Gottfried Wilhelm
Leibniz Universität Hannover
Fakultät für Elektrotechnik und Informatik
Institut für Praktische Informatik
Fachgebiet Software Engineering

Heuristische Unterstützung für die Formulierung von Sicherheitsanforderungen

Masterarbeit

im Studiengang Informatik

von

Sinan Petrus Toma

Prüfer: Prof. Dr. Kurt Schneider
Zweitprüfer: Prof. Dr.-Ing. Christian Grimm
Betreuer: M. Sc. Eric Knauss

Hannover, 27. März 2009

Abstract (Kurzfassung)

Anforderungen sind von zentraler Bedeutung in Softwareprojekten, weil sie die Basis für alle weiteren Prozesse in der Softwareentwicklung darstellen. Kunden sind meistens nicht in der Lage, selbst in unsystematischer Form, ihre Anforderungen zu formulieren und haben keine bzw. sehr geringe Vorstellungen von Sicherheitsanforderungen. Deshalb gerät Sicherheit meistens in den Hintergrund und wird nicht berücksichtigt. Sicherheitslücken werden demzufolge nicht aufgedeckt und von unseriösen Nutzern missbraucht, um die Systeme anzugreifen, was zu großen Schäden führen kann.

Die Entwicklung von sicheren Softwaresystemen ist selbst für Sicherheitsexperten eine komplexe Aufgabe. Diese Arbeit stellt schlüsselwortbasierte und stochastikbasierte Heuristiken aus dem Bereich Information Retrieval und Data Mining vor, die Anforderungen in sicherheitskritische und nicht-sicherheitskritische Anforderung klassifizieren. Damit werden Anforderungsanalysten und Softwareentwicklern, selbst wenn sie keine Sicherheitsexperten sind, dabei unterstützt, Sicherheit bereits in den frühren Entwicklungsphasen einzubinden. Sie werden bei der Anforderungserhebung auf potenzielle sicherheitskritische Anforderungen aufmerksam gemacht. Weiterhin werden sie mit Ansätzen zu Verfeinerung der sicherheitskritischen Anforderungen unterstützt. Damit werden alle sicherheitsrelevanten Informationen erfasst. Diese Informationen fließen im nächsten Schritt in der Modellierung mit UMLsec ein. UMLsec ist eine UML-Erweiterung zur Modellierungen von Sicherheitsaspekten. Somit kann Sicherheit von Anfang an ebenso im Entwurfsprozess integriert werden.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Masterarbeit selbständig und ohne fremde Hilfe verfasst und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel verwendet habe. Die Arbeit hat in dieser oder ähnlicher Form noch keiner anderen Prüfungskommission vorgelegen.

Hannover, den 27.03.2009

Danksagung

Ich bedanke mich ganz herzlich bei Herrn M. Sc. Eric Knauss für die hervorragende und vorbildliche Betreuung meiner Masterarbeit. Für die Motivation, die vielen hilfreichen Hinweise und die konstruktive Kritik, ohne sie diese Arbeit nicht in dieser Form existiert hätte, danke Eric.

Weiterhin möchte ich mich bei Herrn Prof. Dr. Schneider und Herrn Dr. Jan Jürjens bedanken, die mir bei Fragen mit Rat und Tat zur Seite gestanden haben.

Inhaltsverzeichnis

1. Einleitung

Die Einleitung stellt die Problemstellung und die Ziele dieser Arbeit vor. Diese werden von einer Gliederungsbeschreibung dieser Masterarbeit gefolgt.

1.1 Problemstellung

Viele Programme und Systeme, wie z.B. Online-Shops, Online-Kalender und IP-Telefonie sind sicherheitskritisch, weil sie sich dem Internet als Kommunikationsmittel bedienen, was sowohl von seriösen als auch von unseriösen Nutzern verwendet wird.

Solche Informationsnetzwerke, die mit dem Internet verbunden sind, werden oft angegriffen, was meistens zu einem hohen ökonomischen Schaden führt. Diese Attacken können jederzeit von einem beliebigen anonymen Platz durchgeführt werden.

Nach einer Studie von „Computer Security Institute" [1] berichteten 223 Firmen von einem finanziellen Schaden in Höhe von insgesamt 455.848.000 US-Dollar durch Sicherheitslöcher, die meist durch fehlerhafte Protokolle herbeigeführt wurden. Solche Probleme mit der Vertraulichkeit oder der Sicherheit von Daten gelangen durch die Medien in die Öffentlichkeit und verursachen einen großen Image-Schaden für die jeweiligen Unternehmen.

Deswegen ist es sehr wichtig, sichere und zuverlässige Systeme zu entwickeln. Die Modellierung und der Entwurf sicherheitskritischer Systeme ist jedoch selbst für Experten eine schwierige und fehleranfällige Angelegenheit. Weiterhin haben Software-Entwickler meistens kein großes Hintergrundwissen über Computersicherheit.

Das Komplizierteste ist aber vor allem, Anforderungen an die Sicherheit von Systemen zu stellen. Dieser Aspekt wird bereits von den Kunden vernachlässigt, da sie sich darüber zunächst keine Gedanken machen oder nicht wissen wie dieses Problem zu bewältigen ist. Kunden sind, selbst wenn sie über ihre Anforderungen im Klaren sind, oft nicht in der Lage, diese verständlich und eindeutig zu äußern [2]. Diese Gründe führen dazu, dass Sicherheitsanforderungen oft in den Hintergrund geraten oder gar nicht berücksichtigt werden.

Sicherheit wird oft nachträglich beachtet, da viele Mängel in sicherheitskritischen Systemen erst nach der Veröffentlichung und Einsatz dieser Systeme entdeckt werden. Um neu bekannt gewordene Sicherheitslöscher zu schließen, werden meistens Patches verbreitet. Der Nachteil ist, dass die anonymen Zugriffe bereits einen erheblichen Schaden angerichtet haben und andere unseriösen Nutzer dazu motiviert haben nach anderen Sicherheitslöschern zu suchen, da bereits welche gefunden wurden.

Eine weitere Lösung bietet der Einsatz von Formalen Methoden, wo versucht wird, durch mathematische Konzepte, wie z.B. die „Process Algebra CSP" [2], sichere Software zu entwickeln. Der Nachteil dieser Methode ist der hohe Zeitaufwand und das hierfür benötigte, hoch qualifizierte Personal.

Der Sicherheitsstandard „Common Criteria" (CC)[1] ist ein internationaler Standard über die Kriterien zur Bewertung und Zertifizierung der Sicherheit von Computersystemen im Hinblick auf Datensicherheit. Common Criteria bietet eine große Menge von Sicherheitsfachwissen und Richtlinien. Diese Richtlinien sind nicht explizit spezifiziert, sondern eher in den Sprachausdrücken der Sicherheitsdomänen verborgen, sodass Nicht-Sicherheitsexperten davon keinen Gebrauch machen können.

[1] http://www.bsi.bund.de/cc/

Eine Hilfestellung beim Systementwurf für Nicht-Sicherheitsexperten bietet die UML-Erweiterung UMLsec [3]. Nichtsdestotrotz besteht weiterhin das Problem, die nötigen Informationen zu sammeln, die dafür nötig sind, um die Anforderungen einfach mit UMLsec zu modellieren.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist, eine Unterstützung bei der Entwicklung von sicheren Informationssystemen in der Anforderungs- und Entwurfsphase zu bieten. Diese Unterstützung ist an Entwickler gerichtet, die keine Sicherheitsexperten sind, um denen die Möglichkeit zu bieten, Sicherheitsanforderungen wie Geheimhaltung, Integrität und Verfügbarkeit von frühen Entwurfsphasen an in den Entwicklungsprozess zu binden.

Bei der Anforderungserhebung sollen heuristische Ansätze für die Aufdeckung von sicherheitskritischen Anforderungen formuliert werden. Dabei sollen Konzepte und semantische Regeln entwickelt werden, mit deren Hilfe man den sicherheitskritischen Anforderungen auf die Spur kommen kann, wie z.B. die Definition von typischen Schlüsselwörtern für sicherheitsrelevante Ausdrücke und Muster von Ausdrücken und der Einsatz von Methoden aus dem Bereich Information Retrieval und Data Mining. Die Heuristiken sollen bewertet und auf Effizienz untersucht werden.

Am Fachgebiet Software Engineering der Leibniz Universität Hannover werden bereits heuristische Ansätze für die Formulierung von funktionalen Anforderungen verwendet, die beispielsweise eine Warnung auslösen, wenn eine Anforderung im Passiv ausgedrückt wird oder Universalquantoren enthält [4,5,6,7]. Diese Idee soll nun auch speziell auf sicherheitskritische Anforderungen angewandt werden. Es existiert bereits ein Werkzeug (HeRA) für heuristische Unterstützung von Anforderungen. Dieses Werkzeug ist ebenfalls auf den speziellen Fall der sicherheitskritischen Anforderungen anzuwenden und für darüber hinausreichende, neue Konzepte zu erweitern.

Die UML Erweiterung UMLsec erlaubt es, sicherheitsrelevante Informationen in UML-Diagrammen festzuhalten. Bei der Anforderungserhebung mit HeRA sollen Heuristiken für die Analyse und Verfeinerung von abstrakten Anforderungen entwickelt werden, sodass alle Anforderungen in einer Detailstufe vorliegen, die es möglich macht, aus den vorliegenden Anforderungen UMLsec-Modelle zu erstellen. Weiterhin soll die automatische Erstellung von UMLsec-Diagrammen unterstützt werden.

1.3 Gliederung

Im ersten Kapitel wurden die Probleme und die Konsequenzen mangelnder Sicherheit an Informationssystemen diskutiert und die Ansätze dieser Arbeit zur Unterstützung bei der Entwicklung von sicherheitskritischen Systemen vorgestellt. Um die Modellierung von Sicherheitsanforderungen zu erläutern, bietet das zweite Kapitel einen Einstieg in die Themen Anforderungen, UML sowie die Modellierung von Sicherheitsaspekten mit der UML-Erweiterung UMLsec. Anschließend wird im dritten Kapitel näher auf den Begriff der Heuristik eingegangen und Metriken aus dem Bereich Information Retrieval und Data Mining für die Bewertung der Heuristiken vorgestellt.

Somit beinhalten die ersten drei Kapiteln Grundlangen und das nötige Wissen um die darauf folgenden Kapiteln besser nachvollziehen zu können.

Die im Zuge dieser Masterarbeit erarbeiteten Heuristiken, die eine Menge von Anforderungen in sicherheitskritische und nicht-sicherheitskritische Anforderungen klassifizieren, werden in den Kapiteln vier und fünf erläutert. Kapitel vier beschreibt die statischen und Kapitel 5 die dynamischen Klassifizierungsheuristiken.

Kapitel sechs diskutiert Ansätze, mit denen man anfängliche sicherheitskritische Anforderungen auf hoher Abstraktionsebene schrittweise zu einer Menge von Sicherheitsanforderungen verfeinert.

Die Informationsflüsse im Entwicklungsprozess werden Thema des siebten Kapitels sein.

Anschließend wird im achten Kapitel die Implementierung der Klassifizierungsheuristiken und die Verfeinerung von Anforderungen in HeRA vorgestellt. Thema des neunten Kapitels wird die Diskussion der ähnlichen Arbeiten sein. Abschließend werden die wichtigsten Punkte dieser Arbeit und den Ausblick auf zukünftige Arbeiten in Kapitel zehn zusammengefasst.

2. Grundlagen

Dieses Kapitel gibt eine Einführung in das Thema Anforderungen in Softwareprojekten und geht anschließend speziell auf das Thema Sicherheitsanforderung ein. Des Weiteren werden der Systementwurf mittels UML und die UML-Erweiterung UMLsec für die Modellierung von Sicherheitsaspekten erläutert.

2.1 Anforderungen in Softwareprojekten

Nach IEEE 610.12-1990 wird der Begriff „Anforderung“ wie folgt definiert:

Eine Anforderung ist eine Eigenschaft oder Bedingung, die für die Lösung eines Problems oder das Erreichen eines Ziels von einem System oder einer Person verwendet wird. Weiterhin ist eine Anforderung eine Bedingung oder Eigenschaft, die ein System oder ein Teil des Systems aufweisen muss, um einen Vertrag zu erfüllen oder einem Standard oder einer Spezifikation zu genügen. Die Dokumentation einer Eigenschaft oder einer Bedingungen wird ebenfalls als Anforderung betrachtet.

Demzufolge stellen Anforderungen sowohl Wünsche und Ziele der Benutzer, die das zu entwickelnde System verwenden werden, als auch Eigenschaften und Bedingungen des Systems, die beispielsweise aus gesetzlichen oder organisatorischen Gründen zu erfüllen sind.

Die Anforderungen eines Softwareprojekts sind von zentraler Bedeutung, da sie die Grundlage für alle weiteren Schritte im Entwicklungsprozess darstellen. Es gibt unterschiedliche Arten von Anforderungen. Am verbreitetsten ist die Unterteilung in funktionalen und nichtfunktionalen Anforderungen:

- **Funktionale Anforderungen:** Beschreiben die Funktionen und Services, die ein System oder eine Systemkomponente den Nutzern (Personen oder andere Systeme) zur Verfügung stellt. In einer funktionalen Anforderung werden die Eingaben, Ausgaben sowie bekannte Ausnahmen detailliert beschrieben.
- **Nichtfunktionale Anforderungen:** Beschreiben Anforderungen an das System, die nichtfachlicher Natur sind, jedoch entscheidend zur Anwendbarkeit des Systems beitragen. Sie berühren die tatsächliche Funktionalität des Systems nicht. Ziel ist es, klare Qualitätsvorgaben an die unterstützten Geschäftsprozesse und Benutzeraufgaben zu erheben und schließlich umsetzen zu können. Es wird zwischen verschiedenen Arten von nichtfunktionalen Anforderungen an ein System unterschieden. Im Rahmen des ISO Standards 9126 wurden unter Anderem folgende Typen von nichtfunktionalen Anforderungen identifiziert:
    - **Performanzanforderungen:** Beschreiben beispielsweise die Soll-Antwortzeiten aus Sicht des Benutzers oder die Anzahl verarbeitbarer Geschäftsprozesse in einem bestimmten Zeitraum.
    - **Bedienbarkeitsanforderungen:** Beschreiben in der Regel Anforderungen an die Schnittstelle eines Systems zum Benutzer, beispielsweise die Oberfläche einer Onlineanwendung. Das Ziel ist, dass Benutzer die gewünschten Funktionen möglichst intuitiv verwenden können.
    - **Sicherheitsanforderungen:** Anforderungen, die erfüllt werden müssen, damit das System als sicher gilt. Wann ein System die Eigenschaft „sicher“ besitzt, muss im Einzelfall festgelegt werden.

Die Anforderungen an ein System sind dynamisch und können sich im Laufe des Entwicklungsprozess ändern. Einerseits können Kunden Ihre Wünsche ändern, andererseits bestehen zwischen Anforderungen und die darauf folgenden Schritte im Entwicklungsprozess Wechselwirkungen, die dazu führen können, neue Anforderungen zu definieren oder vorhandene Anforderungen zu verfeinern. So können beispielsweise Erkenntnisse im Architekturentwurf zur Modifikation oder Konkretisierung von Anforderungen führen [8].

2.2 Sicherheitsanforderungen

In dieser Arbeit wird zwischen den Begriffen „Sicherheitsanforderung“ und „Sicherheitskritische Anforderung“ unterschieden.

> **Definition**: **Sicherheitsanforderungen** sind funktionale Anforderungen an Systeme oder Komponenten/Prozesse dieser Systeme, die Gefahren verursachen oder Risiken minimieren. Sie dienen als Voraussetzung dafür, dass ein System als sicher gilt und durch externe Angriffe nicht beeinträchtigt wird.

> **Definition**: **Sicherheitskritische Anforderungen** sind potentielle Anforderungen auf abstrakter Ebene, deren Verfeinerung zu Sicherheitsanforderungen führen kann.

Kunden fixieren meistens ihr Interesse an die funktionalen Anforderungen und haben i.d.R. keine oder sehr geringe Vorstellung von Sicherheitsanforderungen. Ein weiterer Punkt ist, dass Sicherheitsanforderungen oft in Konflikt mit anderen Anforderungen an die Software stehen, da eine als sicher zu implementierende Software oft nicht einfach mit anderen Software zu integrieren ist, bietet weniger Funktionalität oder ist langsamer als eine Software, die Sicherheitsaspekte nicht beachtet. Deshalb scheint die Entwicklung von sicheren Systemen zunächst aufwändiger. Dies ist aber nur dann der Fall, wenn der Aufwand für die Beseitigungen eventuell im Nachhinein entdeckten Sicherheitslücken nicht berücksichtigt wird.

Mit dem Begriff "Sicherheit" werden im Deutschen zwei unterschiedliche Systemeigenschaften bezeichnet, die im Englischen klarer unterschieden werden:

- **Safety (Betriebssicherheit):** Bezeichnet den Schutz der Umgebung vor schädlichen Systemeinflüssen.
- **Security (Angriffssicherheit):** Bezeichnet den Schutz des Systems und seiner Daten vor Bedrohungen durch eine feindliche Umgebung.

Weiterhin wird Sicherheit Nach [27] aus zwei unterschiedlichen Sichtweisen betrachtet:

- **Objektiv:** Das Nichtvorhandensein von Gefahr
- **Subjektiv:** Die Gewissheit, vor möglichen Gefahren geschützt zu sein.

Der Begriff Sicherheit in dieser Arbeit bezieht sich auf Angriffssicherheit (Security) aus der subjektiven Betrachtungssicht.

Es werden unterschiedliche Anforderungen bezüglich Sicherheit an Informationssystemen gestellt. Es lassen sich folgende Anforderungsarten unterscheiden [28]:

Zugriffssicherheit:

- **Authentifikation:** Eindeutige Bestimmung der Identität. Die Überprüfung der Identität wird in der Regel mittels Benutzername und Passwort realisiert.
- **Autorisierung:** Es muss festgelegt werden, welche Benutzer oder Prozesse welche Aktionen auf welche Daten durchführen dürfen. Die Vergabe von Aktionsrechten (Au-

torisierung) ist nur mit einer vorher durchgeführten Identifikation (Authentifikation) sinnvoll.

- **Anonymität:** Schutz personenbezogener Daten vor Missbrauch. Informationen, die einen Benutzer eindeutig identifizieren und die Veröffentlichung von diesem nicht erwünscht ist, müssen in anonymisierter Form verwendet werden.

Datensicherheit:

- **Vertraulichkeit:** Schutz personenbezogener Daten vor Diebstahl und Missbrauch. Der Zugriff auf Informationen soll nicht jedem Benutzer oder Prozess im System gewährleistet werden. Es muss vor jedem Zugriff auf vertrauliche Informationen überprüft werden, ob der Zugriff erlaubt ist. Um dies zu realisieren, werden oft Mechanismen der Kryptographie eingesetzt.
- **Datenintegrität:** Modifizierung von Daten ist lediglich durch legitimierten Personen und Prozessen möglich.
- **Verbindlichkeit:** Eine von einem Benutzer durchgeführte Aktion kann nicht rückgängig gemacht werden. Dies spielt beispielsweise eine wichtige Rolle bei der Unterzeichnung von elektronischen Verträgen.
- **Verfügbarkeit:** Wahrscheinlichkeit, ein System zu einem gegebenen Zeitpunkt in einem funktionsfähigen Zustand anzutreffen (DIN 40042).
- **Sicherstellung der Zustellung:** Innerhalb eines Systems muss sichergestellt werden, dass beim Datenaustausch die Daten nicht nur ankommen, sonder auch bei der richtigen Identität ankommen. Sender und Empfänger müssen sicher sein, dass das Gegenüber die Identität besitzt, die angenommen wird.

Systemsicherheit:

- **Monitoring und Traceability:** Durchgeführte Aktionen müssen nachweisbar und überprüfbar sein. Es muss möglich sein, eine durchgeführte Aktion einer Zeit und einem Akteur zuzuordnen. Man spricht auch von Logging oder Autditing. Somit kann im Nachhinein im Falle eines Problems zurückverfolgt werden, welche Benutzer wann welche Aktionen durchgeführt haben.
- **Mehrstufige Sicherheit:** Bedeutet, dass nicht alle Informationen gleich sicherheitskritisch sind und entsprechend verschiedene Mechanismen zur Sicherstellung der Sicherheit eingesetzt werden.

Systeme mit hohen Sicherheitsanforderungen werden als kritische Systeme bezeichnet. Hohe Sicherheitsanforderungen werden an Systemen gestellt, deren Fehlfunktion zu einer der folgenden Konsequenzen führen könnte:

- Verlust von Menschenleben
- Verletzung oder Gesundheitsgefährdung von Menschen
- Schwerwiegende Schädigung der Umgebung
- Bedeutender Verlust der Systemeigenschaft bzw. Systemzerstörung
- Nichterfüllung einer wichtigen Aufgabe
- Großer wirtschaftlicher Verlust

Werden die Anforderungen von kritischen Systemen nicht oder nicht in ausreichender Qualität erfüllt, so besteht das Risiko, dass eine oder mehrere der oben genannten Konsequenzen auftreten. Deshalb ist es wichtig, die Sicherheitsanforderungen möglichst früh im Entwick-

lungsprozess zu erkennen und mit einem geeigneten Systementwurf umzusetzen und korrekt zu implementieren [9].

2.3 Systementwurf mit UML

Die Unified Modeling Language (UML) ist eine von der Object Management Group (OMG)[2] entwickelte und standardisierte Modellierungssprache zur Spezifikation, Visualisierung, Konstruktion und Dokumentation von Modellen für Softwaresysteme. UML besteht aus einem Satz von Notationen zur Formung einer allgemeinen Sprache zur Softwareentwicklung. UML stellt keine Methode dar, die Empfehlungen zur Vorgehensweise bei Entwicklungsprozessen anbietet. Die Modellelemente der UML werden nach Diagrammtypen unterschieden. Mit UML können sowohl Modelle von statischen Strukturen (Strukturdiagramme) als auch von dynamischen Abläufen (Verhaltensdiagramme) erstellt werden.

Zu den Strukturdiagrammen gehören folgende Diagrammtypen:

- **Klassendiagramm (class diagram):** Mit dem Klassendiagramm wird der Aufbau der Klassen, Schnittstellen sowie deren Beziehungen beschrieben.
- **Objektdiagramm (object diagram):** Das Objektdiagramm stellt die existierenden Objekte und deren Zusammenhänge zu einem bestimmten Zeitpunkt während der Systemlaufzeit dar. Da Objekte im Gegensatz zu Klassen während der Laufzeit erstellt und zerstört werden können, kann ein Objektdiagramm als eine Momentaufnahme betrachtet werden. Ein Klassendiagramm ist im Gegensatz dazu zeitlos.
- **Paketdiagramm (package diagram):** Ein Paket ist eine Gruppierung von Klassen oder Schnittstellten. Ähnlich wie beim Klassendiagramm werden im Paktdiagramm die Pakete und deren Beziehungen dargestellt.
- **Komponentendiagramm (component diagram):** Eine Komponente besteht aus mehreren Einheiten und stellt die nächst höhere Ebene über den Klassen dar. Es zeigt eine Sicht auf die Struktur des modellierten Systems und wie Komponenten über Abhängigkeitsbeziehungen miteinander verbunden sind.
- **Kompositionsstrukturdiagramm (composite structure diagram):** Zeigt die interne Struktur von Kompositionen und deren Wechselwirkungen mit der Umgebung.
- **Verteilungsdiagramm (deployment diagram):** Wird für die Modellierung von verteilten Systemen verwendet. Es gibt eine Übersicht, welche Programme auf welchen Systemen ausgeführt werden und wie diese miteinander kommunizieren.

