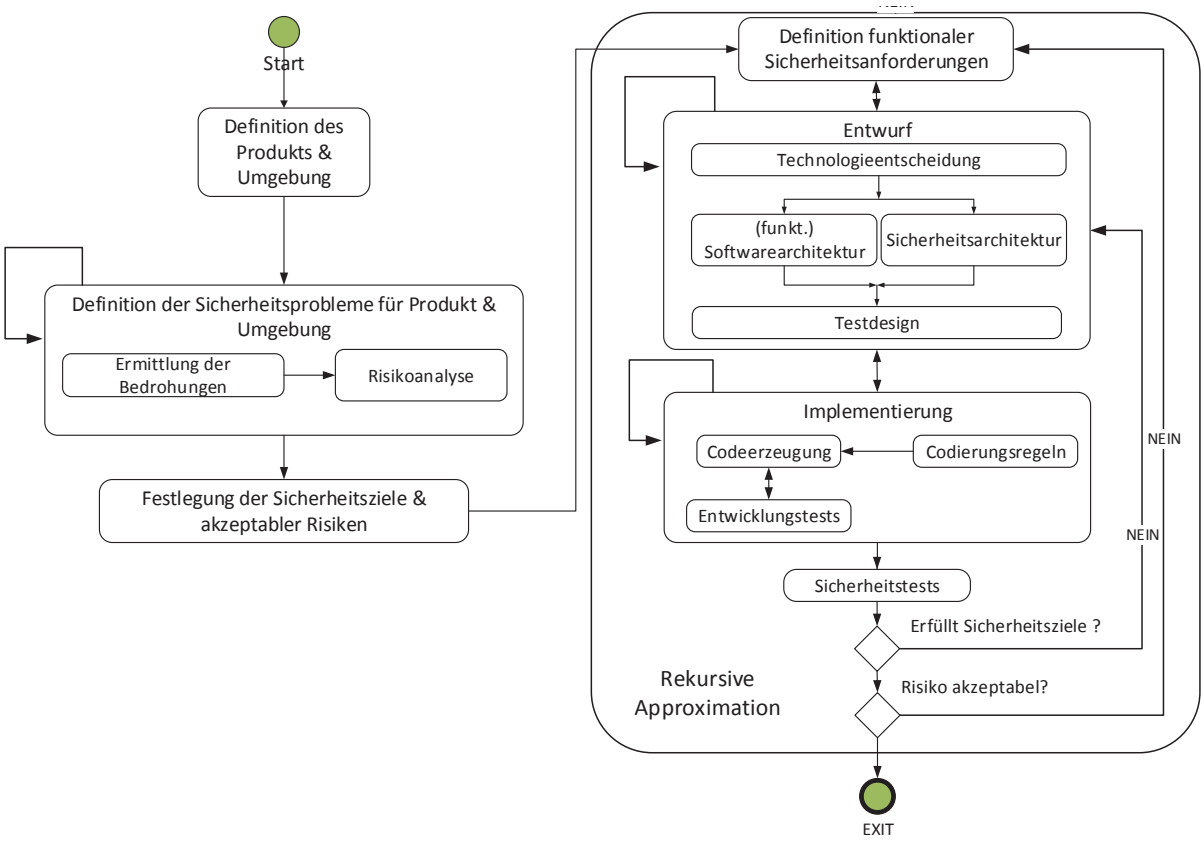

Abb. 2: Sequentielle und approximative Phasen im Vorgehensmodell

5.5 Entwurf

Der nächste Approximationsschritt ist maßgeblich bestimmt von technologischen Entscheidungen, dem Softwaredesign und der Konzeption der Sicherheitsarchitektur. Von entscheidender Bedeutung ist für diese Entwurfsphase die in Sicherheitstests überprüfbare Abbildung, der aus den Sicherheitszielen abgeleiteten funktionalen Sicherheitsanforderungen auf das Design, die Architektur und im speziellen auf die Sicherheitsarchitektur. Dies kann Änderungen im Softwaredesign bzw. der Softwarearchitektur oder auch in den technologischen Konzepten nach sich ziehen. Jede Approximation im Entwurf muss mit einer Korrespondenzanalyse abgeschlossen werden, die für jedes einzelne Sicherheitsziel dokumentiert, mit welcher Design- oder Architekturentscheidung dieses adressiert und auf Grundlage getroffener technologischer Entscheidungen auch umgesetzt werden kann. Für diesen Schritt empfiehlt sich der Einsatz von Modellierungsmethoden wie UML. Die Nutzung von UML ohne entsprechende Profilierung ist allerdings kaum geeignet, sicherheitskritische Sachverhalte darzustellen. Profile wie UMLsec bieten einen methodischen Ansatz zur Modellierung kryptografischer Protokolle und deren Abbildung in den einzelnen Komponenten.

Aus dem Entwurf können neue Bedrohungen und Risiken erwachsen, die in der ursprünglichen Analyse nicht berücksichtigt werden konnten, da zum Zeitpunkt der ersten Bedrohungsanalyse und Risikobewertung noch keine Entwurfsunterlagen verfügbar waren. Zu diesem