Zu den Verhaltensdiagrammen gehören folgende Diagrammtypen:

- **Use-Case-Diagramm (use case diagram):** Beschreibt die Anforderungen an eine Software und wie die Anwender mit der Software interagieren.
- **Aktivitätsdiagramm (activity diagram):** Beschreibt den Ablauf von Aktionen. Es stellt die Aktionen und deren Verbindungen mit Datenflüssen grafisch dar.
- **Zustandsdiagramm (state diagram):** Beschreibt den Zustandswechsel mittels Zustandsautomaten und wird zur Spezifikation des Verhaltens eines Systems eingesetzt.
- **Sequenzdiagramm (sequence diagram):** Stellt die Abläufe und den Austausch von Nachrichten zwischen mehreren miteinander interagierenden Objekten mittels Lebenslinien dar.

[2] http://www.omg.org

- **Kommunikationsdiagramm (communication diagram):** Ähnlich wie das Sequenzdiagramm, wobei nicht die Abläufe zwischen interagierenden Objekten, sondern die interagierenden Objekten selbst im Mittelpunkt stehen. Das Kommunikationsdiagramm ist auch unter dem Namen Kollaborationsdiagramm bekannt.
- **Timing-Diagramm (timing diagram):** Beschreibt nicht nur die Reihenfolge von Abläufen, sondern auch die Zeitangaben der Abläufe zwischen interagierenden Objekten.
- **Interaktionsübersichtsdiagramm (interaction overview diagram):** Kombiniert mehrere Verhaltensdiagramme um deren Zusammenspiel darzustellen.

Die verschiedenen Diagrammtypen von UML bieten eine Möglichkeit, die Software aus unterschiedlichen Blickwinkeln zu betrachten und auf diese Weise die Struktur und das Verhalten der Software übersichtlicher zu gestalten und besser zu verstehen.

Ein Vorteil von UML ist, dass der Sprachumfang von Programmiersprachen und Plattformen unabhängig ist. Die mit UML erstellten Modelle der Softwaresysteme sind somit ebenso programmier- und plattformunabhängig und müssen bei Änderung der eingesetzten Technologien nicht verändert werden [10].

2.3.1 UML-Erweiterung

Mit den Standard-Elementen von UML können keine Sicherheitsaspekte modelliert werden. Es ist jedoch möglich, UML zu erweitern, d.h., neue Klassen oder Attribute hinzuzufügen, ohne das bestehende Metamodell zu entfernen oder zu erweitern [11].

UML bietet einen Erweiterungsmechanismus durch Profile, mit dem die Modellierung von spezifischen Einsatzgebieten ermöglicht wird. Ein Profil ist eine Menge von Stereotypen, Eigenschaftswerten und Einschränkungen:

- **Stereotypen (stereotyp):** Mit Stereotypen werden neue Typen von Modellierungselementen definiert, welche die Semantik von bestehenden Typen oder Klassen erweitern. Ein stereotypisiertes Element eines Modells wird mit dem Namen des Stereotyps eingeschlossen zwischen Guillements dargestellt: «stereotyp».
- **Eigenschaftswerte (tagged values):** Ein Element eines Modells kann mit Eigenschaftswerten ausgezeichnet werden. Ein Eigenschaftswert ist ein Paar aus einem Namen und einem zugewiesenen Wert. Diese Zuweisung wird in geschweiften Klammern eingeschlossen: {Name = Wert}.
- **Einschränkungen (constrains):** Einschränkungen stellen eine andere Möglichkeit dar, um Informationen einem Element hinzuzufügen. Dadurch wird die Semantik des Elements verfeinert [12].

2.4. Entwurf sicherheitskritischer Systeme mit UMLsec

In diesem Abschnitt wird UMLsec, eine Erweiterung von UML zur Entwicklung sicherer Softwaresysteme, vorgestellt.

2.4.1 Das UMLsec-Profil

UMLsec [12] stellt eine Methode zur Modellierung von Sicherheitsaspekten mit UML dar. Sicherheitsanforderungen werden als Stereotypen mit Eigenschaftswerten dargestellt und durch Einschränkungen überprüft. Dadurch können vereinbarte Sicherheitsanforderungen durch gegebene Sicherheitsregeln erfüllt werden. Das UMLsec-Profil besteht aus Stereotypen, Eigenschaftswerten und Einschränkungen. Mittel eines UMLsec-Profils werden Sicherheits-

aspekte in einem UML-Diagramm gekennzeichnet und hervorgehoben, damit diese bei der Implementierung berücksichtigt werden. Somit tragen die UMLsec-Profile zur Entwicklung sicherer Software bei. Die UMLsec-Profile werden komplett in Tabellen 2.1 dargestellt.

Stereotyp	Basisklasse	Eigen-schaftswert	Einschränkungen	Beschreibung
fair exchange	subsystem	start, stop, adversary	nach „start“, wird schließlich „stop“ erreicht	erzwingt einen fairen Austausch
provable	subsystem	action, cert, adversary	“action” ist nicht zurückweisbar	Voraussetzung für Nicht-Zurückweisbarkeit
rbac	subsystem	protected, role, right	nur gewährte Aktionen werden ausgeführt	Erzwingt rollenbasierte Zugriffskontrolle
Internet	Link			Unverschlüsselte Internet-Verbindung
encrypted	Link			verschlüsselte Verbindung
LAN	link, node			LAN-Verbindung
wire	Link			Wireless-Verbindung
smart card	Node			Chipkarte-Knoten
POS device	Node			“POS”-Gerät
issuer node	Node			Ausgabeknoten
secrecy	dependency			erzwingt eingeschränktes Leserecht
integrity	dependency			erzwingt eingeschränktes Änderungsrecht
high	dependency			erzwingt eingeschränktes Lese-, Änder- und Löschrecht.
critical	object, sub-system	secrecy, integ-rity, authentic-ity, high, fresh		ein kritisches Objekt bei dem auf den Informationsfluss geachtet werden muss
secure links	subsystem	adversary	Abhänigkeit security passt mit den links zusammen	erzwingt sichere Kommunikations-verbindungen
secure depen-dency	subsystem		<<call>>, <<send>> beachten data security	Interaktionen beachten die Datensicherheit
data security	subsystem	adversary, integrity, au-thenticity	unterstützt secrecy, integrity, authenticity, freshness	Datensicherheitsanforderun-gen
no downflow	subsystem		verhindert “downflows”	Bedingung für Informationen-fluss
no up-flow	subsystem		verhindert “upflows”	Bedingung für Informationen-fluss
guarded ac-cess	subsystem		“guarded objects” sind nur durch “guards” erreichbar	Zugriffskontrolle mit „guard“-Objekte
guarded	Object	guard		Beschütze Objekte

Tabelle 2.1: UMLsec-Stereotypen [12]

Sicherheitsanforderungen können mittels Use-Case-Diagrammen dargestellt werden. Für die Sicherstellung, dass der Datenaustausch den Sicherheitslevel einhält, werden Klassen- bzw. Komponentendiagramme eingesetzt. Für die Beschreibung von sicherheitskritischen Interaktionen bieten sich Sequenzdiagamme an. Zustandsdiagramme werden für die Einhaltung der Sicherheitsbestimmungen innerhalb der Objekte benutzt. Somit umfasst UMLsec den gesamten Entwurfsprozess.

2.4.2 Anwendungsbeispiel von UMLsec

Das UMLsec-Konzept wird hier durch ein Beispiel für den Stereotyp „fair exchange“ veranschaulicht. Dieser Stereotyp beschreibt die Sicherheitsanforderungen einer Transaktion zwischen Handelspartnern, sodass ein betrügerisches Verhalten verhindert wird. Das Beispiel betrachtet eine Online-Verkaufsanwendung mit folgenden Anforderungen:

- Persönliche Kundendaten sowie Transaktionsdaten sollen nicht an Dritte gelangen
- Bestellung erfolgt über das Internet
- Waren werden nach Bezahlung per Post verschickt
- Erhält der Kunde die Ware zu einem Fälligkeitsdatum nicht, so hat er die Möglichkeit, die Bestellung zu stornieren und seine Zahlung zurück zu fordern
- Nach Erhalt der Ware hat der Kunde keine Möglichkeit die Zahlung zurück zu fordern
- Kunden mit einem Einkaufswert größer als 1000 € werden Extraleistungen gewährt

Die Verkaufsanwendung wird in der folgenden Abbildung grob beschrieben. Die Sicherheitsanforderungen sind in den Prozessen „buys good“ und „sells good“ enthalten.

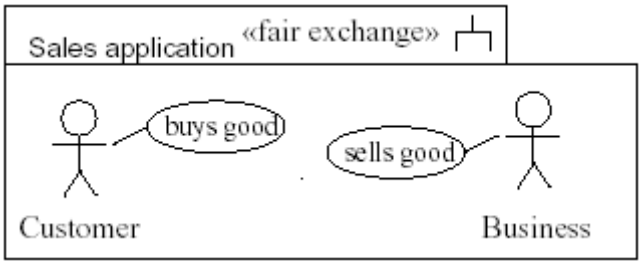


Abbildung 2.1: Use-Case-Diagramm für die Verkaufsanwendung [12]

Die Aktivitäten in einem Diagramm können durch ein anderes Diagramm - unter der Voraussetzung dass letzteres mit dem gleichen Stereotyp markiert ist - verfeinert werden. Es ist anzumerken, dass die Beschreibung des Stereotyps „fair exchange“ sowie die Verfeinerung in diesem Beispiel lediglich informell erfolgt. Das obige Use-Case-Diagramm kann nun durch ein Aktivitätsdiagramm mit dem Stereotyp „fair exchange“ verfeinert werden um spezifische Einschränkungen genauer zu beschreiben. Das folgende Aktivitätsdiagramm beschreibt die Aktivitäten der Käufer und Verkäufer detaillierter.

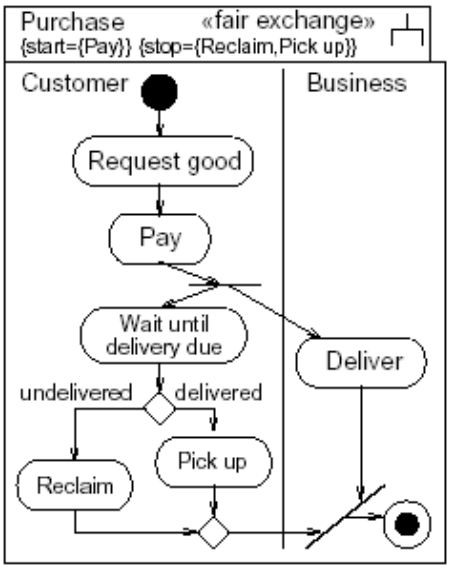


Abbildung 2.2: Aktivitätsdiagramm für die Verkaufsanwendung [12]

Der Eigenschaftswert {start = pay}beschreibt den Beginn der Transaktion. Der Eigenschaftswert {stop = {reclaim, pick up}} versichert, dass der Kunde seine Bestellung erhält oder die Bestellung stornieren kann, falls diese nach dem Lieferungsdatum noch nicht eingetroffen ist. Weiterhin ist für den Verkäufer sichergestellt, dass die Lieferung nur dann erfolgt, wenn die Zahlung bereits getätigt ist und dass die Bestellung nach Annahme der Ware nicht mehr reklamiert werden kann.

Bis jetzt wurden die Sicherheitsanforderungen an die Anwendung auf der logischen Ebene beschrieben. Um diese nun auf der physikalischen Schicht zu modellieren, werden Verteilungsdiagramme verwendet. Zusätzlich wird überprüft, ob die Sicherheitsanforderungen auf der logischen Schicht auch auf der physikalischen erfüllt sind oder ob weitere Sicherheitsmaßnahmen ergriffen werden müssen, wie z.B. die Kodierung der Daten.

Um mit dem obigen Beispiel fort zu fahren, sei die Verkaufsanwendung teil eines eCommerce-Systems, das als Webanwendung realisiert werden soll. Die Bezahlungstransaktion bedingt die Übermittlung von Daten über eine Internetverbindung. Diese Daten, wie z.B. die Kreditkartennummer, müssen geheim gehalten werden. Diese Sicherheitsanforderung auf der physikalischen Schicht wird durch den Stereotyp „secure links“ des Subsystems und dem Stereotyp „secrecy“ für die Kommunikationsverbindung modelliert. Abbildung 2.3 stellt das Verteilungsdiagramm dar.

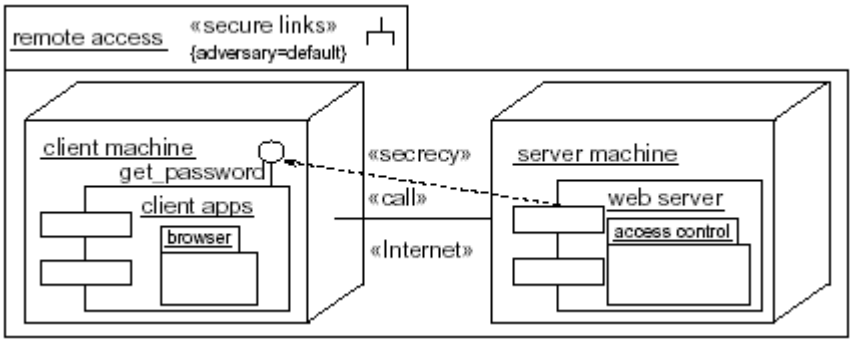


Abbildung 2.3: Verteilungsdiagramm für das eCommerce-System [12]

Um die Interaktionen zwischen den verschiedenen Teilen des Systems zu modellieren, werden Sequenzdiagramme eingesetzt. Diese können mit UMLsec-Stereotypen erweitert werden um relevante Sicherheitsanforderung bezüglich der Interaktion zu modellieren.

Im Hinblick auf unser Beispiel soll nun ein sicherer Kommunikationskanal entworfen werden, um die sensitiven Daten über das unsichere Netz zu übertragen. Für diesen Zweck soll von Verschlüsselung Gebrauch gemacht werden. Es sei angenommen, dass z.B. aus technischen Gründen, nicht ein Standardprotokoll wie SSL, sondern ein kundenspezifisches Protokoll verwendet wird. Das Ziel ist, einen geheimen Schlüssel (secret session key) K, unter Verwendung von öffentlichen Schlüsseln (public keys) K_C und K_S auszutauschen, der dann für die Verschlüsselung der zu schützenden Daten s vor der Übermittlung verwendet wird.

Sei $\{M\}_K$ die Verschlüsselung der Nachricht M mit dem Schlüssel K, $Sign_K(M)$ die Signatur der Nachricht M mit K und :: die Kennzeichnung von Konkatenation, dann wird der Austausch von kritischen Daten mittels des Stereotyps „critical“ wie in Abbildung 2.4 modelliert.

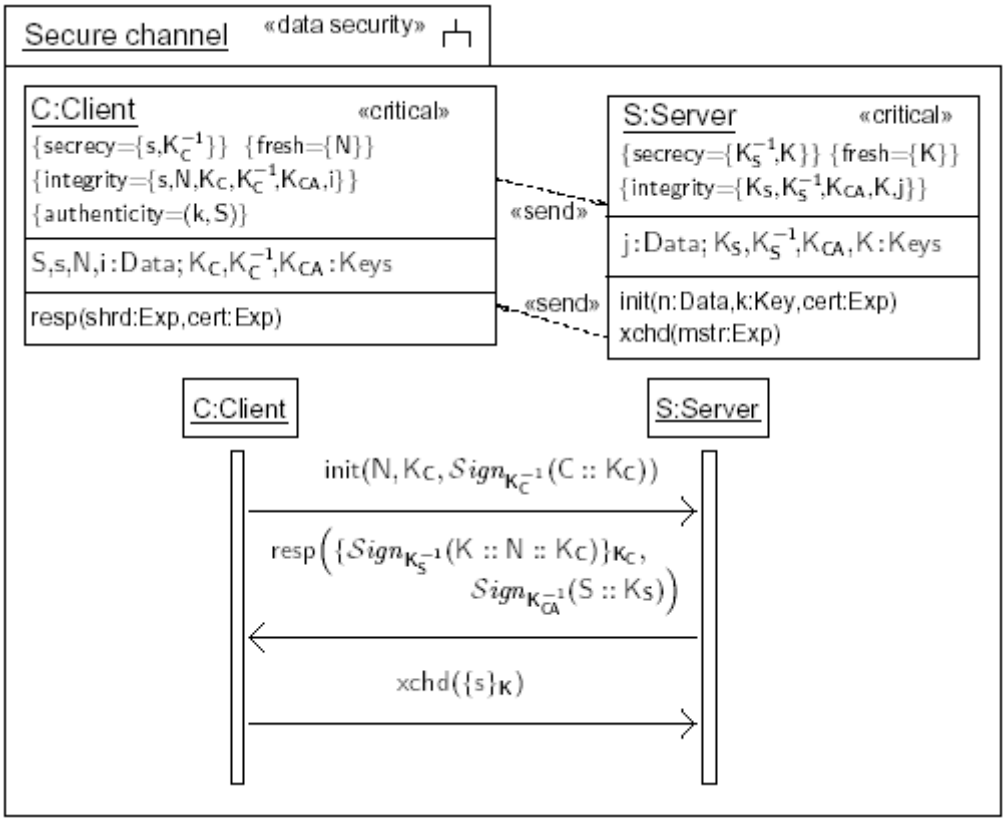


Abbildung 2.4: Sequenzdiagramm für das Schlüssel-Austausch-Protokoll [12]

Der Stereotyp „critical“ kennzeichnet Klassen, die sensitive Daten enthalten und besitzt die Eigenschaftswerte {secrecy}, {integrity}, {authenticity} und {fresh}um die betreffenden Sicherheitsanforderungen an die Daten darzustellen.

Des Weiteren soll nun die Anforderung an die Verkaufsanwendung, dass Kunden mit einem Einkaufswert größer als 1000 Euro eine Extraleistung gewährt wird, betrachtet werden. Aus Vertraulichkeitsgründen soll die Gesamteinkaufsumme des Kunden geheim gehalten werden. Abbildung 2.5 veranschaulicht die Spezifikation des Kundenkonto-Objekts.

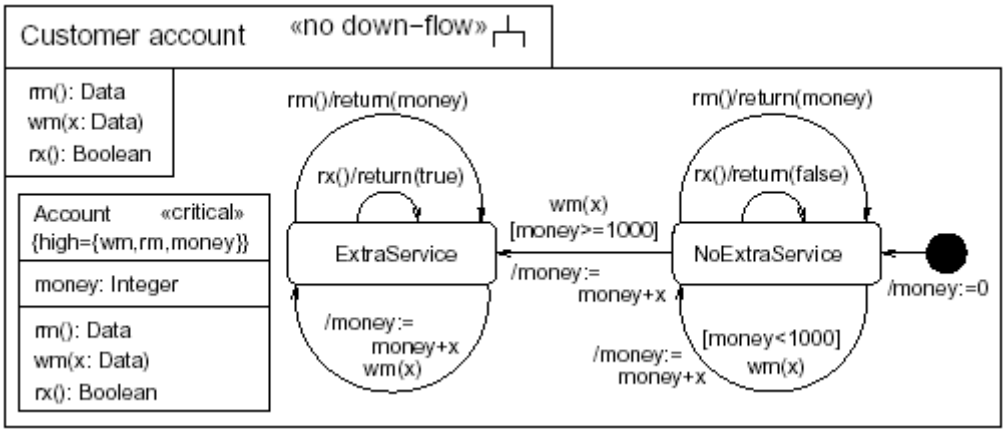


Abbildung 2.5: Zustandsdiagramm für das Kundenkonto-Objekt [12]

Das obige Zustandsdiagramm modelliert den Wechsel des Objektzustands. Der Stereotyp „no down-flow“ weist darauf hin, dass vertrauliche Daten wie z.B. das Attribut „money“ nicht nach Außen gelangen werden dürfen. Mit der Methode rm() wird der Betrag gelesen und mit wm() aktualisiert. Die Methode rx() entscheidet über das Anbieten der Extraleistung [12].

3 Bewertungskriterien für Klassifizierungsheuristiken

Eine Heuristik (altgr.: auffinden, entdecken) ist nach Russel, Norvig [13] eine Methode, die gute, beinahe optimale Lösungen in angemessener Rechenzeit erstrebt, ohne die Realisierbarkeit oder Optimalität zu garantieren.

In dieser Arbeit wird eine Klassifizierungsheuristik wir folgt definiert:

> **Definition:** Eine **Klassifizierungsheuristik** ist ein Verfahren, das auf Basis von bereits gesammeltem Wissen über Sicherheits- und Nicht-Sicherheitsanforderungen, mittels methodischer Gewinnung neuer Erkenntnisse, Anforderungen als sicherheitskritisch oder nicht-sicherheitskritisch klassifiziert, ohne eine optimale Lösung zu garantieren.

Um die Klassifizierungsheuristiken zu bewerten und die Ergebnisse zu analysieren werden zahlreiche Anforderungen aus realen Projekten benötigt. Liegen diese vor, so ist man mit dem Problem konfrontiert, die Anforderungen zunächst manuell zu analysieren und festzulegen, welche als Sicherheitsanforderungen gelten und welche nicht. Die klassifizierten Sicherheitsanforderungen dienen als Referenz für die Beurteilung der Güte der Klassifizierungsheuristiken. Deshalb muss dieser Prozess sehr sorgfältig und möglichst von Sicherheitsexperten durchgeführt werden.

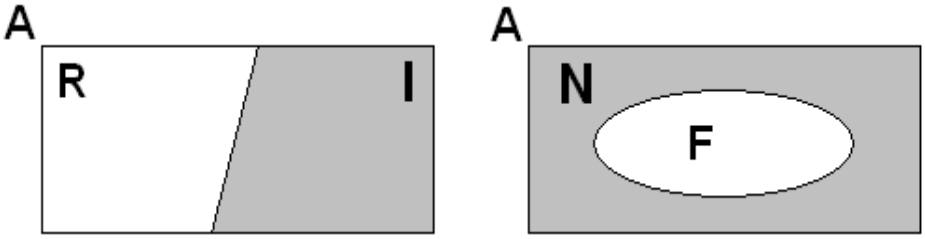
3.1 Recall und Precision

Zur Beurteilung der Güte der Klassifizierungsverfahren werden die bekannten Maße Recall, Precision und das F-Maß aus dem Bereich Information Retrieval verwendet [14].

Sei A (all) die Menge aller Anforderungen, R (relevant) die Menge aller relevanten Anforderungen, d.h. die sicherheitskritischen Anforderungen und I (irrelevant) die Menge der irrelevanten Anforderungen, d.h. die nicht-sicherheitskritischen Anforderungen.

Sei F (found) die Menge der von der Klassifizierungsheuristik als sicherheitskritisch klassifizierten Anforderungen und N (not found) die Menge der von der Klassifizierungsheuristik als nicht-sicherheitskritisch klassifizierten Anforderungen. Es gilt:

$$R \cup I = A$$
$$F \cup N = A$$
$$R \cap I = \varnothing$$
$$F \cap N = \varnothing$$

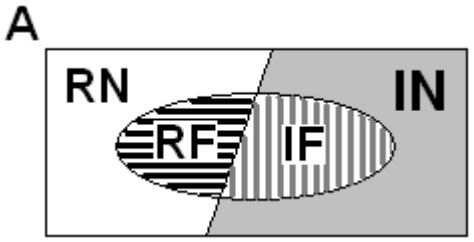


Im Idealfall findet die Klassifizierungsheuristik genau alle relevanten Anforderungen. In der Regel wird die Klassifizierungsheuristik jedoch sowohl relevante als auch irrelevante Anforderungen als relevant klassifizieren.

Sei RF (relevant found) die Menge der relevanten Anforderungen, die von der Klassifizierungsheuristik korrekterweise als relevant klassifiziert werden. Sei RN (relevant not found) die Menge der relevanten Anforderungen, die von der Klassifizierungsheuristik als irrelevant bezeichnet wurden.

Sei IF (irrelevant found) die Menge der irrelevanten Anforderungen, die von der Klassifizierungsheuristik fälschlicherweise als relevant klassifiziert wurden. Sei IN (irrelevant not found) die Menge der irrelevanten Anforderungen, die von der Klassifizierungsheuristik ebenso als irrelevant bezeichnet wurden. Es gilt:

$$RF \cup RN = R$$
$$IF \cup IN = I$$
$$RF \cap RN = \varnothing$$
$$IF \cap IN = \varnothing$$



Der Recall stellt ein Maß für die Vollständigkeit des Suchergebnisses dar und ist definiert als das Verhältnis der Anzahl der relevanten gefundenen Anforderungen zur Gesamtzahl aller relevanten Anforderungen:

$$\text{Recall} = \frac{|\text{R} \cap \text{F}|}{|\text{R}|} = \frac{|\text{RF}|}{|\text{R}|}$$

Der Recall kann Werte zwischen 0 und 1 annehmen. Der Wert 1 wird erreicht, falls alle relevanten Anforderungen gefunden werden. Das Ergebnis ist 0, falls keine der relevanten Anforderungen gefunden wird. Bei der Berechnung des Recall tritt das Problem auf, dass die Menge der relevanten Anforderungen nur selten eindeutig und vollständig definiert werden kann.

Die Precision stellt ein Maß für die Genauigkeit des Suchergebnisses dar und ist definiert als das Verhältnis der Anzahl der relevanten gefundenen Anforderungen zur Gesamtzahl aller gefundenen Anforderungen:

$$\text{Precision} = \frac{|\text{R} \cap \text{F}|}{|\text{F}|} = \frac{|\text{RF}|}{|\text{F}|}$$

Der Recall allein reicht nicht aus, um die Qualität des Ergebnisses festzustellen. Selbst bei einem Recall von 1 kann eine schlechte Qualität vorliegen, falls zu viele irrelevante Anforderungen gefunden wurden. Die Precision stellt zusätzlich ein Maß für die Fähigkeit der Klassifizierungsheuristik, irrelevante Anforderungen ausschließen zu können. Die Precision nimmt den Wert 1 falls alle relevanten Anforderungen gefunden werden und keine der irrelevanten Anforderungen gefunden wird. Die Betrachtung der Precision alleine ist jedoch ebenso problematisch, da diese nicht aussagekräftig ist, falls sehr wenige Anforderungen gefunden werden.

3.2 Das F-Maß

Wie im vorangegangenen Abschnitt erläutert, ist es sinnvoller, der Recall und die Precision immer zusammen zu betrachten. Das F-Maß [14] kombiniert beide Maße mittels des gewichteten harmonischen Mittels und ist folgendermaßen definiert:

$$\text{F} = \frac{2 \cdot (\text{precision} \cdot \text{recall})}{(\text{precision} + \text{recall})}$$

Das F-Maß nimmt den bestmöglichen Wert 1 wenn sowohl Recall als auch Precision den Wert 1 annehmen.

Zur graphischen Betrachtung von Recall und Precision kann das Recall-Precision-Diagramm verwendet werden. Die Werte von Precision werden auf der x-Achse, die von Recall auf der y-Achse eingetragen. Dabei wird sichtbar, dass i.A. hoher Recall zu sinkender Precision führt und umgekehrt [14].

Interessiert man sich für eine große Ergebnismenge und nimmt man in Kauf, dass möglicherweise einige Ergebnisse nicht korrekt sind, so kann man nur Recall betrachten. Steht die Güte der Ergebnismenge im Mittelpunkt, unabhängig von deren Größe, so kann nur Precision betrachtet werden. Sind sowohl Größe und Qualität der Ergebnisse von Interesse, so muss man das F-Maß betrachten, weil es sowohl Recall als auch Precision berücksichtigt.

4. Statische Klassifizierungsheuristiken

Im Folgenden werden einige Heuristiken für die Aufdeckung von sicherheitskritischen Anforderungen, die im Rahmen dieser Arbeit entwickelt wurden, vorgestellt. Die Heuristiken basieren auf der Suche nach bestimmten Schlüsselwörtern und Synonymen von diesen. Zusätzlich werden die Bildung des Wortstammes und das Zusammentreten von bestimmten Wörtern betrachtet. Diese Heuristiken werden statisch genannt, weil die Schlüsselwörter und Synonyme fest definiert sind und die Heuristik nicht selbständig auf die Erfahrung des Nutzers reagieren und von dieser lernen kann um die Suchergebnisse zu verbessern.

Um relevante Begriffe zu bestimmen, die mögliche sicherheitskritische Anforderungen identifizieren könnten, wurden Anforderungen von einem Verwaltungstool für Studentenarbeiten sowie Anforderungen eines Content-Management-Systems analysiert. Diese Anforderungsspezifikationen enthalten jedoch keine Sicherheitsanforderungen, die alle in Tabelle 3.1 aufgeführten UMLsec-Stereotypen abdecken. Deshalb wurden für die Analyse und die Auswertung die Stereotypen „secrecy“, „integrity“, „high“ und „fair exchange“ ausgewählt.

4.1 Suche nach Schlüsselwörtern

Die Suche nach Schlüsselwörtern ist sehr intuitiv und basiert auf der Suche nach festgelegten Begriffen in der Anforderungsbeschreibung. Eine Anforderung wird als sicherheitskritisch klassifiziert, falls einer der festgelegten Begriffe in der Anforderungsbeschreibung vorkommt.

Die Begriffe werden zusätzlich einem UMLsec-Stereotyp zugeordnet, sodass die als sicherheitskritisch identifizierten Anforderungen ebenso diesem UMLsec-Stereotyp zugeordnet werden. In Tabelle 4.1 werden die Schlüsselwörter und die dazu geordneten Stereotypen dargestellt.

UMLsec-Stereotyp	Schlüsselwörter
secrecy	access | account | add | authorize | content | display | download | globalize | identify | log on | password | publish | register | user
integrity	add | access | allow | block | content | delete | display | download | edit | manage | save | upload | user
high	account number | bank account | credit card | decrypt | encrypt | meta data | password | personal
fair exchange	buy | good | internet | lend | reserve | user

Tabelle 4.1: Schlüsselwörter und dazu gehörige UMLsec-Stereotypen

4.2 Suche nach Synonymen

Diese Klassifizierungsheuristik berücksichtigt neben den festgelegten Schlüsselwörtern zusätzlich deren Synonyme um die Trefferquote bei der Suche zu erhöhen. Die Synonyme können mit Hilfe eines Thesaurus oder manuell festgelegt werden.

4.2.1 Datenbank-Thesaurus

Für die Bestimmung der Synonyme wird der Datenbank-Thesaurus WordNet[3] verwendet.

Der Vorteil eines Datenbank-Thesaurus ist der geringe Aufwand für die Bestimmung der Synonyme, da diese bereits in der Datenbank gespeichert sind und nur noch abgefragt werden

[3] http://wordnet.princeton.edu

müssen. Nachteile sind einerseits, dass für einige Begriffe gar keine oder sehr wenige Synonyme gespeichert sind, andererseits dass die Datenbank ebenso Synonyme enthält, die für den Sicherheitskontext irrelevant sind und somit zu einer fehlerhaften Identifizierung von Sicherheitsanforderungen führen könnten.

Beispielsweise liefert die Datenbank für das Wort „data“ lediglich das Wort „information“ als Synonym. Für das Wort „execute“ liefert die Datenbank „put to death“ als Synonym zurück.

4.2.2 Manuell erstellter Thesaurus

Anstatt ein Datenbank-Thesaurus zu verwenden, können die zu betrachtenden Synonyme manuell festgelegt werden. Diese Vorgehensweise hat den Vorteil, dass möglichst viele Synonyme zu einem Begriff berücksichtigt werden und nur diejenigen Synonyme die für den Sicherheitskontext relevant sind in Betracht genommen werden. Tabelle 4.2 zeigt einige UMLsec-Stereotypen, die dazugehörigen Schlüsselwörter und Synonyme. Eine erweiterte Tabelle befindet sich im Anhang A.

UMLsec-Stereotyp	Schlüsselwort	Synonyme
secrecy	access	call | execute | fetch | query | read | request | retrieve | select
	add	annex | append | apply | attach | copy | create | enclose | include | infix | insert | join | paste | put | write
	authorise	accord | allow | appropriate | approve | concede | empower | grant | license | permit
	data	component | content | details| document | file | page | paper | information | site | statement | text | website
	display	advertise | announce | demonstrate | globalize | look | overview | present | record | report | see | show | view | visualize
integrity	display	advertise | announce | demonstrate | globalize | look | overview | present | record | report | see | show | view | visualize
	edit	change | modify | process | replicate | update
high	credit card	bank card | charge card | master card | plastic | smart card | visa card
	encrypt	cipher | code | encipher | encode | scramble
	personal	private
fair exchange	buy	appoint | aquire | disburse | gain | get | pay | purchase | request | shop
	user	acquirer | actor | buyer | client | consumer | customer | handler | operator | purchaser | shopper | somebody | someone | stakeholder | worker

Tabelle 4.2: Beispiele für Schlüsselwörter und die dazugehörigen Synonyme

4.3 Rückführung auf den Wortstamm (Word Stemming)

Ein Stammwort ist der Bestandteil eines Wortes, der das Grundwort einer Wortfamilie bildet. Vom Stammwort ausgehend können Ableitungen und Komposita gebildet werden. Ein Kompositum ist ein Wort, das mehr als ein lexikalisches Morphem enthält, wobei ein Morphem das kleinste bedeutungstragende Element ist. Aus dem englischen Stammwort „add“ können z.B. folgende Wörter gebildet werden: „adds“, „adding“, „addition“, „additional“ und „additionally“.

Durch die in den vorangegangenen Abschnitten vorgestellten Klassifizierungsheuristiken können Begriffe nicht gefunden werden sobald diese in einer abgeleiteten Form vorkommen. Deshalb wird hier der Suchtext zunächst in die einzelnen Wörtern zerlegt und die Wörter auf ihren Wortstamm zurückgeführt. Die Schlüsselwörter sowie die Synonyme werden ebenfalls auf den Wortstamm reduziert. Mit dieser Erweiterung werden nun auch alle Wörter gefunden, die in einer abgeleiteten Form in dem zu untersuchenden Text vorkommen.

Für die Stammbildung existieren unterschiedliche Algorithmen. In dieser Arbeit wird ein weit verbreiteter Algorithmus verwendet, der Porter-Stemmer-Algorithmus [18]. Der Porter-Stemmer ist speziell für die englische Sprache entwickelt worden und basiert auf einer Reihe von Verkürzungsregeln, die auf ein zu stemmendes Wort solange angewendet werden, bis dieses eine minimale Anzahl von Silben aufweist. Zudem existieren dazu bereits einige Implementierungen in verschiedenen Programmiersprachen, die im Internet frei verfügbar sind.[4]

4.4 N-Gramme

Bisher wurden bei der Suche lediglich einzelne Schlüsselwörter betrachtet. Diese Schlüsselwörter könnten jedoch in einem Text auftreten, der keinen sicherheitsrelevanten Inhalt aufweist. Um die Präzision der Suchergebnisse zu verbessern werden anstatt von einzelnen Schlüsselwörtern N-Gramme betrachtet.

Ein N-Gramm ist eine Folge von N Wörtern, die sehr häufig in der gleichen Reihenfolge auftreten. Ein Beispiel für ein 2-Gramm (Bigramm) ist „customer buys“ und für ein 3-Gramm (Trigramm) „user logs on“. Ein einziges Schlüsselwort wird als Unigramm bezeichnet.

Das Ziel ist, N-Gramme zu finden, die häufig im Sicherheitskontext vorkommen und typisch für einen UMLsec-Stereotyp sind. Nach der Analyse von o.g. Anforderungsspezifikationen wurden die in Tabelle 4.3 geführten N-Gramme festgelegt.

UMLsec-Stereotyp	N-Gramme
secrecy	authorise password | authorise user | identify password | identify user | logon user | logon password | register user | different data | different view | user access | user add | user display | user download | user publish | add data | add account | display data | download data | publish data
integrity	allow user | block user | user access | user add | user delete | user edit | user manage | user save | user upload | add data | delete data | edit data | manage data | save data | upload data
high	account number | bank account | credit card | decrypt data | encrypt data | personal data | meta data | decrypt password | encrypt password
fair exchange	user buy internet | user lend internet | user reserve internet | business deliver | business sell internet | buy good internet | deliver good | lend good internt | sell good internet

Tabelle 4.3: N-Gramme und dazu gehörige UMLsec-Stereotypen

[4] http://tartarus.org/~martin/PorterStemmer

Um bessere Ergebnisse zu erzielen berücksichtigt diese Klassifizierungsheuristik nicht nur die festgelegten N-Gramme, sondern auch die dazu gehörigen Synonyme. In einem Text werden beispielsweise die Wörter „create content“ erkannt, da die Klassifizierungsheuristik das N-Gramm „add data“ berücksichtigt und „create“ bzw. „content“ Synonyme von „add“ bzw. „data“ sind.

Zusätzlich wird vor der Suche die Wortstammbildung angewendet um N-Gramme in abgeleiteter Form ebenso zu erkennen. Damit werden beispielsweise die Wörter „inserting components“ erkannt, da „insertig“ auf „insert“ und „components“ auf „component“ zurückgeführt werden und diese Synonyme von „add“ bzw. „data“ sind.

4.4.1 Entfernen von Stoppwörtern und Stoppzeichen

Mit Stoppwörtern werden die Wörter bezeichnet, die in einer Sprache sehr häufig vorkommen und für die semantische Bedeutung des Textes keine Relevanz besitzen. Darunter fallen Artikeln, Präpositionen, Konjunktionen, usw. Solche Wörter sind für die Informationsauffindung ohne Bedeutung. Beispiele aus der englischen Sprache sind u.a. „a“, „an“, „over“ und „the“. Eine umfangreiche Liste der englischen Stoppwörter ist im Internet veröffentlicht.[5] Als Stoppzeichen werden alle Zeichen wie Punkt, Komma, Semikolon, Klammer usw. bezeichnet.

Natürlichsprachig formulierte Texte enthalten viele Stoppwörter und Stoppzeichen, die in der Regel zwischen den Hauptwörtern stehen und dazu führen, dass die oben aufgeführten N-Gramme nicht erkannt werden. Aus diesem Grund werden im ersten Schritt die Stoppwörter und die Stoppzeichen aus dem zu durchsuchenden Text entfernt. Mit dieser Erweiterung wird beispielsweise in dem Satz „When a customer buys over the internet, he receives a confirmation per email.“ das N-Gramm „customer buy internet“ erkannt.

4.5 Bewertung der statischen Klassifizierungsheuristiken

Für die Bewertung der statischen Klassifizierungsheuristiken für das UMLsec-Stereotyp „secrecy“ wurden zufällig 200 Anforderungen aus den „Common Electronic Purse Specifications“ [16] ausgewählt.

Es wurden 85 Anforderungen als sicherheitskritisch (relevant) und 115 Anforderungen als nicht-sicherheitskritisch (irrelevant) klassifiziert. Die Ergebnisse sind in Tabelle 4.4 dargestellt. In der linken Spalte stehen folgende Klassifizierungsheuristiken:

1) Schlüsselwörter
2) Schlüsselwörter + Synonyme (WordNet)
3) Schlüsselwörter + Synonyme (manuell)
4) Schlüsselwörter + Synonyme (WordNet) + Stemming
5) Schlüsselwörter + Synonyme (manuell) + Stemming
6) N-Gramme + Synonyme (manuell) + Stemming

[5] ftp://ftp.cs.cornell.edu/pub/smart/english.stop

Secrecy |A|=200 |R|=85 |I|=115	Relevante Anforderungen		Nicht relevante Anforderungen	
	gefunden |RF|	nicht gef. |RN|	gefunden |IF|	nicht gef. |IN|
1)	16	69	13	102
2)	19	66	18	97
3)	26	59	34	81
4)	27	58	22	93
5)	43	42	59	56
6)	2	83	1	114

Tabelle 4.4: Ergebnisse der Klassifizierungsheuristiken für das UMLsec-Stereotyp „secrecy“

In Tabelle 4.5 werden Recall, Precision und das F-Maß dargestellt:

Secrecy	**Recall**	**Precision**	**F-Maß**
1)	0,19	0,55	0,28
2)	0,22	0,51	0,31
3)	0,31	0,43	0,36
4)	0,32	0,55	0,40
5)	**0,51**	**0,42**	**0,46**
6)	0,02	0,66	0,45

Tabelle 4.5: Recall, Precision und das F-Maß für “secrecy”

Diagramm 4.1 veranschaulicht graphisch das Verhältnis von Recall und Precision zueinander.

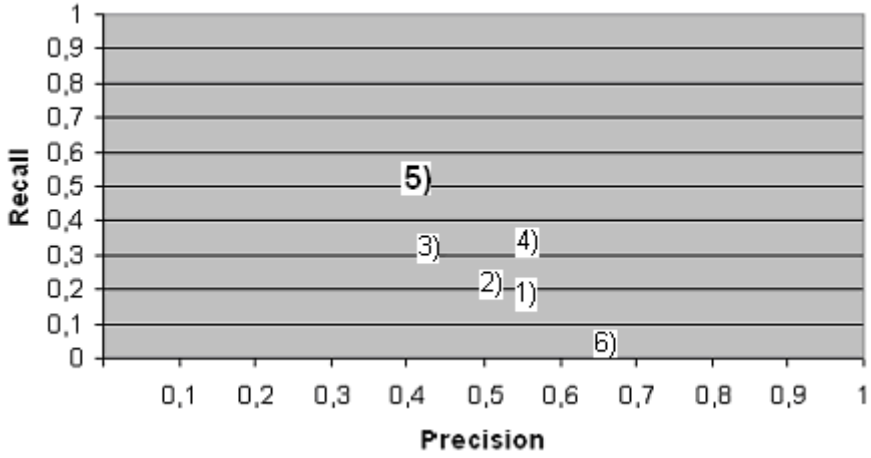


Diagramm 4.1: Recall-Precision-Diagramm für „secrecy“

Die Klassifizierungsheuristik für N-Gramme wurde nur im Zusammenhang mit manuell definierten Synonymen betrachten, da diese bereits bessere Ergebnisse als die Datenbank-Synonyme geliefert haben.

Die Suche mit N-Gramme liefert sehr wenige Ergebnisse zurück, sodass daraus keine aussagekräftigen Folgerungen abgeleitet werden konnten. Ein Grund dafür ist, dass die N-Gramm-Suche sehr präzise ist und die untersuchten Anforderungsspezifikationen für die Bestimmung der N-Gramme einen völlig unterschiedlichen Kontext aufweisen im Vergleich zum Kontext der elektronischen Geldbörse, deren Anforderungsspezifikation für die Tests verwendet wurde. Hätte man die gleiche Anforderungsspezifikation sowohl für die Bestimmung der N-Gramme als auch für die Tests verwendet, wäre das Ergebnis nicht brauchbar, da dieses keine reelle Situation darstellt.

Der Tabelle 4.5 ist zu entnehmen, dass die auf manuell definierten Synonymen basierende Klassifizierungsheuristik mit Wortstammbildung bezüglich Precision den niedrigsten Wert

aufweist. Zum einen ist dieser Wert jedoch nicht viel kleiner als der der anderen Heuristiken, zum anderen hält sich der Recall-Wert - obwohl dieser der höchste von allen ist - bei 51 Prozent in Grenzen. Folglich liefert diese Klassifizierungsheuristik bezüglich dem F-Maß das beste Ergebnis. Der Erfolg dieser Klassifizierungsheuristik ist auf umfangreiche Definition von Synonymen und gleichzeitiger Verwendung von Stemming zurückzuführen, sodass eine große Menge an Wörtern betrachtet wurde.

Ein Problem der statischen Klassifizierungsheuristiken ist zunächst Anforderungsspezifikationen zu sammeln und diese auf Sicherheitsanforderungen zu untersuchen. Der nächste Schritt ist, die gefundenen Sicherheitsanforderungen den in Tabelle 3.1 definierten UMLsec-Stereotypen zuzuordnen. Diese Zuordnung soll am Besten von einem Sicherheitsexperten vorgenommen werden, der aber in den meisten Fällen nicht zur Verfügung steht.

Ist dies geschehen, folgt der große manuelle Aufwand, die Schlüsselwörter und die Synonyme zu definieren.

Ein weiteres Problem ist, dass die Formulierung von Anforderungen - und von natürlichsprachigen Texten im allgemein – keine Grenzen gesetzt sind, sodass die festgelegten Schlüsselwörter und Synonyme mit sehr großer Wahrscheinlichkeit nicht ausreichen werden, um alle Fälle zu decken. Eine ständige manuelle Anpassung ist enorm aufwendig und kann deshalb nicht als akzeptable Lösung betrachtet werden.

5. Dynamische Klassifizierungsheuristiken

Während im letzten Kapitel statische Klassifizierungsheuristiken untersucht wurden, die manuell angepasst mussten, werden in diesem Kapitel dynamische Klassifizierungsheuristiken vorgesellt. Diese Klassifizierungsheuristiken können auf die Angaben des Benutzers reagieren und sich automatisch anpassen, um bessere Ergebnisse zu erzielen. Die hier verwendeten Klassifizierungsheuristiken stammen aus dem Bereich Information Retrieval und Data Mining und basieren auf der Häufigkeitsberechnung der Wörter.

5.1 Begriffsdatenbank

Als Basis für die Berechnungen der dynamischen Klassifizierungsheuristiken wird eine Datenbank konstruiert, die die Anzahl des Auftretens der Begriffe in den Anforderungen enthält.

5.1.1 Erstellung der Datenbank

Zunächst werden möglichst viele Anforderungen gesammelt und idealerweise von einem Sicherheitsexperten in zwei Mengen – Sicherheits- Nicht-Sicherheitsanforderungen – unterteilt. Aus diesen zwei Mengen werden nun die Worthäufigkeiten berechnet und in einer Datenbank festgehalten. Die Idee ist in Abbildung 5.1 veranschaulicht.

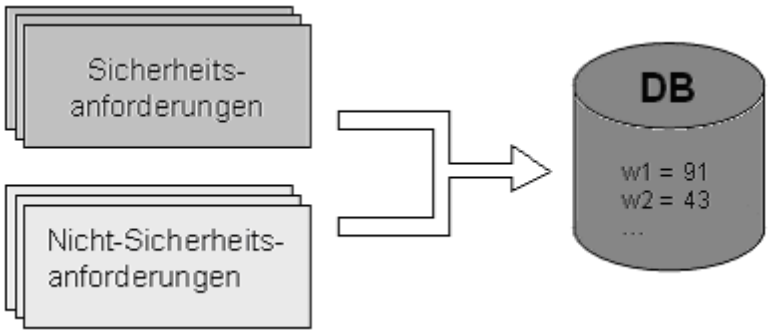


Abbildung 5.1: Datenbank für Worthäufigkeiten

Tabelle 5.1 zeigt einige Beispiele für Anforderungen, die bereits klassifiziert sind und als Basisdaten dienen.

ID	Anforderungsbeschreibung	Sicherheit
1	The service must allow provision for entities to be assigned a URI which uniquely identifies them and their different identities.	ja
2	The records used to test the prototype should represent issues that the service will need to be able to deal with, including those raised in the use case scenarios.	nein
3	...	….

Tabelle 5.1: Beispiele für Anforderungen

Die Anforderungen werden in einzelne Wörter zerlegt. Eine Erweiterung in dieser Arbeit ist, dass die Stoppwörter zunächst entfernt werden und in die Berechnung nicht einfließen weil sie keine semantische Bedeutung aufweisen (siehe Abschnitt 4.4.1).