Zeitpunkt muss ein Review identifizierter Bedrohungen vor dem Hintergrund der gewählten Architektur und des gewählten Designs erfolgen. Es erfolgt eine Rückkopplung auf die Phase der Definition des Sicherheitsproblems, die in Folge dessen neue Sicherheitsziele und neue funktionale Sicherheitsanforderungen hervorbringen kann. In diesem Punkt unterscheidet sich das Vorgehensmodell vom gewählten Ansatz der Common Criteria. Diese arbeitet mit einer festen Menge an Bedrohungen, Sicherheitszielen und Sicherheitsanforderungen und adressiert Schwächen aus Technologie und Design mit einer Schwachstellenanalyse. Dieser entscheidende Schritt erfolgt jedoch erst zum Ende der Produktprüfung am bereits fertig entwickelten Produkt, was sich im Falle einer aufgedeckten Schwachstelle negativ auf den gesamten Entwicklungsprozess auswirkt. Bereits in der Entwurfsphase muss demnach das Testdesign berücksichtigt werden. Die Testbarkeit der implementierten Sicherheitsfunktionalität muss bereits in der Entwurfsphase berücksichtigt werden, um zum Ende der approximativen Phase den Sicherheitstest durchführen zu können. Nah an den Techniken der Softwareentwicklung steht dabei das UML Testing Profile [UTP], welches das Design der Testfälle mithilfe der UML ermöglicht. Auf Grundlage dieses formalisierten Vorgehens werden Architektur und Testdesign zusammengeführt. Die approximative Entwicklung des Architekturdesigns und die Technologieauswahl sind beendet, wenn die umzusetzenden Anforderungen abgebildet sind und keine neuen, in der Entwurfsphase begründeten Bedrohungen mehr aufgedeckt werden.