Als Endergebnis steht in der Datenbank für jedes Wort die Anzahl des Auftretens in Sicherheitsanforderung und die Anzahl des Auftretens in Nicht-Sicherheitsanforderungen.

Tabelle 5.2 zeigt ein Beispiel für die Worthäufigkeitsdatenbank.

Anzahl Sicherheitsanforderungen: 85 Anzahl Nicht-Sicherheitsanforderungen : 115		
Wort	**Häufigkeit in Sicherheitsan-forderungen**	**Häufigkeit in Nicht-Sicherheitsanforderungen**
service	43	12
entities	91	11
including	3	19
…	…	…

Tabelle 5.2: Datenbank für Worthäufigkeiten

Diese Datenbank kann zu einem späteren Zeitpunkt mit bereits klassifizierten Anforderungen erweitert werden um zuverlässigere Ergebnisse zu erhalten.

5.1.2 Dynamischer Schlüsselwort-Relevanzfilter

Alternativ zum Entfernen von Stoppwörtern, wie in Abschnitt 4.4.1 beschrieben, kann ein dynamischer Schlüsselwort-Relevanzfilter verwendet werden um unerwünschte Wörter zu entfernen. Ein Schlüsselwort-Relevanzfilter [22] wird eingesetzt, um unerwünschte Wörter aus einer Anforderung auszuscheiden und so die Menge der betrachteten Wörter zu reduzieren und damit die Brauchbarkeit dieser als Unterscheidungskriterium für Anforderungen zu erhöhen.

Sobald es sich um eine sehr große Menge an Daten handelt, werden die Wörter, die entweder sehr häufig oder sehr selten vorkommen, in der Gesamtbetrachtung gleichermaßen wie die Stoppwörter keine Relevanz für die Klassifizierung darstellen.

Die erstelle Begriffdatenbank, wie in Abschnitt 5.1.1 beschrieben, kann für die Implementierung eines dynamischen Schlüsselwort-Relevanzfilters verwendet werden. Es wird für jedes Wort die Anzahl des Vorkommens in der Datenbank bestimmt und mittels zwei Grenzwerte entschieden, ob es sich um ein relevantes Wort handelt oder nicht. Es werden lediglich diejenigen Wörter als relevant betrachtet, deren Häufigkeit größer als ein Minimumswert und kleiner als ein Maximumswert ist. Alle Wörter mit Häufigkeiten außerhalb dieser beiden Grenzwerte werden ausgeschieden. Das folgende Bild stellt den Zusammenhang zwischen Worthäufigkeit und Wortrelevanz graphisch dar.

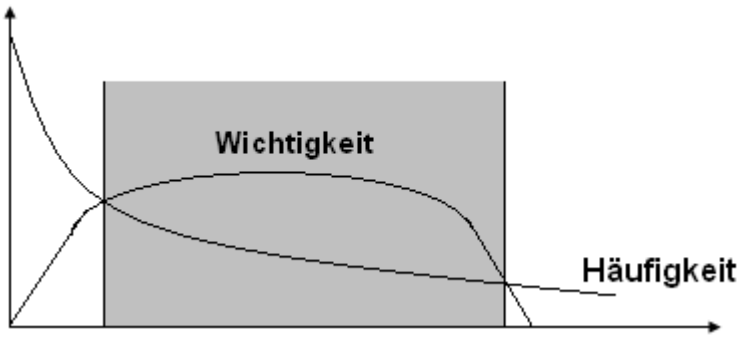


Abbildung 5.2: Wichtigkeit vs. Häufigkeit [22]

Die grau markierte Fläche in der obigen Abbildung kennzeichnet die Wörter, die vom Filter als relevant betrachtet werden. Bei sehr kleiner oder sehr großer Häufigkeit, ist die Wichtigkeit der Wörter sehr klein, d.h. sie sind ungeeignet für die Klassifizierung.

Die Datenbank unterliegt ständigen Veränderungen, deshalb ist es wichtig, dass die ausgeschiedenen Wörter nicht aus der Datenbank gelöscht werden, da sich deren Häufigkeit im Laufe der Zeit durch die Aktualisierung der Datenbank ändern könnte und somit zu einem zukünftigen Zeitpunkt vom Filter als relevant klassifiziert werden könnten.

Die Grenzwerte werden abhängig von der Informationsmenge in der Datenbank und vom erwünschten Ziel des Schlüsselwort-Relevanzfilters festgelegt. Ist das Ziel möglichst wenige Wörter als relevant zu betrachten, so werden die Grenzwerte weiter im Inneren des möglichen Wertebereiches, ansonsten weiter nahe an der oberen und unteren Grenze positioniert.

5.2 Bayesscher Filter

Der Bayessche Filter oder Bayes-Filter (benannt nach dem englischen Mathematiker Thomas Bayes, 1702 - 1761) ist ein statistischer Filter, der zuerst 1998 von Sahami et al. [24] zur Identifizierung von Spam-E-Mails vorgeschlagen wurde. Der Filter basiert auf den Bayesschen Wahrscheinlichkeitsbegriff und setzt auf Worthäufigkeiten von bereits als Spam oder als nicht Spam klassifizierten E-Mails. Laut einer Meldung der britischen BBC werden mit dem Bayes-Filter über 99,7 % der Spam-E-Mails erkannt. Die Bayessche Filtertechnologie basiert auf dem mathematischen Prinzip, dass die meisten Ereignisse voneinander abhängig sind und dass die Wahrscheinlichkeit eines zukünftigen Ereignisses aus vorherigen Ereigniseintritten abgeleitet werden kann.

Das gleiche Prinzip wird in dieser Arbeit zur Identifizierung von sicherheitskritischen Anforderungen verwendet. Es sei gemerkt, dass der Bayes-Filter für E-Mails weitere Attribute – wie z.B. Domäne des Absenders oder Länge der E-Mail usw. – für die Identifizierung von Spam-E-Mails verwendet. Da diese Attribute bei einer Anforderung nicht existieren wird lediglich die Worthäufigkeit für die Klassifizierung der Anforderung berücksichtigt.

5.2.1 Grundidee

Die Grundidee des Filters ist, dass bestimmte Begriffe oft nur in Sicherheitsanforderungen verwendet werden und diese in Nicht-Sicherheitsanforderungen selten vorkommen. Werden dieselben Begriffe in einer neuen Anforderung vorkommen, so wird diese auf Basis der bisher gesammelten Begriffshäufigkeiten mit großer Wahrscheinlichkeit als sicherheitskritisch klassifiziert. Der Bayes-Filter verwendet diese Daten um eine Klassifizierung von neuen Anforderungen vorzuschlagen.

Wird diese Klassifizierung von Sicherheitsexperten bestätigt oder abgelehnt, so kann in jedem Fall die neue Erkenntnis in die Datenbank einfließen, indem die Worthäufigkeiten in den neuen Anforderungen wie in Abschnitt 5.1.1 beschrieben berechnet werden. Somit ist der Bayes-Filter ein iterativer lernender Prozess, der im Laufe der Zeit mühelos angepasst und erweitert werden kann. Der Aufbau der Datenbank wird als Trainingsphase des Bayes-Filters bezeichnet.

5.2.2 Berechnung der Wahrscheinlichkeiten für einzelne Begriffe

Zunächst wird basierend auf die Anzahl des Auftretens für jedes Wort die Wahrscheinlichkeit berechnet, dass dieses ein Indiz für eine sicherheitskritische Anforderung ist.

Diese Berechnung basiert auf den Bayesschen Wahrscheinlichkeitsbegriff, der eine subjektive Wahrscheinlichkeit darstellt, die den Grad der Sicherheit angibt, dass ein Ereignis eintritt. Somit unterscheidet sich dieser von dem objektiven Wahrscheinlichkeitsbegriff, der die Wahrscheinlichkeit als relative Häufigkeit interpretiert.

Im Folgenden seien A und B beliebige Ereignisse und die Wahrscheinlichkeiten P(A) und P(B) größer Null. Dann ist die Bedingte Wahrscheinlichkeit P(A | B), die Wahrscheinlichkeit des Eintretens des Ereignisses A unter der Bedingung, dass das Ereignis B bereits eingetreten ist, wie folgt definiert:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Demnach ist die bedingte Wahrscheinlichkeiten für $P(B \mid A)$ wie folgt:

$$P(B \mid A) = \frac{P(A \cap B)}{P(A)}$$

Setzt man die beiden Terme für $P(A \cap B)$ gleich und teilt durch P(A), so erhält man

$$\boxed{P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B)}}$$

Diese Gleichung wird als Bayessche Regel (auch Bayessches Gesetzt, Bayessches Theorem oder Satz von Bayes) bezeichnet. Sie liegt allen modernen KI-Systemen für probabilistische Inferenz zugrunde [19].

Ist die Wahrscheinlichkeit P(B) unbekannt, so muss diese wie folgt auf bekannten Größen zurückgeführt werden:

$$\begin{aligned} P(B) &= P(B) \cdot 1 \\ &= P(B) \cdot (P(A^c \mid B) + P(A \mid B)) \\ &= P(B) \cdot P(A^c \mid B) + P(B) \cdot P(A \mid B) \\ &= P(B \mid A^c) \cdot P(A^c) + P(B \mid A) \cdot P(A) \end{aligned}$$

Nach dieser Umformung kann die Bayessche Regel wie folgt geschrieben werden:

$$P(A \mid B) = \frac{P(B \mid A) \cdot P(A)}{P(B \mid A^c) \cdot P(A^c) + P(B \mid A) \cdot P(A)}$$

Die Bayessche Regel erlaubt im gewissen Sinne das Umkehren von Schlussfolgerungen. Die Berechnung von P(Ereignis | Ursache) ist häufig einfach, jedoch ist oft P(Ursache | Ereignis) gesucht. Der Ansatz des Bayes-Filters ist, von charakteristischen Wörtern in einer Anforderung (Ereignis) auf die Eigenschaft, sicherheitskritisch zu sein (Ursache), zu schließen. Konkret heiß das, aus der vorherigen Erkenntnis

> *"...wenn es sich um eine Sicherheitsanforderung handelt, kommt mit x% Wahrscheinlichkeit das Wort y drin vor."*

auf das Ergebnis

> *"...falls das Wort y drin vorkommt, handelt es sich mit x% Wahrscheinlichkeit um eine Sicherheitsanforderung."*

zu schließen.

Seien TS (total security) die Anzahl der Sicherheitsanforderungen und TN (total non-security) die Anzahl der Nicht-Sicherheitsanforderungen in der Datenbank. Weiterhin sei für jedes Wort AS (appearance security) die Anazahl des Auftretens in Sicherheitsanforderungen und AN (appearance non-security) die Anazahl des Auftretens in Nicht-Sicherheitsanforderungen. Dann ist die Konkrete Formel für die Berechnung des Wahrscheinlichkeitswertes P_w für jedes Wort nach „Graham's Technique" [20] wie folgt:

$$P_w = \frac{AS/TS}{AN/TN + AS/TS}$$

Für das Wort „service“ aus Tabelle 5.2 gilt:

$$P_{service} = \frac{43/85}{12/115 + 43/85} = \frac{0{,}51}{0{,}10} = 0{,}83$$

Die Formel liefert einen Wert zwischen 0 und 1. Je größer der Wert, desto wahrscheinlicher ist es, dass es sich um eine sicherheitskritische Anforderung handelt.

Diese Berechnung wird für jeden Begriff in der zu klassifizierenden Anforderung durchgeführt. Einige Beispielswerte sind in Tabelle 5.3 dargestellt.

Wort	**P_w**
service	0,83
entities	0,91
including	0,18
...	…

Tabelle 5.3: Wahrscheinlichkeitswerte für einzelne Begriffe

5.2.3 Wahrscheinlichkeitsberechnung für die gesamte Anforderung

Nachdem die Wahrscheinlichkeiten für die einzelnen Wörter in der Anforderung vorliegen, wird im nächsten Schritt die Gesamtwahrscheinlich berechnet, ob es sich um eine sicherheitskritische Anforderung handelt. Falls eine Anforderung mehr als 15 Wörter enthält, werden die größten 15 Wahrscheinlichkeitswerte ausgewählt [20]. Diese 15 Wahrscheinlichkeitswerte werden miteinander kombiniert, um ein einziges Ergebnis zu erhalten. Seien P_1, P_2, …, P_n die ausgewählten Wahrscheinlichkeitswerte, dann sieht Grahams-Ansatz [20] für die Berechnung der Gesamtwahrscheinlichkeit P_{req} folgende Formel vor:

$$P_{req} = \frac{P_1 \cdot P_2 \cdot ... \cdot P_n}{P_1 \cdot P_2 \cdot ... \cdot P_n + (1-P_1) \cdot (1-P_2) \cdot ... \cdot (1-P_n)}$$

Für das Beispiel aus Tabelle 5.3, falls nur 3 anstatt 15 einzelne Wahrscheinlichkeitswerte betrachtet werden, ist die Gesamtwahrscheinlichkeit

$$P_{Anf.} = \frac{0{,}83 \cdot 0{,}91 \cdot 0{,}18}{0{,}83 \cdot 0{,}91 \cdot 0{,}18 + (1-0{,}83) \cdot (1-0{,}91) \cdot (1-0{,}18)} = 0{,}92$$

Diese Formel ergibt in der Regel sehr extreme Werte, d.h., Werte die sehr dicht bei 0 oder bei 1 liegen. Da die Werte entweder sehr klein oder sehr groß sind, lässt diese Formel nur einen kleinen Bereich für Ungewissheit zu. Ist das Endergebnis größer als einen bestimmten Grenzwert, so wird diese Anforderung als sicherheitskritisch eingestuft. Die Grahams-Implementierung [20] verwendet einen Grenzwert von 0,90.

Die Berechnung der Wahrscheinlichkeiten und die darauf basierende Klassifizierung der Anforderung werden als Klassifizierungsphase bezeichnet.

Diagramm 5.1 fasst die Aktivitäten des Bayes-Filters zusammen.

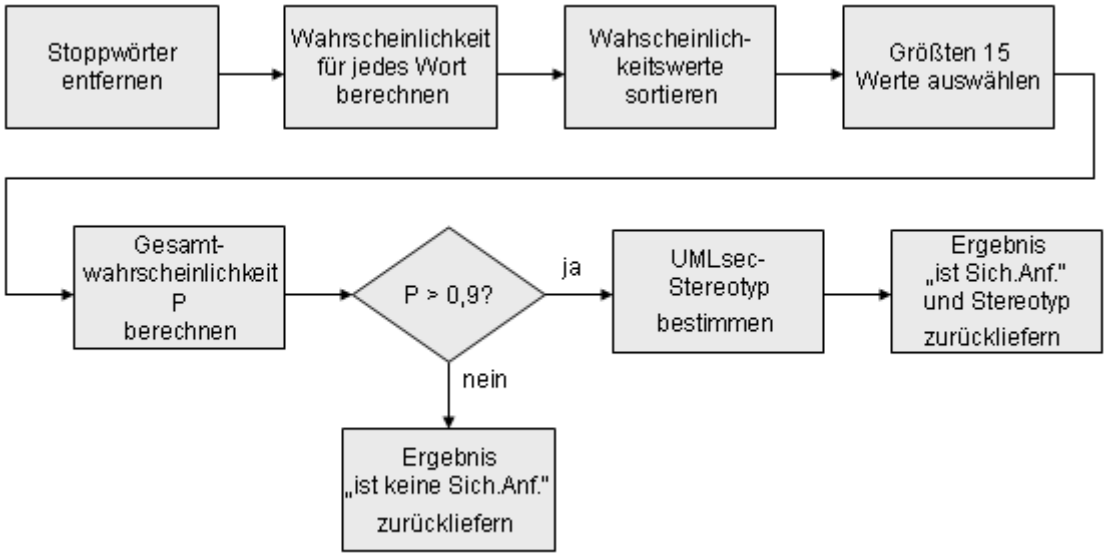


Diagramm 5.1: Aktivitätsdiagramm für Bayes-Filter

5.2.4 Erweiterung mit Wortstammbildung

Wie in Abschnitt 4.3 erläutert, können Wörter aus der gleichen Wortfamilie auf das Stammwort zurückgeführt werden. Mit dieser Ereweiterung werden die Häufigkeiten der Wörter mit der gleichen semantischen Bedeutung zusammengefasst, obwohl diese in verschieden Formen – wie z.B. Nomen, Adjektiv usw. – in den Anforderungen auftreten

Das Zurückführen auf das Stammwort muss sowohl bei der Erstellung der Begriffsdatenbank als auch bei der Klassifizierung einer neuen Anforderung durchgeführt werden.

5.3 TFxIDF

TFxIDF (Term Frequency, Inverse Document Frequency) [21] ist eine Methode aus dem Bereich Text Mining zur Bestimmung der Ähnlichkeit zwischen Dokumenten. Dieser Ansatz basiert auf die Worthäufigkeit in dem zu untersuchenden Dokument (Term Frequency) und die Worthäufigkeit in allen anderen Dokumenten (Document Frequency), mit denen die Ähnlichkeit bestimmt werden soll. TFxIDF wird in dieser Arbeit zur Klassifizierung der Anforderungen als sicherheitskritisch oder nicht-sicherheitskritisch angewendet.

5.3.1 Grundidee

Als erstes wird eine Begriffsdatenbank genau wie in Abschnitt 5.1.1 konstruiert. Die Gesamtheit aller Sicherheitsanforderungen sowie die der Nicht-Sicherheitsanforderungen werden als einzelne Dokumente betrachtet. Soll eine neue Anforderung klassifiziert werden, so wird die Ähnlichkeit zwischen dieser Anforderung und den beiden Dokumenten berechnet. Auf Basis dieser Berechnung wird die Anforderung als sicherheitskritisch oder nicht-sicherheitskritisch klassifiziert.

Bei der Bestimmung der Ähnlichkeit zwischen zwei Dokumenten mittels TFxIDF werden Wörtern in Überschriften besonders hohe Werte zugewiesen. Da es sich bei den zu klassifizierenden Anforderungen meistens um einzelne Sätze handelt und diese keine Überschriften enthalten, kann diese Ereweitung nicht verwendet werden.

5.3.2 Bestimmung der Ähnlichkeit

Bei der Klassifizierung einer neuen Anforderung wird diese zuerst in einzelnen Wörtern zerlegt und die Stoppwörter und Stoppzeichen entfernt.

Für jedes Wort in der zu klassifizierenden Anforderung werden folgende Werte bestimmt:

TF_w (Term Frequency): gibt die Häufigkeit des Auftretens eines Wortes in der zu klassifizierenden Anforderung an.

DF_{w_sec} (Document Frequency Security): gibt die Anzahl des Auftretens eines Wortes in den Sicherheitsanforderungen in der Datenbank an.

$DF_{w\_non\_sec}$ (Document Frequency Non-Security): gibt die Anzahl des Auftretens eines Wortes in den Nicht-Sicherheitsanforderungen in der Datenbank an.

N_{sec}: Die Gesamtzahl der Sicherheitsanforderungen in der Datenbank.
N_{non_ec}: Die Gesamtzahl der Nicht-Sicherheitsanforderungen in der Datenbank.

Um einen Wert zu erhalten, der darüber Auskunft gibt, wie sehr sich ein Wort dazu eignet,

Anforderungen zu unterscheiden, wird die Inverse Document Frequency (IDFw) definiert:

$$IDF_{w\_sec} = \log(\frac{N_{sec}}{DF_{w\_sec}})$$

$$IDF_{w\_non\_sec} = \log(\frac{N_{non\_sec}}{DF_{w\_non\_sec}})$$

Die beiden Werte TF_w und IDF_w werden zu einem einzigen Maß kombiniert, das sowohl Wörter, die sehr häufig in den Anforderungen auftreten, als auch solche, die sich sehr gut zur Unterscheidung von Anforderungen eignen, berücksichtigt. Dieser Wert wird als Gewicht (Weight) W_w eines Wortes bezeichnet und ist wie folgt gegeben:

$$W_{w\_sec} = TF_w \cdot IDF_{w\_sec}$$

$$W_{w\_non\_sec} = TF_w \cdot IDF_{w\_non\_sec}$$

Um ein Maß für die Ähnlichkeit (Similarity) S_{req} einer zu klassifizierenden Anforderung bezüglich der Anforderungen in der Datenbank zu bestimmen, werden die Gewichte für jedes Wort einerseits bezüglich der Sicherheitsanforderungen (S_{req_sec}), andererseits bezüglich der Nicht-Sicherheitsanforderungen ($S_{req\_non\text{-}sec}$) berechnet und zusammenaddiert:

$$S_{req\_sec} = \sum_w (W_{w\_sec})$$

$$S_{req\_non\text{-}sec} = \sum_w (W_{w\_non\_sec})$$

Anhand dieser beiden Ähnlichkeitswerte wird die Anforderung im letzten Schritt als sicherheitskritisch bzw. nicht-sicherheitskritisch klassifiziert.

Diagramm 5.2 fasst die Aktivitäten des TFxIDF-Verfahrens zusammen.

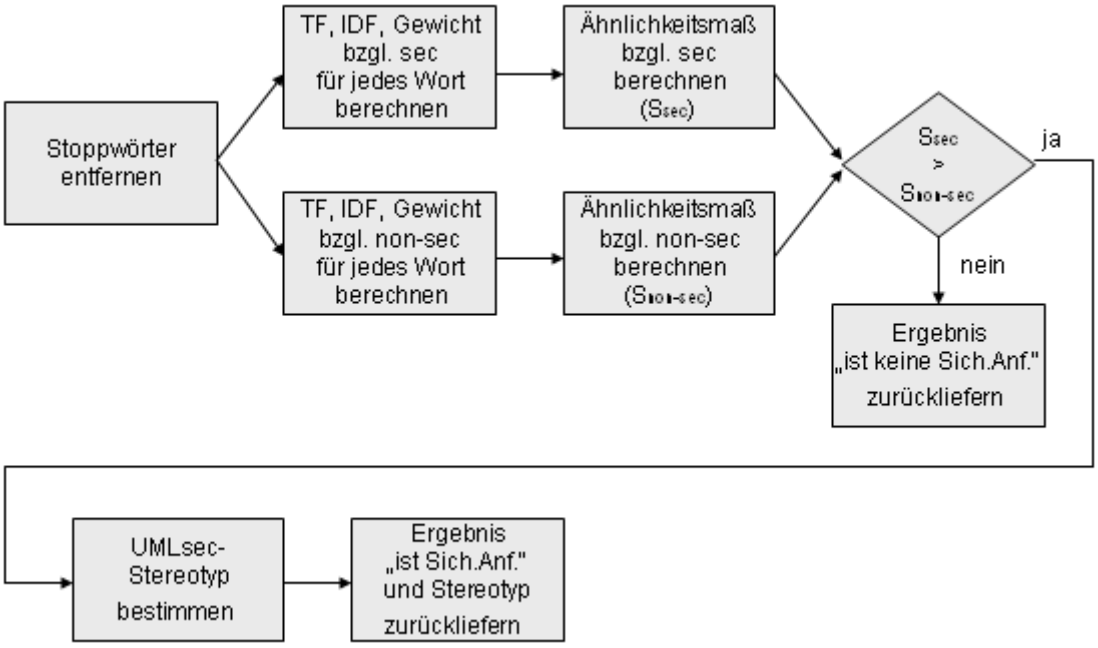


Diagramm 5.4: Aktivitätsdiagramm für das TFxIDF-Verfahren

Um die Berechnungen besser nachzuvollziehen, betrachten wir die Beispieldatenbank aus Tabelle 5.2 und eine Beispielanforderung, die die Wörter „service“, „entities“ und „including“ enthält, wobei das Wort „service“ zwei Mal in der Anforderung vorkommt. Somit können folgende Informationen ohne Berechnung aus der Datenbank abgelesen werden. Im Kopf der Tabelle 5.2 steht die Anzahl der Sicherheits- und Nichtsicherheitsanforderungen. Daraus folgt:

$\mathbf{N_{sec}} = 85$

$\mathbf{N_{non\_ec}} = 115$

Tabelle 5.4 zeigt $\mathbf{TF_w}$, $\mathbf{DF_{w\_sec}}$ und $\mathbf{DF_{w\_non\_sec}}$ für die einzelnen Wörter in der Anforderung

Wort	$\mathbf{TF_w}$	$\mathbf{DF_{w\_sec}}$	$\mathbf{DF_{w\_non\_sec}}$
service	2	43	12
entities	1	91	11
including	1	3	19
...			

Tabelle 5.4: $\mathbf{TF_w}$, $\mathbf{DF_{w\_sec}}$ und $\mathbf{DF_{w\_non\_sec}}$ für die einzelnen Wörter

Das Wort „service“ kommt in der zu klassifizierenden Anforderung zwei Mal vor, deshalb ist $\mathbf{TF_w}$ für „service“ gleich 2. In den bereits klassifizierten Anforderungen in der Datenbank kommt es 43 Mal in Sicherheitsanforderungen und 12 Mal in Nicht-Sicherheitsanforderungen vor, deshalb ist $\mathbf{DF_{w\_sec}}$ gleich 43 und $\mathbf{DF_{w\_non\_sec}}$ gleich 12.

Folglich werden die Werte $\mathbf{IDF_{w\_sec}}$, $\mathbf{IDF_{w\_non\_sec}}$, $\mathbf{W_{w\_sec}}$ und $\mathbf{W_{w\_non\_sec}}$ mit den oben geführten Formeln berechnet. Tabelle 5.5 zeigt die Ergebnisse.

Wort	$\mathbf{IDF_{w\_sec}}$	$\mathbf{IDF_{w\_non\_sec}}$	$\mathbf{W_{w\_sec}}$	$\mathbf{W_{w\_non\_sec}}$
service	log(85/43)=0,3	log(115/12)=0,98	2*0,3=0,6	2*0,98=1,96
entities	log(85/91)=-0,03	log(115/11)=1,04	1*-0,03=-0,03	1*1,04=1,04
including	log(85/3)=1,45	log(115/19)=0,78	1*1,45=1,45	1*0,78=0,78
...				

Tabelle 5.5: Werte für $\mathbf{IDF_{w\_sec}}$, $\mathbf{IDF_{w\_non\_sec}}$, $\mathbf{W_{w\_sec}}$ und $\mathbf{W_{w\_non\_sec}}$

Im nächsten Schritt werden die Ähnlichkeitswerte berechnet indem die Gewichte $\mathbf{W_{w\_sec}}$ und $\mathbf{W_{w\_non\_sec}}$ aufsummiert werden:

$\mathbf{S_{req\_sec}} = 0{,}6 + -0{,}03 + 1{,}45 = 2{,}02$

$\mathbf{S_{req\_nonsec}} = 1{,}96 + 1{,}04 + 0{,}78 = 3{,}78$

Da der Ähnlichkeitswert für Nicht-Sicherheitsanforderungen (3,78) größer ist als der Ähnlichkeitswert für Sicherheitsanforderung (2,02) wird die Anforderung als nicht-sicherheitskritisch klassifiziert.

5.4 Vorteile der dynamischen Klassifizierungsverfahren

Die dynamischen Klassifizierungsverfahren bringen folgende Vorteile mit sich:

Berücksichtigung der gesamten Anforderungen und nicht nur einzelne Schlüsselwörter.

Automatische und kontinuierliche Aktualisierung der Begriffsdatenbank, sodass im Laufe der Zeit zuverlässigere Ergebnisse erzielt werden.

Klassifizierung von Anforderungen in sämtlichen Sprachen möglich, ohne vorherige manuelle Definition von Schlüsselwörtern.

5.5 Laufzeitkomplexität

Im Folgenden wird die Laufzeitkomplexität der dynamischen Klassifizierungsheuristiken analysiert. Die bei der Klassifizierung einer Anforderung mit n Wörtern durchlaufenen Aktivitäten und die dafür benötigten Zeiten für den Bayes-Filter werden in Tabelle 5.6 dargestellt.

Nr.	Aktivität	Laufzeit
1	Entfernen von Stoppwörtern: Bei jedem Wort wird überprüft, ob es sich um ein Stoppwort handelt.	$O(n)$
2	Wahrscheinlichkeitswert P_w: Für jedes Wort wird P_w in konstanter Zeit berechnet.	$O(n)$
3	Sortierung der Wahrscheinlichkeitswerte (z.B. mit Merge Sort).	$O(n \cdot \log(n))$
4	Selektion der größten 15 Werte: Unabhängig von der Anforderungslänge.	$O(1)$
5	Berechnung der Gesamtwahrscheinlichkeit P_{req}: Unabhängig von der Anforderungslänge.	$O(1)$
6	Vergleich mit dem Grenzwert.	$O(1)$
7	Bestimmung des UMLsec-Stereotypes: Unabhängig von der Anforderungslänge bei konstanter Anzahl Stereotypen.	$O(1)$

Tabelle 5.6: Laufzeitkomplexität des Bayes-Filters

Praktisch können Schritt 1, 2 und 3 in einem Durchlauf der Anforderung abgearbeitet werden. Somit ergibt sich eine Gasamtlaufzeit für den Bayes-Filter von $O(n \cdot \log(n))$.

Die Aktivitäten und die dazugehörigen Lautzeiten des TFxIDF-Verfahrens zeigt Tabelle 5.7.

Nr.	Aktivität	Laufzeit
1	Entfernen von Stoppwörtern: Bei jedem Wort wird überprüft, ob es sich um ein Stoppwort handelt.	O(n)
2	Berechnung von TF, IDF und Gewicht W: Für jedes Wort wird TF, IDF und W in konstanter Zeit berechnet.	O(n)
3	Ähnlichkeitsmaß berechnen	O(n)
4	Ähnlichkeitsmaße vergleichen	O(1)
5	Bestimmung des UMLsec-Stereotypes: Unabhängig von der Anforderungslänge bei konstanter Anzahl von UMLsec-Stereotypen.	O(1)

Tabelle 5.7: Laufzeitkomplexität des TFxIDF-Verfahrens

Praktisch können Schritt 1, 2 und 3 in einem Durchlauf der Anforderung abgearbeitet werden. Somit ergibt sich eine Gasamtlaufzeit für das TFxIDF-Verfahren von $O(n)$.

Demzufolge ist TFxIDF das schnellere Verfahren, da es im Gegensatz zum Bayes-Filter keine Sortierung benötigt.

Um praktische Laufzeiten zu messen, wurden 100 Anforderungen mit einer Durchschnittslänge von 16 Wörtern 1000 Mal auf einem handelsüblichen Computer (1,6 GH, 2GB RAM) analysiert. Tabelle 5.6 gibt den minimalen, maximalen und durchschnittlichen Wert für die Analyse von 100 Anforderungen mit Hilfe des Bayes-Filter bzw. des TFxIDF-Verfahrens in Sekunden an.

	Bayes-Filter	TFxIDF
Minimum	0,034	0,033
Maximum	0,776	0,701
Mittelwert	0,388	0,363

Tabelle 5.8: Praktische Laufzeiten für 100 Anforderungen in Sekunden

Beide Verfahren benötigen für die Analyse von 100 Anforderungen im schlechtesten Fall deutlich weniger als eine Sekunde, was eine sehr akzeptable Zeit für eine „on the fly“ Anforderungsklassifizierung ist.

5.6 Erkennung von UMLsec-Stereotypen

Bei der Erstellung der Begriffsdatenbank wie im Abschnitt 5.1.1 erläutert, wird bei den Sicherheitsanforderungen zusätzlich für jedes Wort die Anzahl des Auftretens bei den UMLsec-Stereotypen in der Datenbank festgehalten. Tabelle 5.7 zeigt ein Beispiel für einige Worthäufigkeiten bei UMLsec-Stereotypen.

Wort/Stereotyp	critical	secrecy	data security	…
authentication	17	3	5	
processing	8	21	2	…
…	…	…	…	…

Tabelle 5.8: Beispiel für Worthäufigkeiten bei UMLsec-Stereotypen

Wird eine Anforderung als sicherheitskritisch klassifiziert, so kann ihr im nächsten Schritt zusätzlich ein UMLsec-Stereotyp zugeordnet werden. Für die Festlegung des UMLsec-Stereotyps wird das signifikanteste Wort in der Anforderung bestimmt.

Bei der Verwendung des Bayes-Filters ist das signifikanteste Wort das mit dem höchsten Wahrscheinlichkeitswert P_w (siehe Abschnitt 5.2.2), bei der Anwendung von TFxIDF ist es das mit dem größten Gewichtes W_w (siehe Abschnitt 5.3.2).

Ist das signifikanteste Wort bestimmt, so wird der zugehörige UMLsec-Stereotyp, bei dem dieses Wort am häufigsten vorkommt, als UMLsec-Stereotyp für die als sicherheitskritisch identifizierte Anforderung ausgewählt. Wird beispielsweise das Wort „processing“ als signifikant bestimmt, so wird laut Tabelle 5.7 der UMLsec-Stereotyp „secrecy“ ausgewählt.

Der Vorteil dieser Methode ist, dass die Worthäufigkeiten bei der Erweiterung der Datenbank mit weiteren Sicherheitsanforderungen und zugehörigen UMLsec-Stereotypen automatisch angepasst werden.

5.7 Bewertung der dynamischen Klassifizierungsheuristiken

Für die Bewertung der dynamischen Klassifizierungsheuristiken wurden zufällig 200 Anforderungen aus den „Common Electronic Purse Specifications“ [16] ausgewählt.

Es wurden 85 Sicherheitsanforderungen und 115 Nicht-Sicherheitsanforderungen identifiziert.

Die Begriffsdatenbank wurde mit 50 Sicherheitsanforderungen und 50 Nicht-Sicherheitsanforderungen aufgebaut. Somit blieben 35 Sicherheitsanforderungen und 65 Nicht-Sicherheitsanforderungen für die Tests übrig. Die Ergebnisse sind in Tabelle 5.9 dargestellt. In der linken Spalte stehen folgende Klassifizierungsheuristiken:

1) Bayes-Filter (TFxIDF)
2) Bayes-Filter mit Wortstammbildung (TFxIDF mit Wortstammbildung)

	Relevante Anforderungen		Nicht relevante Anforderungen	
	gefunden |RF|	nicht gef. |RN|	gefunden |IF|	nicht gef. |IN|
1)	17 (14)	18 (20)	26 (14)	39 (51)
2)	11 (19)	24 (16)	24 (30)	41 (35)

Tabelle 5.9: Ergebnisse der dynamischen Klassifizierungsheuristiken

In Tabelle 5.10 werden Recall, Precision und das F-Maß dargestellt:

	Recall	**Precision**	**F-Maß**
1)	0,49 (0,43)	0,40 (0,52)	0,44 (0,47)
2)	0,31 (0,54)	0,31 (0,39)	0,31 (0,45)

Tabelle 5.10: Recall, Precision und F-Maß für dynamische Klassifizierungsheuristiken

Demnach liefern der Bayes-Filter mit einem F-Maß von 0,44 und TFxIDF mit 0,47 ähnliche Ergebnisse, wobei der Bayes-Filter einen höheren Recall, TFxIDF eine höhere Precision auszeichnet.

Auffällig ist, insbesondere beim Bayes-Filter, dass sich bei der Wortstammbildung die Ergebnisse verschlechtern. Das könnte den Grund haben, dass Wörter in einer anderen Form teilweise in einer anderen semantischen Bedeutung auftreten und der Bayes-Filter bessere Ergebnisse liefert, wenn diese separat betrachtet.

Tabelle 5.11 vergleicht die Ergebnisse der dynamischen Klassifizierungsverfahren mit den Ergebnissen des besten statischen Klassifizierungsverfahrens.

Verfahren	Recall	Precision	F-Maß
Statisch: Schlüsselwörter + Synonyme (manuell) + Stemming	0,51	0,42	0,46
Dynamisch: Bayes-Filter	0,49	0,40	0,44
Dynamisch: TFxIDF	0,43	0,52	0,47

Tabelle 5.11: Vergleich zwischen statischen und dynamischen Klassifizierungsverfahren

Demzufolge liefern alle Verfahren sehr ähnliche Ergebnisse. Der Nachteil der statischen Verfahren ist, dass eine Liste der relevanten Schlüsselwörter manuell erstellt werden muss, was mit hohem Aufwand verbunden ist. Dieser Punkt entfällt bei den dynamischen Verfahren.

Im Folgenden wird untersucht, wie sich die Ergebnisse der dynamischen Verfahren bei verschiedenen Datenkonstellationen in der Datenbank verändern und ob Tests mit mehr Daten bessere Ergebnisse liefern.

Da der Bayes-Filter und TFxIDF auf Berechnungen der Worthäufigkeiten basieren und im Gegensatz zu den statischen Klassifizierungsheuristiken sprachunabhängig sind, wurden für die weiteren Tests Anforderungsspezifikationen in der deutschen Sprach von 30 studentischen Softwareprojekten am Fachgebiet Software Engineering der Leibniz Universität Hannover analysiert und die Anforderungen extrahiert. Das Ergebnis war eine Sammlung von 272 Sicherheitsanforderungen und 636 Nicht-Sicherheitsanforderungen.

Die Tests werden mit 100, jeweils 50 Sicherheitsanforderungen und 50 Nicht-Sicherheitsanforderungen durchgeführt. Die restlichen zur Verfügung stehenden Anforderungen werden für unterschiedliche Konstellationen für den Datenbankaufbau verwendet.

Tabellen 5.12 und 5.13 zeigen die Ergebnisse für den Fall, dass die Datenbank mit gleicher Anzahl von Sicherheits- sowie Nicht-Sicherheitsanforderungen aufgebaut ist.

Bayes-Filter (TFxIDF)						
	Datenbankstruktur		Relevante Anforderungen		Nicht relevante Anforderungen	
	Anzahl Sicher.Anf.	Anzahl Nicht-S.Anf.	gefunden |RF|	nicht gef. |RN|	gefunden |IF|	nicht gef. |IN|
1)	50	50	18 (20)	32 (30)	8 (10)	42 (40)
2)	100	100	23 (23)	27 (27)	5 (6)	45 (44)
3)	150	150	20 (21)	30 (29)	3 (4)	47 (46)
4)	222	222	23 (23)	27 (27)	6 (7)	44 (43)

Tabelle 5.12: Ergebnisse bei gleicher Anazahl von Sicherheits- und Nicht-Sicherheitsanf.

	Recall	Precision	F-Maß
1)	0,37 (0,41)	0,69 (0,67)	0,48 (0,51)
2)	0,47 (0,47)	0,82 (0,79)	**0,60 (0,59)**
3)	0,41 (0,43)	0,87 (0,84)	**0,56 (0,57)**
4)	0,47 (0,47)	0,79 (0,76)	**0,59 (0,58)**

Tabelle 5.13: Recall, Precision und F-Maß bei gleicher An. von Sicherheits- und Nicht-Sicherheitsanf.

Den Ergebnissen zufolge liefert ein weiterer Aufbau der Datenbank keine besseren Ergebnisse wie zunächst vermutet. Der Grund dafür ist, dass eventuell gleiche Begriffe sowohl in Sicherheits- als auch in Nicht-Sicherheitsanforderungen auftreten und somit zur Klassifizierung keinen Beitrag leisten.

Eine weitere mögliche Konstellation ist, dass die Datenbank mehr Nicht-Sicherheitsanforderungen enthält als Sicherheitsanforderungen. Tabellen 5.14 und 5.15 stellen die Ergebnisse für diesen Fall dar.

Bayes-Filter (TFxIDF)						
	Datenbankstruktur		Relevante Anforderungen		Nicht relevante Anforderungen	
	Anzahl Sicher.Anf.	Anzahl Nicht-S.Anf.	gefunden |RF|	nicht gef. |RN|	gefunden |IF|	nicht gef. |IN|
5)	222	333	16 (17)	34 (33)	3 (6)	47 (44)
6)	222	444	15 (17)	35 (33)	2 (4)	48 (46)
7)	222	586	12 (13)	38 (37)	3 (4)	47 (46)

Tabelle 5.14: Ergebnisse bei mehr Nicht-Sicherheitsanforderungen

	Recall	**Precision**	**F-Maß**
5)	0,33 (0,35)	0,84 (0,74)	**0,47 (0,47)**
6)	0,31 (0,35)	0,88 (0,81)	**0,45 (0,49)**
7)	0,24 (0,27)	0,80 (0,76)	**0,38 (0,39)**

Tabelle 5.15: Recall, Precision und F-Maß bei mehr Nicht-Sicherheitsanforderungen

Aus diesen Tests wird deutlich, dass sich die Ergebnisse bei steigender Anzahl von Nicht-Sicherheitsanforderungen und konstant bleibender Anzahl von Sicherheitsanforderungen in der Datenbank deutlich verschlechtern, da die Worthäufigkeiten bei Nicht-Sicherheitsanforderungen mehr ins Gewicht fallen.

Des Weiteren besteht die Möglichkeit, die Datenbank mit mehr Sicherheitsanforderungen als Nicht-Sicherheitsanforderungen aufzubauen. Die Ergebnisse für diese Konstellation zeigen Tabellen 5.16 und 5.17.

Bayes-Filter (TFxIDF)						
	Datenbankstruktur		Relevante Anforderungen		Nicht relevante Anforderungen	
	Anzahl Sicher.Anf.	Anzahl Nicht-S.Anf.	gefunden |RF|	nicht gef. |RN|	gefunden |IF|	nicht gef. |IN|
8)	100	50	31 (31)	19 (19)	10 (14)	40 (36)
9)	150	50	34 (34)	16 (16)	13 (15)	37 (35)
10)	222	50	35 (35)	15 (15)	20 (21)	30 (29)

Tabelle 5.16: Ergebnisse bei mehr Sicherheitsanforderungen

	Recall	**Precision**	**F-Maß**
8)	0,63 (0,63)	0,76 (0,69)	**0,69 (0,66)**
9)	0,69 (0,69)	0,72 (0,69)	**0,71 (0,69)**
10)	0,71 (0,71)	0,64 (0,63)	**0,67 (0,67)**

Tabelle 5.17: Recall, Precision und F-Maß bei mehr Sicherheitsanforderungen

Es ist ersichtlich, dass diese Kombination die besten Ergebnisse zurückliefert, jedoch bei steigender Anzahl von Sicherheitsanforderungen keine großen Veränderungen zu verzeichnen sind. Eine Mögliche Erklärung ist, dass die zusätzlichen Sicherheitsanforderungen die gleichen Begriffe aufweisen wie die bereits in der Datenbank existierenden, die dazu beitragen, die Worthäufigkeiten zu steigern, jedoch nicht für die Auffindung neuer Sicherheitsanforderungen.

Da sich die Ergebnisse des Bayes-Filters und die des TFxIDF sehr ähneln, werden in Abbildung 5.3 stellvertretend die F-Maße für den Bayes-Filter bei den verschiedenen Konstellationen dargestellt.

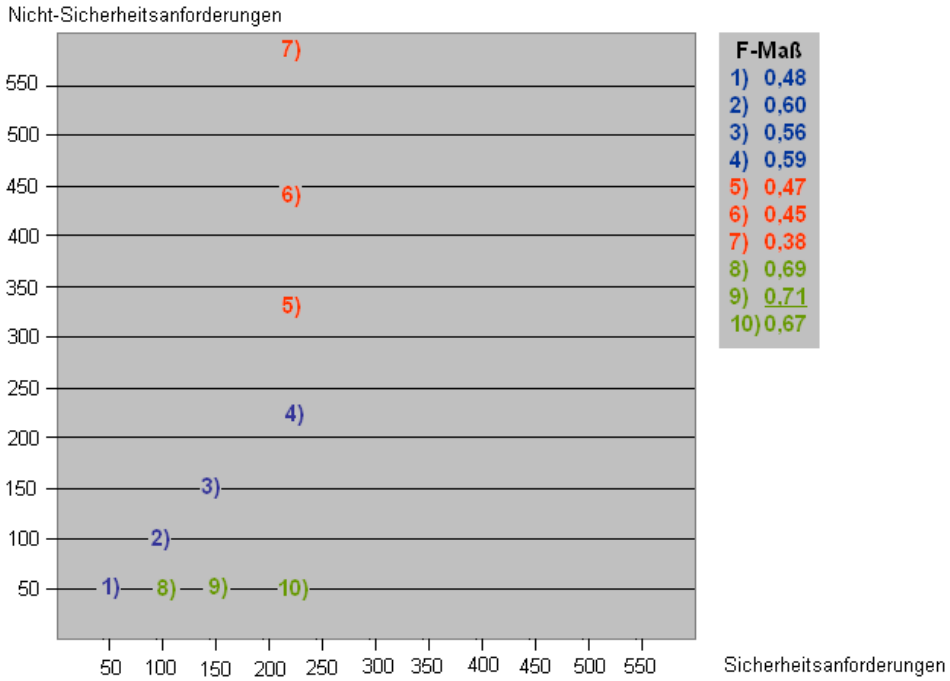


Abbildung 5.3: F-Maß für verschiedene Testkonstellationen

Als Gesamtergebnis kann man schlussfolgern, dass der Aufbau einer Datenbank mit mehr Sicherheitsanforderungen die beste Strategie ist, um neue Sicherheitsanforderungen aufzudecken. D.h., im Laufe der Zeit muss darauf geachtet werden, dass die Datenbank grundsätzlich mit Sicherheitsanforderungen erweitert wird.

5.8 Diskussion der Aussagekraft der Bewertung

In diesem Abschnitt werden die Testergebnisse der statischen und dynamischen Klassifizierungsheuristiken kritisch hinsichtlich ihrer Aussagekraft untersucht.

Die manuelle Klassifizierung in Sicherheits- und Nicht-Sicherheitsanforderungen, die als Referenz beim Testen der Heuristiken dient, sollte von einem Sicherheitsexperten durchgeführten werden, um die Aussagekraft der Ergebnisse zu verbessern. Während dieser Arbeit stand kein Sicherheitsexperte zur Verfügung, sodass die Anforderungsklassifizierung im Rahmen dieser Arbeit nach bestem Wissen und Gewissen durchgeführt wurde. Das könnte zur Folge haben, dass einige sicherheitskritische Anforderungen nicht als solche betrachtet wurden und umgekehrt, was zur Abweichung der Ergebnisse führt.

Weiterhin wird die Aussagekraft der Ergebnisse sowohl durch die Anzahl der für die Tests verwendeten Anforderungen als auch durch die Anzahl der analysierten Anforderungen für die Bestimmung der sicherheitsrelevanten Schlüsselwörter im Falle der statischen Klassifizierungsheuristiken bzw. für den Aufbau der Begriffsdatenbank im Falle der dynamischen Klassifizierungsheuristiken beeinflusst. Je mehr Anforderungen verwendeten werden, desto mehr Wissen über die Wörter und deren Häufigkeiten wird gesammelt, was zu aussagekräftigeren Ergebnissen führt.

6. Verfeinerung von Anforderungen

Ein Ziel dieser Arbeit ist, die Softwareanalysten bei der Verfeinerung von sicherheitskritischen Anforderungen zu unterstützen.

Softwareanforderungen werden in der Regel in verschiedensten Abstraktionsebenen erfasst. In der Softwareentwicklung stammen die Anforderungen meistens von verschiedenen Quellen, z.B. intern von Entwicklern oder vom Management bzw. extern von Kunden oder Vertragspartnern [29]. Der wesentliche Vorteil der Anforderungsverfeinerung ist, dass alle Anforderungen detailliert genug beschrieben werden, sodass mit der Entwicklung begonnen werden kann.