5.6 Implementierung

Ohne eine korrekte Umsetzung der Sicherheitsziele im Quellcode und der aus ihm erzeugten ausführbaren Funktionen des Systems ist eine Gewährleistung der erwarteten oder auch zugesicherten Sicherheitseigenschaften nicht vorstellbar. Von besonderer Bedeutung ist es dabei, die Umsetzung der Architektur und des Datenflusses in der Kodierung zu verfolgen. Bislang sind dazu keine besonders geeigneten Verfahren dokumentiert. Die Common Criteria behelfen sich mit einem Korrespondenznachweis zwischen Implementierung und funktionaler Sicherheitsanforderung. Ein besserer Ansatz ist die Nutzung eines geeigneten Round-Trip Engineering Werkzeugs, mit dessen Hilfe Codeartefakte den jeweiligen Sicherheitsanforderungen zugeordnet und die Einhaltung zulässiger Datentypen und Konstrukte weitgehend automatisch verfolgt werden können. Typische Hilfsmittel sind dabei statische Code-Analysatoren, Reverse-Engineering Tools zur automatischen Erzeugung von Klassen- und Komponentenansichten aus dem Code, aber auch vom Compiler generierte Warnungen. Eine weitere wichtige und mächtige Möglichkeit zur Bewertung der Codequalität ist die Komplexitätsbewertung. Dabei werden Metriken eingesetzt, die eine Aussage über Anzahl der Operanden, Verschachtelungen, Ablaufzweige ermöglichen. Steigt die automatisiert ermittelte Komplexität über einen zuvor definierten Schwellwert an, entstehen neue Risiken. Entweder reicht die Überarbeitung des betreffenden Codeabschnitts aus, oder die Komplexität ist bereits aus der gewählten Architektur und Technologie sehr hoch, so dass ein neuerlicher Einstieg in die Phase der Architektur und des Designs ratsam sein kann. Während diese Werkzeuge in der Hauptsache die Qualität der Kodierung in den Mittelpunkt stellen, ist jedoch auch die Korrektheit der Implementierung von Interesse. Hierzu sollten Werkzeuge für die Verifikation, sog. Model Checker, eingesetzt werden (siehe dazu vertiefend [BaKa07]). Neben der Nutzbarkeit für den Beweis der Fehlerfreiheit einer Spezifikation können diese auch zur Prüfung der korrekten Ausführung von programmiertem Code eingesetzt werden. Für Anwendungen mit großer Codebasis sind diese

Werkzeuge noch nicht geeignet, jedoch lassen sich tatsächlich sicherheitskritische Anteile mit dieser Methodik prüfen. Davon ausgehend ist die Implementierung beendet, wenn sowohl die korrekte Implementierung in funktionaler Hinsicht als auch die Qualität des Codes hinsichtlich potentieller Sicherheitsprobleme gezeigt und dokumentiert wurde.

5.7 Test

Jede Approximation wird mit einem Sicherheitstest abgeschlossen. Ziel dieses Tests ist die Überprüfung, dass die für die Behandlung der ermittelten Risiken zugesicherten Sicherheitseigenschaften tatsächlich nachgewiesen werden können. Ist dies nicht oder nicht vollständig der Fall oder wurde zwischenzeitlich die Anforderung modifiziert, wird eine neue Approximation im Entwurf notwendig, da nicht ausgeschlossen werden kann, dass unscharfe Anforderungen, technologische Entscheidungen oder Mängel in der Sicherheitsarchitektur an diesem Ergebnis ursächlich beteiligt sind. Sicherheitsorientierte Testverfahren testen sowohl die Sicherheitsfunktionalität entlang der funktionalen Sicherheitsanforderungen, nutzen aber auch Verfahren wie heuristische Testverfahren oder auch Penetrationstests. Das genutzte Testverfahren orientiert sich dabei streng an den zu prüfenden Sicherheitsfunktionen, die das Ergebnis der Analyse der einwirkenden Bedrohungen, gesetzter Annahmen und abgeleiteter Sicherheitsanforderungen sind. Werden keine Verletzungen der Sicherheitsziele gefunden, ist der Sicherheitstest erfolgreich.

6 Zusammenfassung und Ausblick

Das vorgestellte Vorgehensmodell unterstützt die Entwicklung sicherheitskritischer Anwendungen durch schrittweise Schärfung der Sicherheitsanforderungen. Dies äußert sich in einem rekursiv approximativen Vorgehen und sichert durch definierte Abbruchkriterien die systematische und vollständige Umsetzung aller funktionalen Sicherheitsanforderungen zu. Es greift das in der Common Criteria dokumentierte Paradigma der Definition funktionaler Sicherheitsanforderungen auf, verzichtet aber auf die dort angewandte Formalisierung. Die Stärke des Modells liegt in der Nutzung etablierter Methoden der IT-Sicherheit in einem Vorgehen für den Softwareentwicklungsprozess. Die genutzten Methoden konzentrieren sich jeweils auf spezielle Teilaspekte der IT-Sicherheit, im vorgestellten Ansatz werden diese in einem Entwicklungsvorgehen nutzbar gemacht. Zukünftig wird dieser Ansatz weiter verfeinert und als Framework für die Entwicklung sicherer Software nutzbar gemacht.