Vor der eigentlichen Verfeinerung muss jedoch zunächst festgelegt werden, welche Abstraktionsebenen es gibt und auf welcher Abstraktionsebene jede Anforderung liegt. Nachdem einer Anforderung eine Abstraktionsebene zugewiesen wird, kann von dort aus die Verfeinerung beginnen, falls diese Abstraktionsebene zu hoch ist, d.h., falls die Anforderung sehr abstrakt ist.

In [29] wird ein Modell mit vier Abstraktionsebenen vorgestellt, das es erlaubt, jede Anforderung einer Abstraktionsebene zuzuordnen. Das Modell heißt RAM (Requirements Abstraction Modell).

6.1 RAM (Requirements Abstraction Modell)

RAM unterstützt die Abstraktion und die Verfeinerung von Anforderungen, sodass Anforderungen gleicher Abstraktionsebene miteinander vergleichbar sind. Die Vergleichbarkeit von Anforderungen ermöglicht eine effektive Priorisierung und Planung. RAM nimmt eine Menge von Anforderungen verschiedenen Typs (verschiedene Abstraktionsebenen) als Eingabe und ermöglicht die Strukturierung dieser Anforderungen in verschiedenen Abstraktionsebenen und die Verfeinerung zu detaillierten Anforderungen.

Im Folgenden werden zwei Anforderungen vorgestellt, die auf verschiedenen Abstraktionsebenen liegen und unterschiedlich detailliert beschrieben sind.

Anforderung 1:

Titel: Unterstützung von Standardformaten.

Beschreibung: Das System sollte Standardformate unterstützen.

Anforderung 2:

ID: 113.

Titel: Ausgabe als XML-Format speichern.

Beschreibung: Ein Benutzer sollte in der Lage sein, Ausgabedaten als XML-Datei zu speichern um diese in anderen Systemen zu verwenden.

Voraussetzung: Anforderung 78 muss vor dieser Anforderung implementiert sein.

Quelle: Herr Klaus Mustermann.

Die Erläuterung von RAM wird anhand mehrerer Beispiele unterstützt. Diese Beispiele sind nicht domainen- oder firmenspezifisch und ermöglichen eine bessere Übersicht zum Verständnis des Modells. Die Anforderungsanalyse mit RAM umfasst drei Schritte:

1. **Specify:** Die rohen Anforderungen bestimmen und genügend Informationen herauslocken und die Anforderungen mit einer Anzahl von Attributen versehen.
2. **Place:** Jede Anforderung einer Abstraktionsebene zuordnen.
3. **Abstraction:** Jede Anforderung durchläuft einen Abstraktionsprozess.

Abbildung 6.1 stellt die Aktionen im RAM graphisch dar.

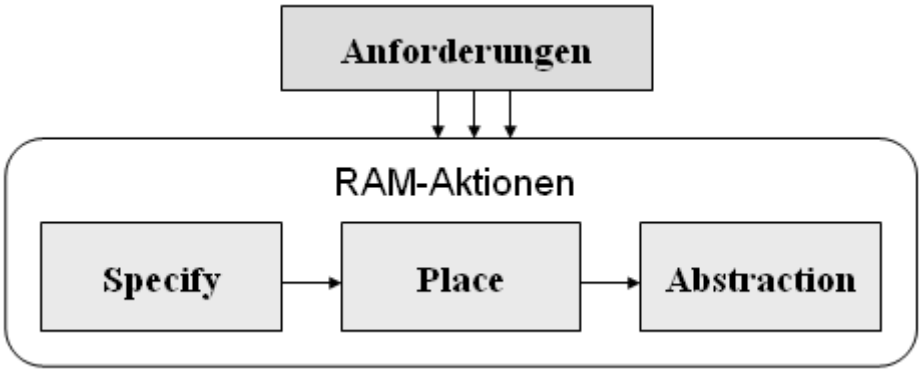


Abbildung 6.1: Aktionsschritte im RAM

Diese Schritte werden im Folgenden detaillierter beschrieben.

6.1.1 Specify:

Ziel dieses Schritts ist, eine einheitliche Struktur über die spezifizierten rohen Anforderungen zu erhalten um diese besser zu verstehen. Zu diesem Zweck werden jeder Anforderung vier Attribute zugeordnet:

1. Titel: Der Titel muss den Inhalt der Anforderung reflektieren und soll nicht länger als 5 Wörter sein. Im Folgenden ein Beispiel für den Titel einiger Anforderungen:

Anforderung A: Standardisierte Formate für die Kommunikation.
Anforderung B: Drucken von Systeminformationen.
Anforderung C: Unterstützung von Mehrsprachigkeit.

2. Beschreibung: Die Anforderungsbeschreibung soll maximal 5 Sätze beinhalten. Diese sollen die Hauptaussage der Anforderung möglichst deutlich beschreiben. Im Folgenden ein Beispiel für einige Anforderungsbeschreibungen:

Anforderung A: Das System soll in der Lage sein, standardisierte Formate für die Kommunikation mit der Umgebung zu verwenden.

Anforderung B: Der Benutzer soll in der Lage sein, Informationen vom System zu drucken. Die Druckfunktion soll immer dem Benutzer angeboten werden, wenn Informationen vom System angezeigt werden.
Anforderung C: Das System soll Mehrsprachigkeit unterstützen.

3. Anlass/Nutzen: Dieses Attribut besteht aus zwei Teilen:

ANLASS: Warum wurde diese Anforderung spezifiziert?

NUTZEN: Was ist der Nutzen dieser Anforderung? Der Nutzen soll im Kontext des Subjekts in der Anforderung betrachtet werden. Ist das Subjekt der Benutzer, so soll sein Nutzen erläutert werden (siehe Anforderung B im Folgenden Beispiel). Ist das Subjekt das Produkt selbst, z.B. im Falle einer nichtfunktionalen Anforderung, so soll der Nutzen aus der Anforderungsperspektive betrachtet werden (siehe Anforderungen A und C im Folgenden Beispiel):

Anforderung A: ANLASS: Ausgaben vom System in anderen Systemen verwenden. NUTZEN: Betriebsfähigkeit in der Umgebung von anderen Systemen.
Anforderung B: ANLASS: Der Benutzer benötigt Informationen in ausgedruckter Form. NUTZEN: Benutzer kann auswählen, wie er auf Informationen zugreift und/oder verteilt.
Anforderung C: ANLASS: Viele Benutzer verstehen die Sprache der Anwendung nicht und bevorzugen ihre Muttersprache. NUTZEN: Die Verwendung des Systems ist attraktiver (und einfacher) im internationalen Umfeld.

4. Abgrenzung/Risiko: Dieses Attribut beschreibt die Abgrenzung und/oder das Risiko, das in der Anforderungsbeschreibung nicht offensichtlich ist. Im Folgenden ein Beispiel für dieses Attribut:

Anforderung A: Verwendung von Standards bedeutet auch Abhängigkeit von anderen externen Organisationen.
Anforderung B: Kommunikation mit Druckergeräte könnte ein Problem sein. Die Art der Kommunikation soll detaillierter beschrieben werden.
Anforderung C: Gibt es eine Abgrenzung für Sprachen mit lateinischem Alphabet?

6.1.2 Place:

RAM schlägt vier Abstraktionsebenen vor. In diesem zweiten Schritt wird jede Anforderung analysiert und einer der folgenden Abstraktionsebenen (Abstraction Levels) zugeordnet.

Product Level: Das ist die abstrakteste Ebene. Anforderungen auf dieser Ebene sind den Geschäftszielen ähnlich und erfüllen die Definition einer Anforderung als eindeutig und testbar [30] im Allgemeinen nicht. Deshalb ist die Bezeichnung „Anforderung“ in diesem Fall fragwürdig. In RAM wird der Begriff „Anforderung“ allerdings für alle Aussagen auf den vier Abstraktionsebenen verwendet.

Anforderungen auf dieser Ebene sind so abstrakt, dass sie mit der Produktstrategie und der Organisationsstrategie vergleichbar sind. Die Produktstrategie umfasst z.B. Ziele und Visionen, die das Produkt betreffen.

Feature Level: Das ist die nächste, weniger abstrakte Ebene. Die Anforderungen auf dieser Ebene sind Features, die vom Produkt unterstützt werden. Sie enthalten keine Details über die benötigten Funktionen, die ein Feature anbietet.

Function Level: Auf dieser Ebene liegen die funktionalen und nichtfunktionalen Anforderungen (siehe Abschnitt 2.1). Anforderungen auf dieser Ebene sind im Allgemeinen detailliert und vollständig genug, um von einem Software-Designer evaluiert und modelliert zu werden.

Component Level: Das ist die letzte Abstraktionsebene in RAM. Anforderungen auf dieser Ebene beschreiben in sehr detaillierter Form, evtl. mit Beispielen, *wie* ein Problem gelöst werden soll.

Abbildung 6.2 zeigt die Abstraktionsebenen von RAM.

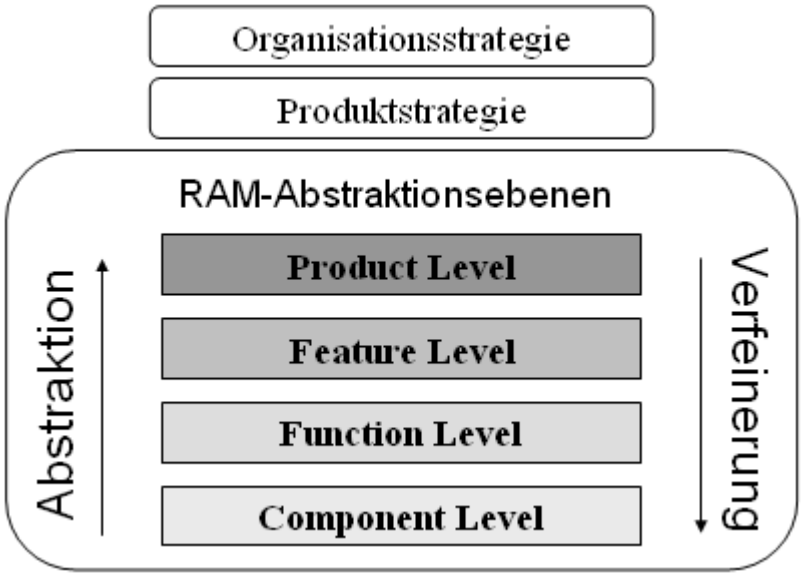


Abbildung 6.2: Abstraktionsebenen von RAM

RAM bietet einen Leitfaden, um eine Anforderungen einer der Abstraktionsebenen zuzuordnen. Dieser Leitfaden basiert auf einer Reihe von Fragen, die in Diagramm 6.1 dargestellt sind:

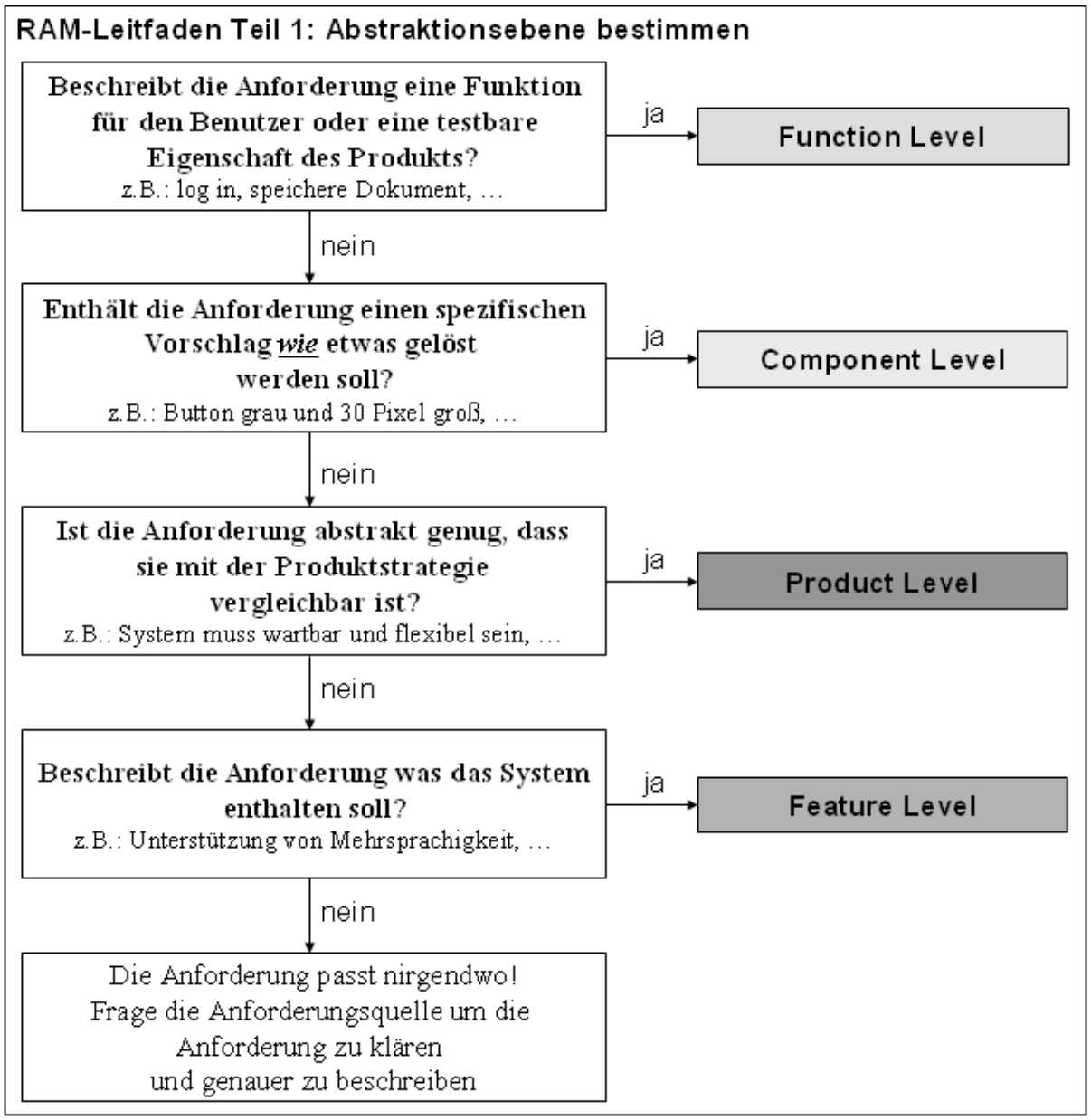


Diagramm 6.1: RAM-Leitfaden zur Bestimmung der Abstraktionsebene

Anhand dieses Leitfadens wird im Folgenden Beispiel versucht, die Anforderung C aus dem letzten Abschnitt „Unterstützung von Mehrsprachigkeit“ einer Abstraktionsebene zuzuweisen:

Die erste Frage, ob die Anforderung funktional ist und ob sie testbare Eigenschaften des Systems beschreibt, trifft nicht zu. Diese Anforderung beschreibt nicht, *wie* etwas gelöst werden soll, deshalb ist die zweite Frage auch zu verneinen. Die dritte Frage ist, ob die Anforderung abstrakt genug ist, um mit der Produktstrategie verglichen werden zu können. Die Anforderung C ist zwar abstrakt, die dazugehörige Anforderung auf der Produkt-Ebene lautet aber sehr wahrscheinlich „Produkt im internationalen Markt einführen“. Letztendlich wird die vierte und letzte Frage gestellt, ob die Anforderung beschreibt, was das System leisten oder bieten soll. Diese Frage wird mit „Ja“ beantwortet. Deshalb wird Anforderung C der Feature-Ebene zugeordnet.

6.1.3 Abstraction:

Im dritten Schritt wird eine Anforderung – abhängig von der im zweiten Schritt bestimmten Abstraktionsebene – anschließend abstrahiert und/oder verfeinert. Dieser Schritt beinhaltet zwei Regeln:

1. Regel (R1): Liegt eine Anforderung auf der Feature-Ebene oder niedriger, dann soll sie bis zur Produkt-Ebene abstrahiert werden. Im Abstraktionsprozess werden, je nach Situation, neue Anforderungen in der nächst höheren Abstraktionsebene oder Verlinkungen mit bereits existierenden Anforderungen erstellt. Ziel der Abstraktion ist, dass jede Anforderung in Verbindung mit mindestens einem Produktziel steht.

2. Regel (R2): Liegt eine Anforderung auf der Produkt- oder Feature-Ebene, dann soll sie mindestens bis zur Funktions-Ebene verfeinert werden. Ziel der Verfeinerung ist, dass jede Anforderung in einer Detaillierungsstufe liegt, um als Basis für den Entwurf zu dienen.

Diagramm 6.2 stellt einen Leitfaden dar, wie eine Anforderung abstrahiert und/oder verfeinert wird:

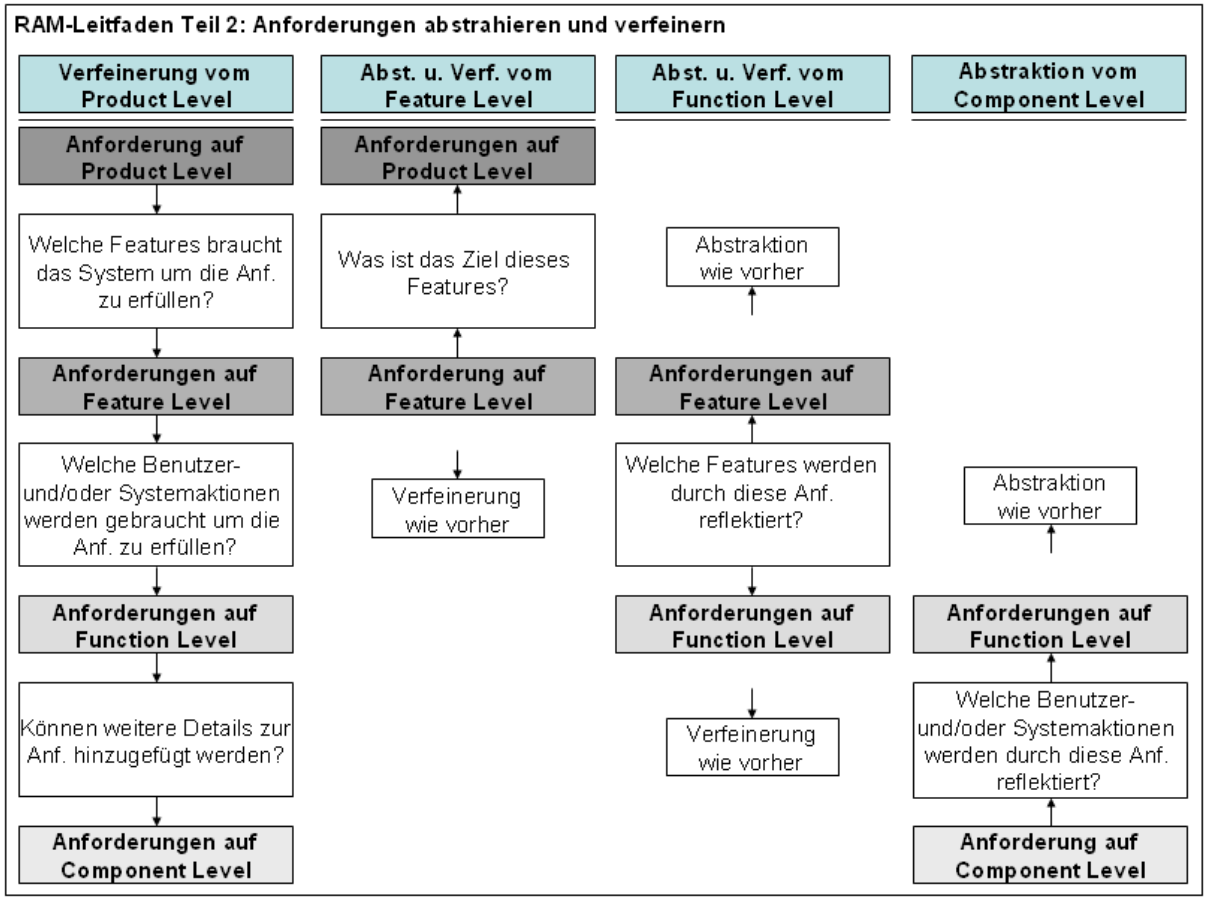


Diagramm 6.2: Leitfaden zur Abstraktion und/oder Verfeinerung

Als Beispiel wird wieder die Anforderung C „Unterstützung von Mehrsprachigkeit“ aus dem letzten Abschnitt betrachtet. Diese Anforderung wurde im zweiten Schritt „Place“ in der Feature-Ebene platziert.

Nach Regel Nummer 1 wird eine neue Anforderungen auf der Produkt-Ebene erzeugt, z.B. „Produkt im internationalen Markt einführen“ oder „Internationale Bedienbarkeit“ und mit der Anforderung C verlinkt. Falls solche Anforderung auf der Produkt-Ebene bereits existiert, so wird Anforderung C mit dieser Anforderung lediglich verlinkt.

Nach Regel Nummer 2 wird die Anforderung C in eine oder mehreren funktionalen Anforderungen verfeinert, wie z.B. „Sprache auswählen“, „Daten in der ausgewählten Sprache anzeigen“ und „Neue Sprache zum System hinzufügen“.

Die neu erzeugten funktionalen Anforderungen können im nächsten Schritt weiter verfeinert werden. Es werden Vorschläge gemacht, wie diese Anforderungen implementiert werden sollen, wie z.B. „Web-Schnittstelle zum Hinzufügen einer neuen Sprache“. Diese Anforderung liegt dann auf der Komponenten-Ebene.

Abbildung 6.3 zeigt die Originalanforderung C und die im Rahmen der Abstraktion und Verfeinerung neu erzeugten Anforderungen:

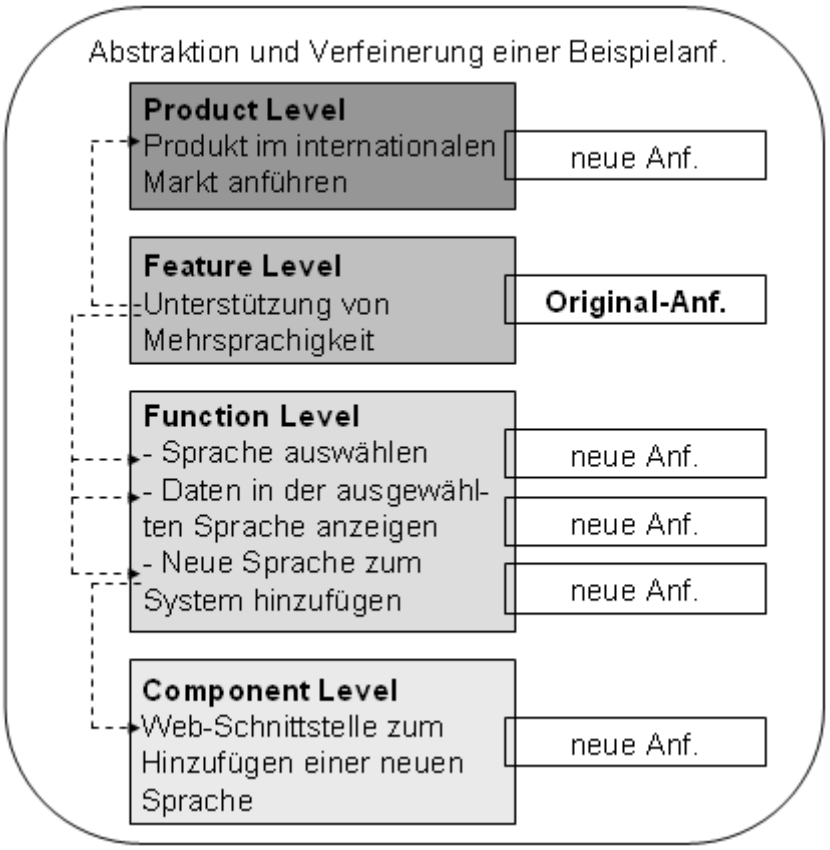


Abbildung 6.3: Abstraktion und Verfeinerung einer Beispielanforderung

6.2 Verfeinerung von Sicherheitsanforderungen

Wie im letzten Abschnitt vorgeführt, besteht die Verfeinerung von Anforderungen nach RAM aus zwei Teilen:

Teil 1: Der Anforderung einer Abstraktionsebene zuordnen (siehe Diagramm 6.1).

Teil 2: Neue Anforderungen mit mehr Details herleiten (siehe Diagramm 6.2).

RAM wird in dieser Arbeit für die Zuordnung der Sicherheitsanforderungen den Abstraktionsebenen verwendet (Teil 1).

Die Fragen für die Herleitung von neuen Anforderungen in RAM (Teil 2) sind sehr unspezifisch und eignen sich nicht für die Verfeinerung von sicherheitskritischen Anforderungen. Ein Ziel dieser Arbeit ist, die sicherheitskritischen Anforderungen abhängig von ihrem UMLsec-Stereotyp zu verfeinern. Dieser Aspekt fehlt komplett in RAM.

Die Deutschen IT-Sicherheitskriterien[6] bieten einige sicherheitsspezifische Fragen, um Informationen für bestimmte Sicherheitsfunktionen zu erheben. Tabelle 6.1 zeigt die Fragen für die Sicherheitsfunktionen „Identifikation und Authentisierung“, „Rechteverwaltung“, „Rechteprüfung“ und „Beweissicherung“:

Sicherheitsfunktion	Fragen
Identifikation **u. Authentisierung**	Welche Subjekte und Objekte müssen identifiziert werden?
	Welche Subjekte und Objekte müssen identifiziert und authentisiert werden?
	Unter welchen Umständen muss eine Identifikation und Authentisierung erfolgen?
	Welche Aktionen müssen bei nicht erfolgreicher Identifikation und Authentisierung ergriffen werden?
Rechteverwaltung	Welche Subjekte bzw. Subjektklassen und welche Objekte bzw. Objektklassen unterliegen der Rechteverwaltung?
	Welche Arten von Rechten können zwischen Subjekten und Objekten existieren?
	Wer darf Rechte vergeben bzw. ändern?
	Welche Regeln müssen bei der Vergabe bzw. Änderung von Rechten eingehalten werden?
	Welche Voraussetzungen müssen vor einer Vergabe oder Änderung von Rechten erfüllt sein?
	Welche Rollen müssen durch die Rechteverwaltung definiert werden?
	Welche Rechte sind an spezielle Rollen gebunden?
	Welche Rollen sind miteinander unvereinbar?
Rechteprüfung	Bei welchen Aktionen soll eine Rechteprüfung erfolgen?
	Welche Aktionen sollen ergriffen werden, wenn versucht wird, eine Aktion ohne das zugehörige Recht auszuführen?
	welche Ausnahmen soll es bei der Rechteprüfung geben und unter welchen Umständen sollen diese Ausnahmen gültig sein?
Beweissicherung	Welche Ereignisse sollen protokolliert werden?
	Welche Informationen sollen dabei aufgezeichnet werden?
	Wo sollen diese Informationen aufgezeichnet werden?
	wer, wie und wann darf auf diese Informationen zugreifen?
	Nach welchen Kriterien sollen diese Informationen ausgewertet werden?

Tabelle 6.1: Fragen aus den Deutschen IT-Sicherheitskriterien

Diese Fragen werden in dieser Arbeit für die Verfeinerung von sicherheitskritischen Anforderungen verwendet. Die Fragen für die Sicherheitsfunktion „Identifikation und Authentisierung“ eignen sich für sicherheitskritische Anforderungen des UMLsec-Stereotyps „secrecy“ und die für „Rechteprüfung“ und „Rechte Verwaltung“ für sicherheitskritische Anforderungen des UMLsec-Stereotyps „integrity“.

Da die vorgeschlagenen Fragen in den Deutschen IT-Sicherheitskriterien nicht alle UMLsec-Stereotypen abdecken, muss auf die Erfahrungen von Sicherheitsexperten zurückgegriffen werden, um die fehlenden Fragen zu formulieren. In dieser Arbeit wird bei der Umsetzung

[6] http://www.bsi.de/zertifiz/itkrit/dtitsec.htm

mit HeRA (siehe Abschnitt 8.2.4) den Anforderungsanalysten die Möglichkeit angeboten, interaktiv neue Fragen einzutragen und zu bewerten. Somit werden über die Zeit immer neue Fragen für die Verfeinerung von sicherheitskritischen Anforderungen gesammelt, die im Nachhinein für andere Projekte verwendet werden können. Die Bewertung der Fragen, ob diese erfolgreich zu einer Verfeinerung durchgeführt haben, ermöglicht den Erfahrungsaustausch zwischen Anforderungsanalysten um bessere Ergebnisse zu erzielen.

7. Informationsflüsse im Sicherheitsentwicklungsprozess

In diesem Kapitel wird der im Rahmen dieser Arbeit entwickelte Prozess zur Klassifizierung, Verfeinerung und Modellierung von Anforderungen vorgestellt. Der Prozess beschreibt wer was wann mit welchem Tool welche Aktion durchführen soll. Um die Informationsflüsse in diesem Prozess zu modellieren, wird FLOW [26] verwendet, weil es eine einfache und überschaubare Notation zur Modellierung von Informationsflüssen anbietet.

7.1 FLOW

Um die Informationsflüsse zu modellieren wird FLOW verwendet. Das Forschungsprojekt FLOW wurde 2004 mit Gründung des Fachgebiets Software Engineering an der Leibniz Universität Hannover initiiert und beschäftigt sich mit der systematischen Auseinandersetzung mit Informationsflüssen in der Softwareentwicklung.

Eine wichtige Eigenschaft von FLOW ist die Modellierung von festen (Dokumente, Quellcode, etc.) und flüssigen Informationsflüssen (informelle E-Mails, Chat-Protokolle, etc.). Um die explizite Unterscheidung zwischen festen und flüssigen Informationsflüssen zu erlauben, wurde in FLOW eine Notation entwickelt, die im Wesentlichen je ein Symbol für feste und flüssige Informationsspeicher sowie je ein Pfeiltyp für Informationsflüsse enthält. Weiterhin wird zwischen Projektinformationen (schwarze Pfeile) und Erfahrungen (graue Pfeile) unterschieden. Abbildung 7.1 zeigt die Notationselemente von FLOW.

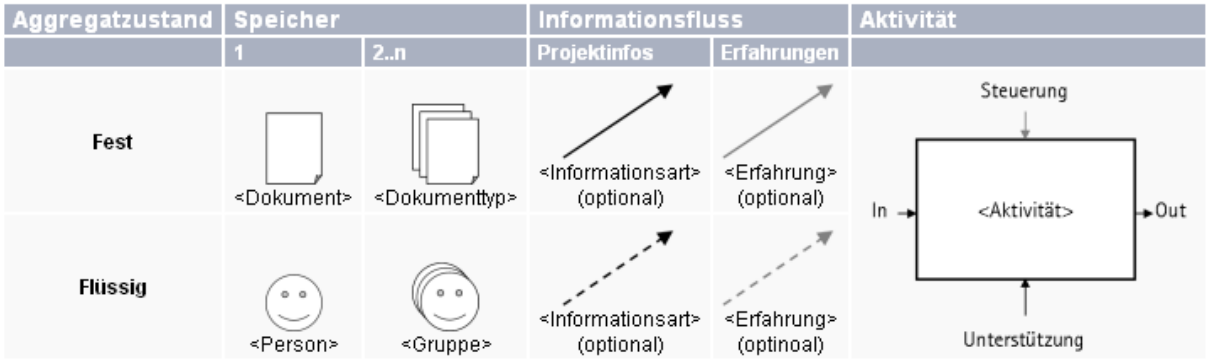


Abbildung 7.1: Notation in FLOW [26]

Das Aktivitätensymbol wurde in Anlehnung an IDEFØ[7] verwendet:

- **Links:** Eingehende Flüsse, die in der Aktivität verarbeitet werden
- **Rechts:** Ausgehende Flüsse.
- **Unten:** Mechanismen zur Unterstützung der Ausführung einer Aktivität wie beispielsweise Tools oder Personen, die eine Aktion durchführen.
- **Oben:** Steuernde und kontrollierende Informationsquellen wie beispielsweise Erfahrungen oder spezielle Kenntnisse einer Person.

Das Aktivitätensymbol dient einerseits als Black-Box, die Details von Informationsflüssen zusammenfassen bzw. verbergen und andererseits als Verbindung zu bestehenden Prozessnotationen. Im Folgenden Abschnitt wird diese Notation zur Modellierung von Dokumenten, Aktivitäten, beteiligten Personen sowie die Informationsflüsse im gesamten Prozess – von

[7] http://www.idef.com/idef0.html

dem Aufbau der Begriffsdatenbank, über die Klassifizierung der Anforderungen bis hin zur Verfeinerung und Modellierung der Anforderungen mit UMLsec – verwendet.

7.2 Aufdeckung, Verfeinerung und Modellierung von Sicherheitsanforderungen

Im ersten Schritt wird die Begriffsdatenbank gebaut. Für diesen Zweck muss zunächst eine Menge von bereits vorhandenen Anforderungen durch einen Sicherheitsexperten in zwei Mengen – Sicherheitsanforderungen und Nicht-Sicherheitsanforderungen – manuell klassifiziert werden. Diese beiden Mengen von Anforderungen werden mit Unterstützung des Tools HeRA für den Aufbau der Begriffsdatenbank verwendet. Diese Anforderungen werden hier als Trainingsanforderungen bezeichnet. Können im Laufe der Zeit weitere Trainingsanforderungen mit Hilfe eines Sicherheitsexperten manuell klassifiziert werden, so kann dieser Prozess erneut durchgeführt werden um die Datenbank zu erweitern und dadurch aussagekräftigere Statistiken aus der Datenbank aufstellen zu können. Dieser Ablauf wird im oberen Teil „Build Data Base“ vom Diagramm 7.1 veranschaulicht.

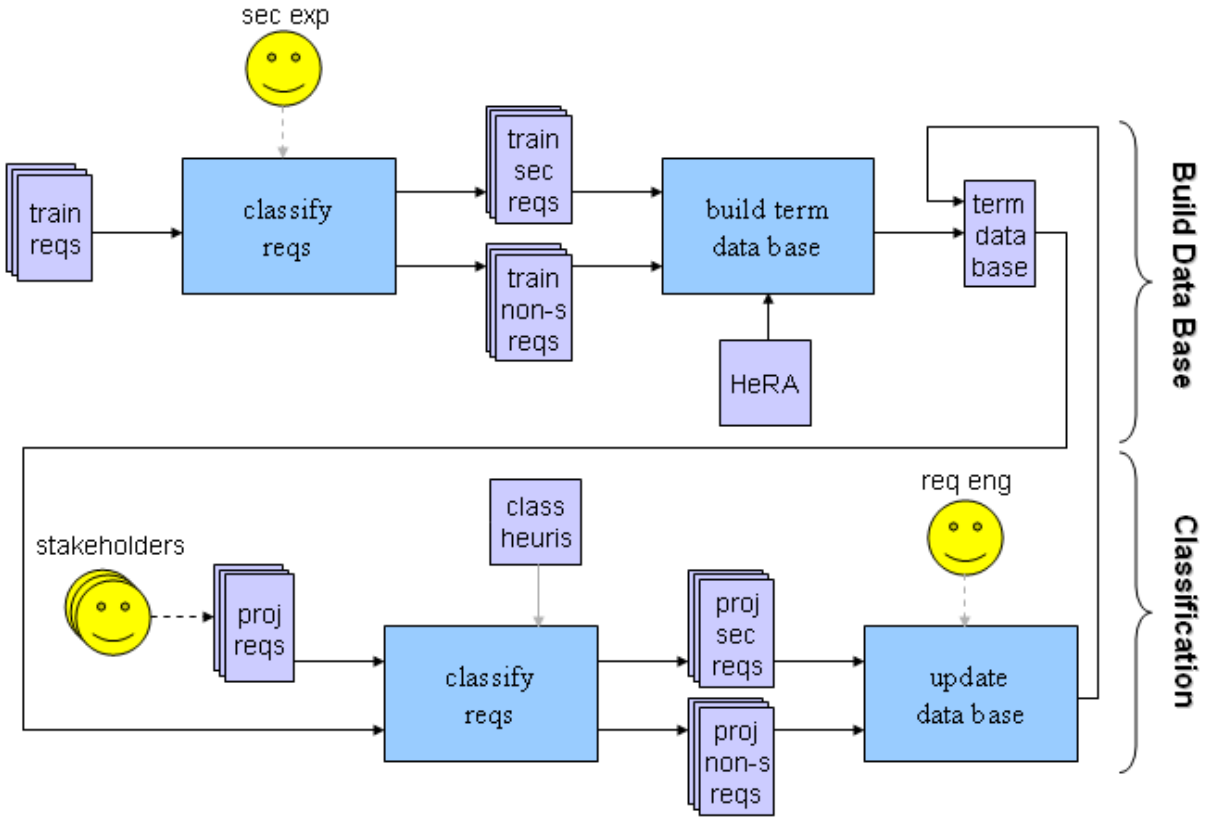


Diagramm 7.1: Begriffsdatenbank und Anforderungsklassifizierung

Sollen nun Anforderungen eines laufenden Softwareprojekts, hier Projektanforderungen genannt, klassifiziert werden, ohne dass ein Sicherheitsexperte zur Verfügung steht, um möglichst früh potenzielle Sicherheitsanforderungen zu entdecken, so kann das Tool HeRA für die Klassifizierung verwendet werden. Für diesen Zweck greift die im Rahmen dieser Arbeit entwickelte Erweiterung von HeRA auf die bereits konstruierte Datenbank und die in Kapiteln 4 und 5 formulierten Klassifizierungsheuristiken. Die Projektanforderungen können sowohl sicherheitskritische Anforderungen (auf abstrakter Ebene) als auch Sicherheitsanforderungen (funktional) enthalten.

Die von HeRA vorgenommene Klassifizierung wird von den Anforderungsanalysten genutzt, um sicherheitskritische Anforderungen zu entdecken. Ist der Anforderungsanalyst der Meinung, dass die Klassifizierung von HeRA korrekt ist, so kann er diese neue Erkenntnis in die Datenbank einfließen lassen. Dieser Ablauf ist im unteren Teil „Classificatoin“ vom Diagramm 7.1 modelliert.

In zweiten Schritt können die als sicherheitskritisch identifizierten Anforderungen verfeinert werden, falls diese auf einer abstrakten Ebene liegen, wie z.B. Geschäftsanforderungen. Um dem Anforderungsanalyst Unterstützung zu bieten, die abstakten Anforderungen zu konkreten funktionalen Anforderungen zu verfeinern, werden ihm abhängig vom UMLsec-Stereotyp, der der Anforderungen zugeordnet wurde, Hinweise vorgeschlagen, welche Daten erhoben werden müssen, mit deren Hilfe verfeinerte Anforderungen formuliert werden können. Diese Hinweise werden von den Deutschen IT-Sicherheitskriterien[8] abgeleitet (siehe Abschnitt 6.2). Der Ablauf dieses Prozesses wird im Diagramm 7.2 modelliert.

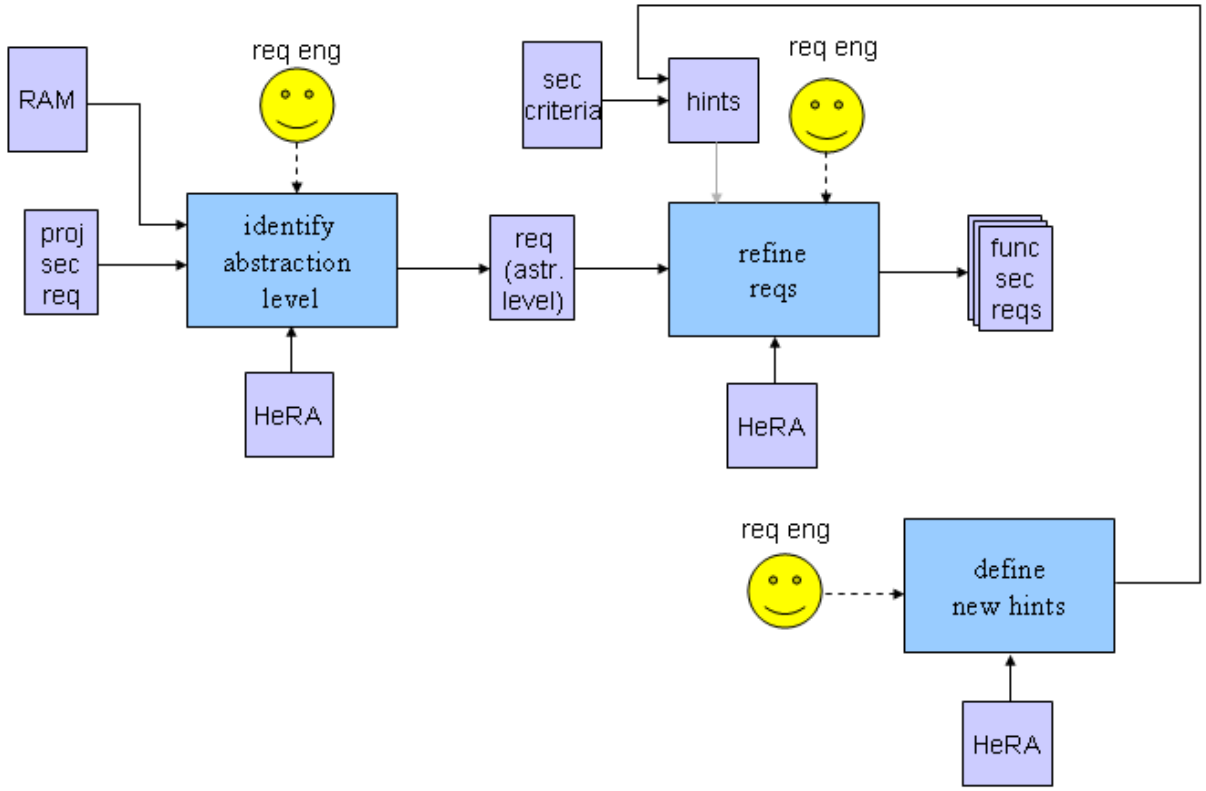


Diagramm 7.2: Verfeinerung von Anforderungen

Des Weiteren kann ausgehend von den identifizierten sicherheitskritischen Anforderungen den Anforderungsanalysten und Software-Designern Unterstützung bei der Modellierung der Sicherheitsanforderungen mit UMLsec geboten werden. Anhand von Fragen, die aus UMLsec-Beispielen abgeleitet wurden, sammelt der Software-Analyst Daten, die für die Erstellung eines unvollständigen UMLsec-Diagramms mit Hilfe von HeRA verwendet werden. Dieser Prozess wird im oberen Teil vom Diagramm 7.3 veranschaulicht.

[8] http://www.bsi.de/zertifiz/itkrit/dtitsec.htm

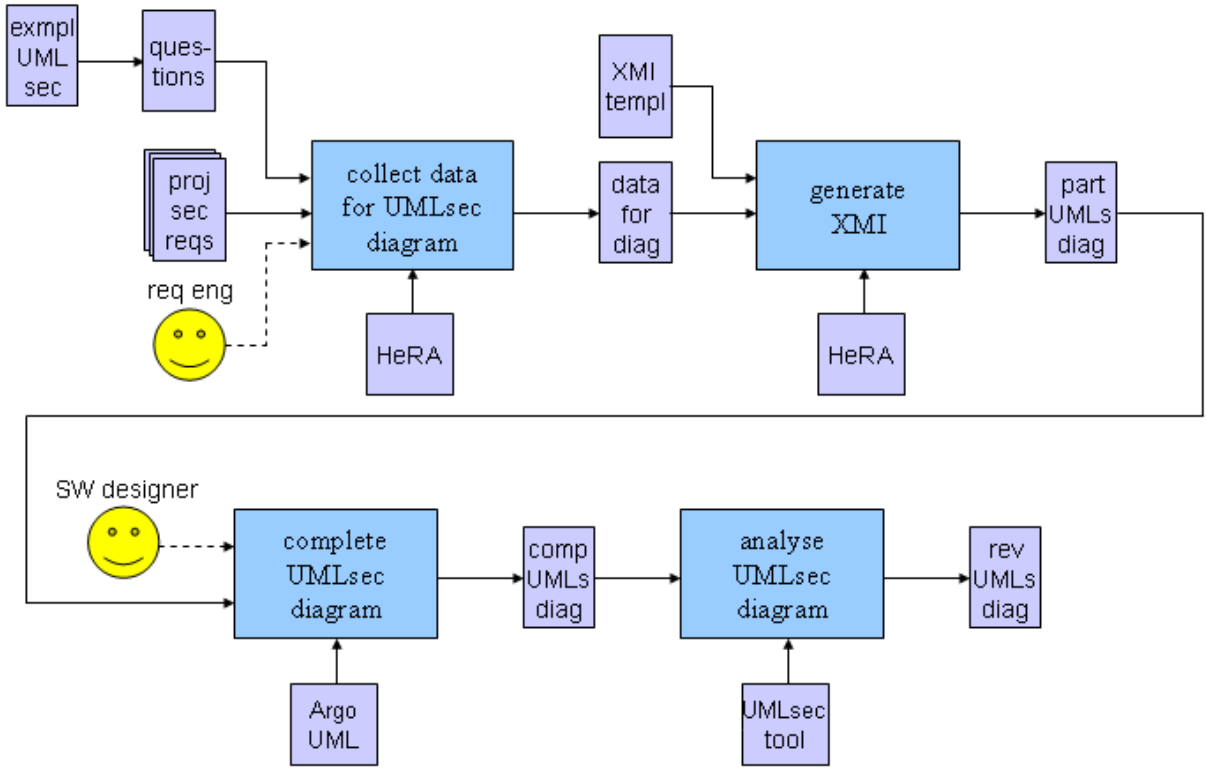


Diagramm 7.3: Erstellung von UMLsec-Diagrammen

Ist das unvollständige Diagramm automatisch mit HeRA erstellt worden, so muss zunächst der Software-Designer das Diagramm manuell mit einem UML-Editor (beispielsweise ArgoUML[9]) vervollständigen. Das vollständige Diagramm kann im nächsten Schritt mit Hilfe des UMlsec-Tools[10] auf Richtigkeit analysiert werden. Dieser Ablauf ist im unteren Teil vom Diagramm 7.3 darstellt.

[9] http://argouml.tigris.org
[10] http://computing-research.open.ac.uk/jj2924/umlsectool/users.html

8. Umsetzung mit HeRA

HeRA (Heuristic Requirements Assistent) ist ein Feedback-Basiertes Werkzeug zur Unterstützung der Anforderungsanalysten zur Erhebung von konsistenten, vollständigen und hochqualitativen Anforderungen. HeRA wurde am Fachgebiet Software Engineering der Leibniz Universität Hannover entwickelt. Basierend auf eine Reihe von heuristischen Regeln unterstützt HeRA die Analysten mit wichtigen Informationen, wie beispielsweise wie konsistent eine Anforderungen mit anderen ist oder welche Begriffe in einer Anforderungen vermieden werden sollen und wie diese besser formuliert werden können.