Literatur

[AGM] Agile Manifesto. Online: http://agilemanifesto.org.

[BaKa07] C. Baier, J.-P. Katoen: Principles of Model Checking. The MIT Press (2007).

[Bart15] C. Bartelt: V-Modell XT Das deutsche Referenzmodell für Systementwicklungsprojekte. Verein zur Weiterentwicklung des V-Modell XT e.V. (2015).

[BDGH+08] P. Baker, Z. Dai, J. Grabowski, O. Haugen, I. Schieferdecker, C. Williams: Model-Driven Testing – Using the UML Testing Profile. Springer-Verlag Berlin Heidelberg (2008).

[Beck15] K. Beckers: Pattern and Security Requirements. Springer-Verlag Berlin Heidelberg (2015).

[Boeh88] B. Boehm: A Spiral Model for Software Development and Enhancement. In: *IEEE Computer*, 21, 5 (1988), 61 – 72.

[Boeh02] B. Boehm: Get ready for agile methods with care. In: *Computer*, 35, 1 (2002), 64–69.

[CCP12a] Common Criteria for Information Technology Security Evaluation, Part 1: Introduction and general model, Version 3.1, Revision 4 (2012).

[CCP12b] Common Criteria for Information Technology Security Evaluation, Part 2: Security functional components, Version 3.1, Revision 4 (2012).

[CCP12c] Common Criteria for Information Technology Security Evaluation, Part 3: Security assurance components, Version 3.1, Revision 4 (2012).

[Evan04] E. Evans: Domain Driven Design. Addison-Wesley (2004).

[GMGM09] I. Gomes, P. Morgado, T. Gomes, R. Moreira: An overview on the Static Code Analysis approach in Software Development. In: *Faculdade de Engenharia da Universidade do Porto, Portugal* (2009).

[Hals77] M. H. Halstead: Elements of software science. Operating and programming systems series, Elsevier (1977).

[Jür13] J. Jürjens: UMLsec: Extending UML for Secure Systems Development. In: *http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.19.2850* (2013).

[KuLi14] M. Kuhrmann, O. Linssen: Welche Vorgehensmodelle nutzt Deutschland. In: *PMV* (2014).

[Lebl] D. Leblanc: DREADful. Online: http://blogs.msdn.com/b/david_leblanc/-archive/2007/08/13/dreadful.aspx.

[LuSS11] M. Lund, B. Solhaug, K. Stolen: Model-Driven Risk Analysis – The CORAS Approach. Springer-Verlag Berlin Heidelberg (2011).

[McCa76] T. J. McCabe: A Complexity Measure. In: *IEEE Transactions on Software Engineering*, SE-2, 4 (1976), 308 – 320.

[Micr] Microsoft: The STRIDE Threat Model. Online: http://msdn.microsoft.com/en-us/library/ee823878

[OWA] Open Web Application Security Project (OWASP): Application Threat Modeling. Online: https://www.owasp.org/index.php/Application_Threat_Modeling.

[PaFe] S. R. Palmer, M. Felsing: A Practical Guide to Feature-Driven Development. Pearson Education.

[PoVL13] H. Pohl, T. Voß, A. Lunkeit: Threat Modeling Smart Meter Gateways. In: (2013).

[Prit14] C. L. Pritchard: Risk management: concepts and guidance. CRC Press (2014).

[Royc70] W. Royce: Managing the Development of Large Scale Software Systems. In: *Technical Papers of Western Electronic Show and Convention* (1970).

[UtLe07] M. Utting, B. Legeard: Practical Model-Based Testing. Morgan Kaufmann (2007).