8.1 HeRA-Komponenten

HeRA basiert auf Fischer's Architektur für DODE (domain oriented design enviroments) [23]. Der zentrale Teil dieser Architektur ist eine Konstruktionskomponente (construction component). Im Fall von HeRA sind es Editoren für Anforderungen, Use-Cases und ein Glossar, mit deren Hilfe die Anforderungen erfasst werden. Weiterhin bietet HeRA weitere zwei DODE-Komponenten, die Argumentationskomponente (argumentation component) und die Simulationskomponente (simulation component). Die Argumentationskomponente in HeRA besteht aus zwei Teilen, eine Kritikansicht (Critic View) und eine Problemansicht (problem view). Die Kritikansicht zeigt den Benutzern Erfahrungen (Hinweise, Warnungen oder allgemeine Informationen) zur Formulierung der aktuell zu bearbeitende Anforderung bzw. Use-Case, während die Problemansicht eine Zusammenfassung aller Anforderung bzw. Use-Cases darstellt, für die HeRA Warnungen oder Hinweise vorschlägt. Die Simulationskomponente gibt den Benutzern einen Ausblick, welche Auswirkungen die formulierte Anforderung auf die Use-Case-Modellierung haben könnte [7]. Abbildung 8.1 zeigt eine Übersicht der Komponenten von HeRA.

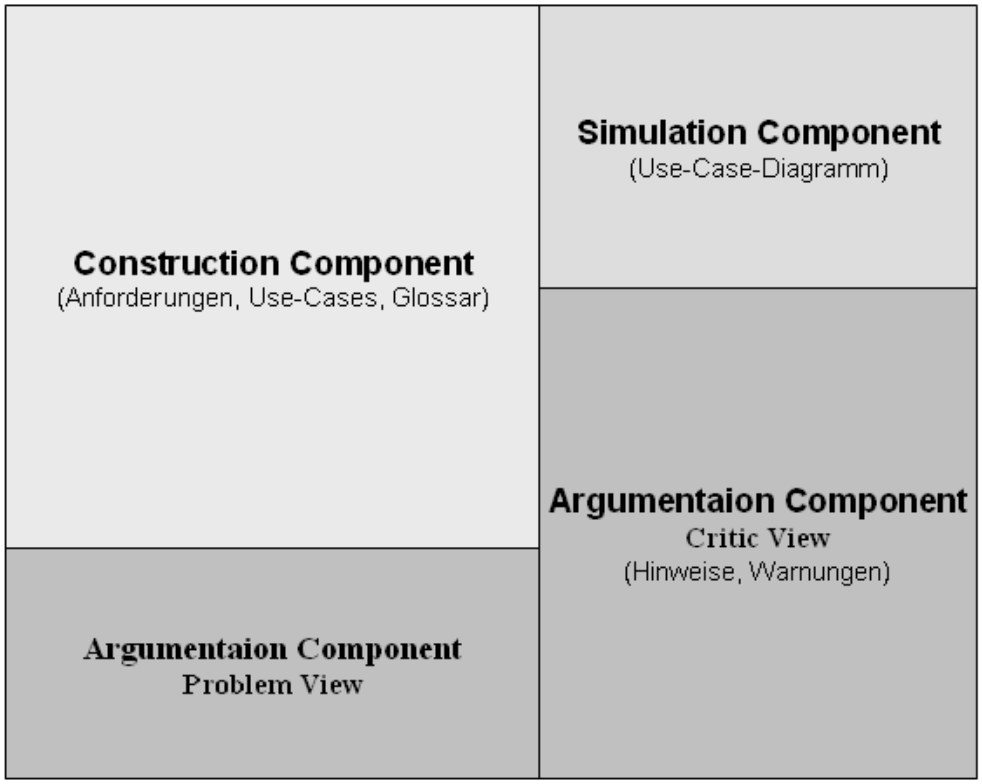


Abbildung 8.1: Übersicht der Komponenten von HeRA

Im Rahmen einer Bachelorarbeit am Fachgebiet Software Engineering der Leibniz Universität Hannover [25] wurde das Kritiksystem von HeRA erweitert und verbessert, sodass der Anwender zur Laufzeit neue Heuristiken und Erfahrungen erfassen oder bestehende anpassen kann. Abbildung 8.2 zeigt ein Screenshot von HeRA. Links wird eine Übersicht angezeigt, welche Anforderungen und Use-Cases für ein bestimmtes Projekt erfasst wurden. Diese können ausgewählt und in der Konstruktionskomponente in der Mitte oben bearbeitet werden. Rechts oben ist die Simulationskomponente, die eine Graphische Darstellung des Use-Cases darstellt. Rechts unten ist die Kritikansicht. In diesem Bereich werden dem Benutzer basierend auf heuristischen Regeln Erfahrungen angezeigt, die eine Unterstützung bei der Formulierung anbieten, um hochqualitative Anforderungen zu erfassen. Alle im Projekt vorhanden Anforderungen bzw. Use-Cases, für die das Kritiksystem Erfahrungen anzeigt, werden in der Problemansicht in der Mitte unten angezeigt.

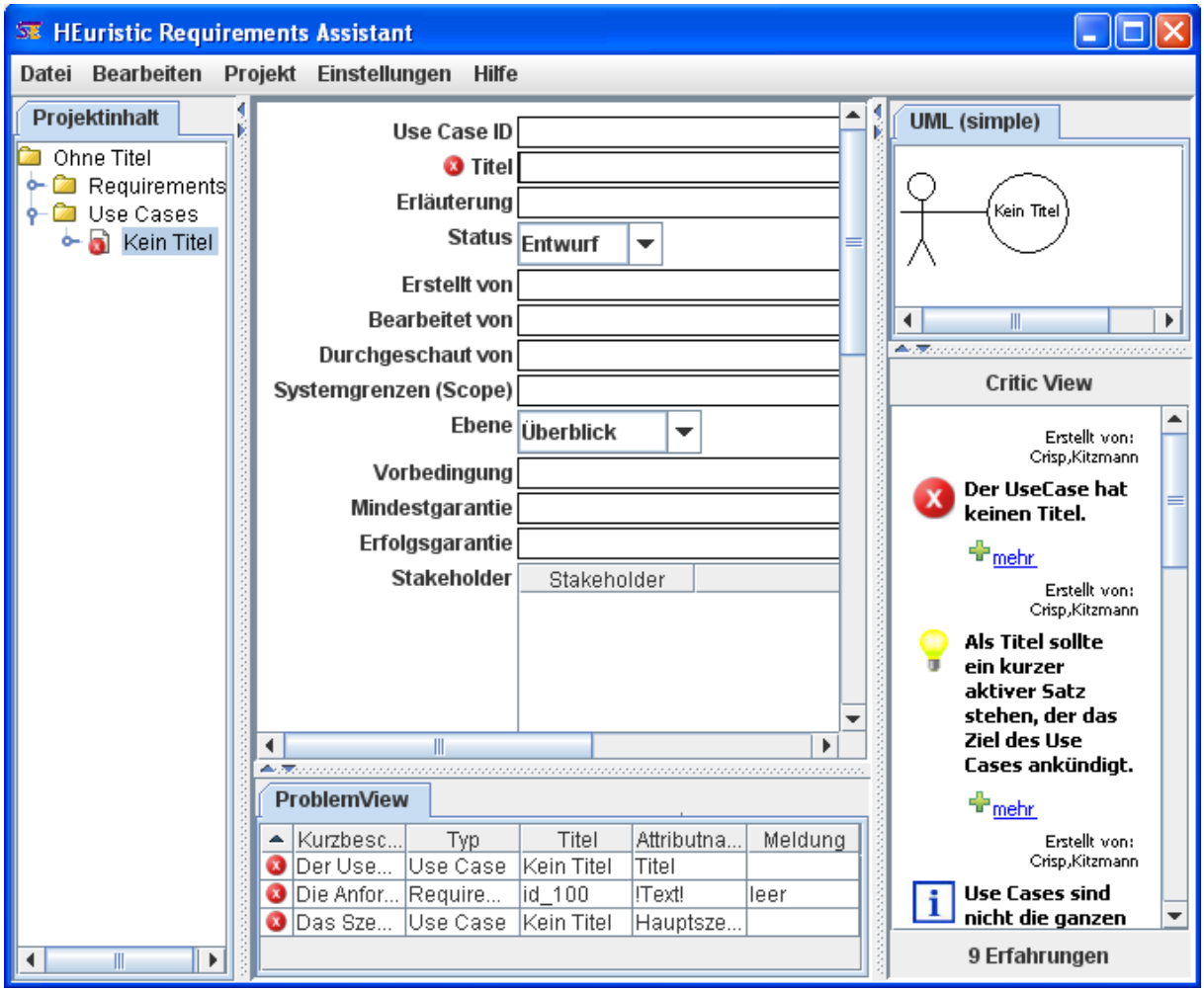


Abbildung 8.2: Screenshot von HeRA

Die Erfahrungen können in HeRA mit Hilfe einer Erfahrungsbasis verwaltet werden [25]. Abbildung 8.3 zeigt ein Screenshot der Erfahrungsbasis. Es kann zwischen verschiednen Erfahrungstypen (Fehler, Warnung, Hinweis, Information) unterschieden werden. Erfahrungen können näher erläutet und gewichtet werden. Im Bereich „Parameter“ werden die Begriffe festgelegt, bei denen diese Erfahrung in der Kritikansicht in HeRA angezeigt wird. Weiterhin bietet die Erfahrungsbasis einen Wizard zur Implementierung der Heuristiken in JavaScript. Damit können Erfahrungen sehr flexibel zur Laufzeit erweitert oder angepasst werden.

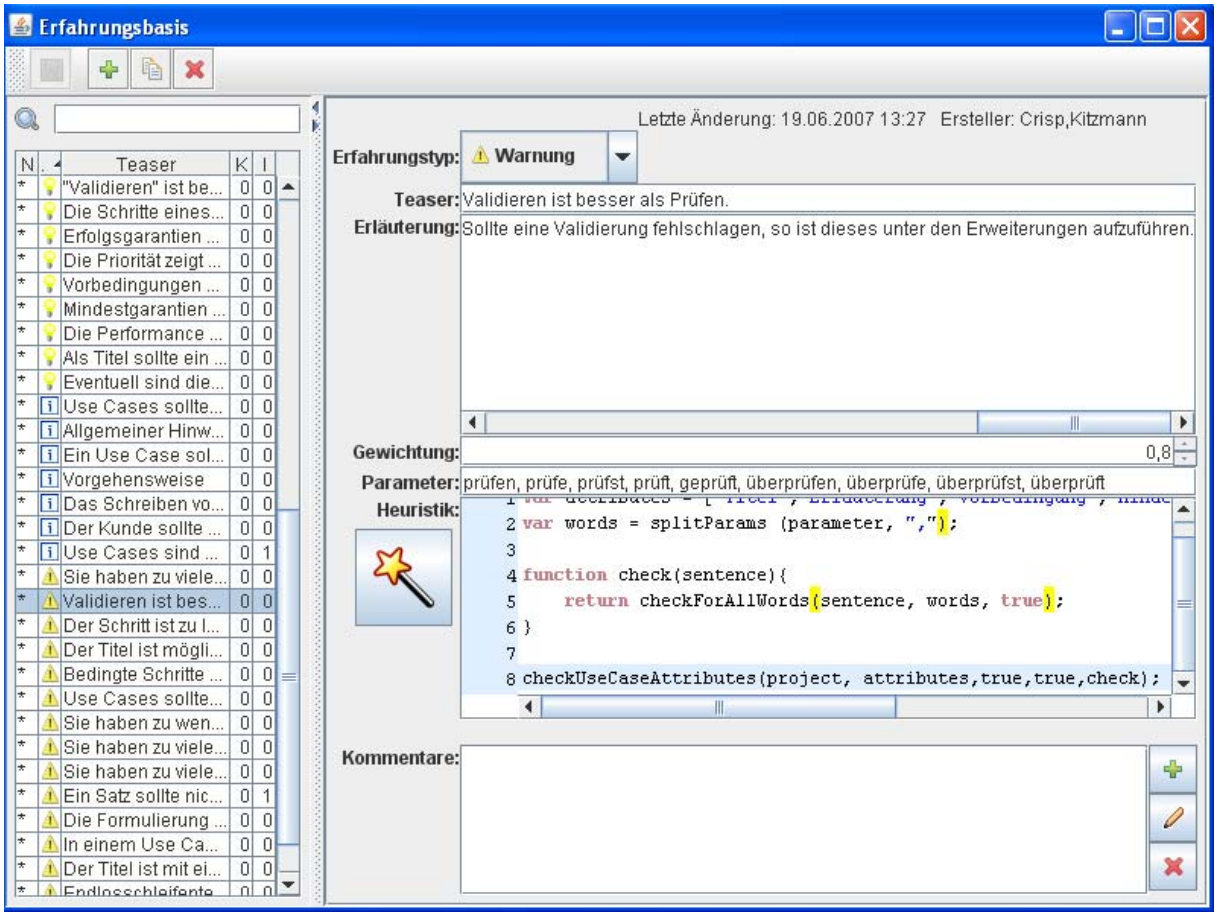


Abbildung 8.3: Erfahrungsbasis in HeRA

8.2 Erweiterung von HeRA

Analog zum Kritiksystem im letzten Abschnitt wurde in dieser Arbeit ein Sicherheitskritiksystem bestehend aus Sicherheitsansicht (Securtiy View) und Sicherheitskritikansicht (Securtiy Critic View) entwickelt. Das Sicherheitskritiksystem basiert auf den in Kapiteln 4 und 5 vorgestellten Klassifizierungsheuristiken. Neben dem „Problem View“ wurde ein weiterer Reiter „Securtiy View“ gebaut, sodass zwischen „Critic View“ und „Security Critic View“ umgeschaltet werden kann. Diese beiden Ansichten können unabhängig von einander verwendet werden. Somit kann sich der Anforderungsanalyst zunächst in der „Critic View“ auf die Formulierung der Anforderung konzentrieren und im Anschluss zur „Security View“ wechseln, um zusätzlich eventuelle Sicherheitskriterien zu berücksichtigen.

In der „Security View“ werden alle Anforderungen zusammengefasst, die von der Klassifizierungsheuristik als sicherheitskritisch klassifiziert wurden. In der „Security Critic View“ wird angezeigt, dass es sich um eine sicherheitskritische Anforderung handeln könnte sowie der von der Klassifizierungsheuristik vorgeschlagene UMLsec-Stereotyp. Diese Ansicht enthält zusätzlich folgende Optionen:

- **Als Sicherheitsanforderung bestätigen:** Durch diese Option kann die Begriffsdatenbank mit einer Sicherheitsanforderung erweitert werden.
- **Als Nicht-Sicherheitsanforderung bestätigen:** Diese Option erweitert die Begriffsdatenbank mit einer Nicht-Sicherheitsanforderung.
- **Abstraktionsebene bestimmen:** Mit Hilfe dieser Option kann einer Anforderung eine Abstraktionsebene zugeordnet werden.

- **Verfeinerungshinweise anzeigen:** Diese Option zeigt Hinweise, wie eine Anforderung verfeinert werden kann. Zusätzlich können neue Hinweise hinzufügt werden.
- **UMLsec-Diagramm erstellen:** Mit Hilfe dieser Option kann ein UMLsec-Diagramm als eine XMI-Datei erstellt werden.
- **Datenbank erweitern:** Diese Option bietet die Möglichkeit, die Begriffsdatenbank für die dynamischen Klassifizierungsheuristiken zu erweitern (siehe Abschnitt 5.1).

Abbildung 8.4 zeigt die „Securtiy View“ und die „Securtiy Critic View“.

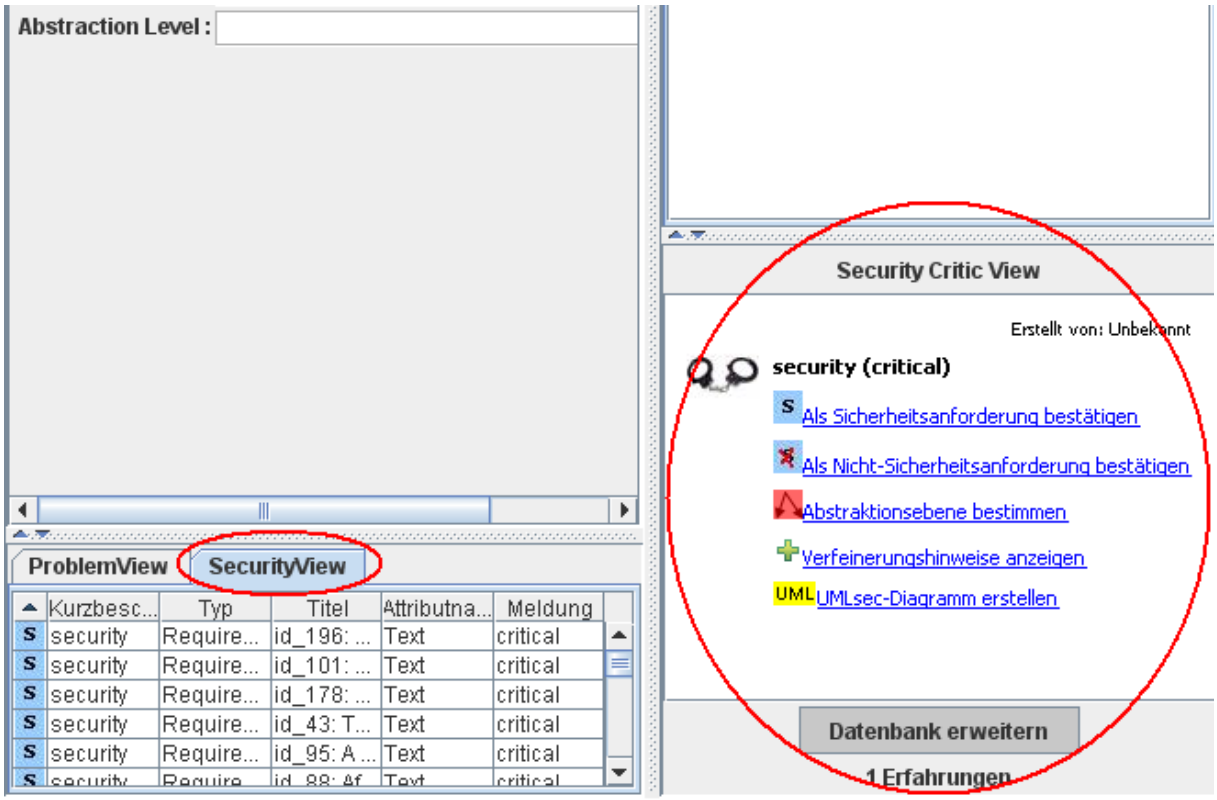


Abbildung 8.4: „Securtiy View“ und die „Securtiy Critic View“

Der Aufruf einer Klassifizierungsheuristik wird mit Hilfe der Erfahrungsbasis realisiert. Es wird eine neue Erfahrung hinterlegt und über einen Parameter gesteuert, welche der in Kapitel 4 und 5 vorgestellten Klassifizierungsheuristiken verwendet wird. Beispielsweise „KEYWORD“ für die statische Klassifizierungsheuristik „Suche nach Schlüsselwörter“ oder „BAYES_FILTER“ für die dynamische Klassifizierungsheuristik „Bayes-Filter“. Somit kann zur Laufzeit zwischen den verschiedenen Klassifizierungsheuristiken gewechselt werden.

Im Falle der statischen Klassifizierungsheuristiken, die auf der Suche nach Schlüsselwörtern basieren, müssen die Schlüsselwörter im Bereich „Parameter“ angegeben werden. Abbildung 8.5 zeigt ein Beispiel.

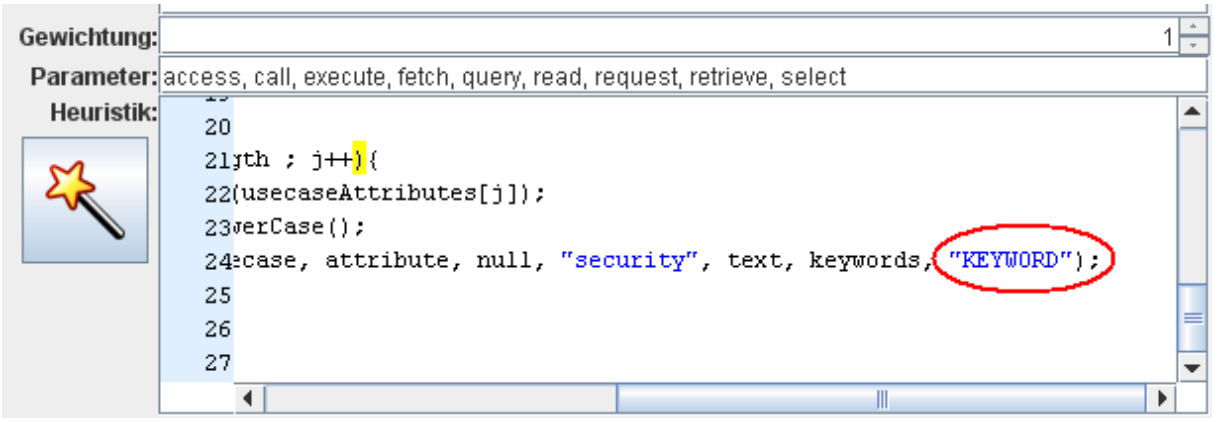


Abbildung 8.5: Aufruf einer Klassifizierungsheuristik

Im Falle der dynamischen Klassifizierungsheuristiken müssen keine Schlüsselwörter angegeben werden, da diese die Begriffsdatenbank für ihre Berechnungen verwenden.

8.2.1 Als Sicherheitsanforderung bestätigen

Die dynamischen Klassifizierungsheuristiken (siehe Kapitel 5) basieren bei deren Berechnungen auf die Begriffsdatenbank. Um bessere Ergebnisse bei der Klassifizierung zu erhalten, soll die Begriffsdatenbank erweitert werden. Der Anforderungsanalyst kann in der „Security Critic View“ auf den Link Als Sicherheitsanforderung bestätigen klicken. Daraufhin wird folgendes Fenster angezeigt:

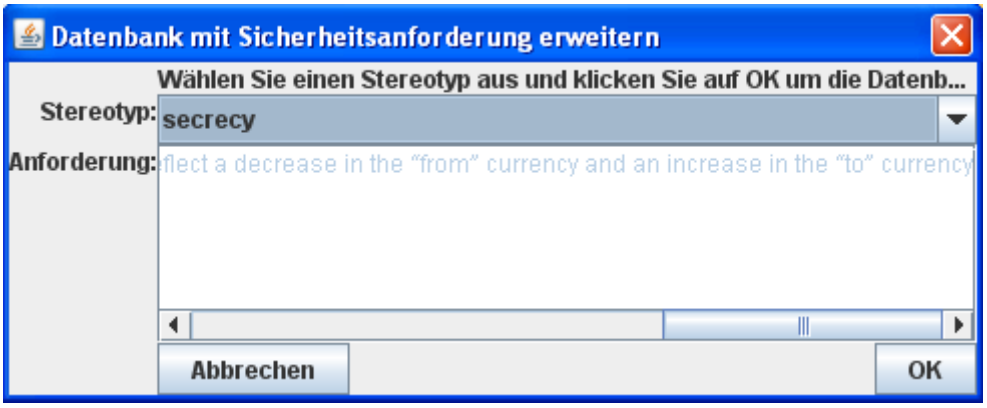


Abbildung 8.6: Datenbank mit Sicherheitsanforderung erweitern

Zusätzlich kann ein UMLsec-Stereotyp ausgewählt werden. Beim Klicken auf den Button „OK“ werden die Wörter in der Anforderung als sicherheitsrelevante Begriffe in die Begriffsdatenbank einfließen.

8.2.2 Als Nicht-Sicherheitsanforderung bestätigen

Die Datenbank kann weiterhin mit Nicht-Sicherheitsanforderungen erweitert werden. In diesem Fall muss der Link Als Nicht-Sicherheitsanforderung bestätigen angeklickt werden. Daraufhin erscheint eine Meldung zur Bestätigung der Aktion:

Abbildung 8.7: Bestätigung der Nicht-Sicherheitsanforderung

Wird auf „Ja“ geklickt, so werden die Wörter in der Anforderung als nicht-sicherheitsrelevante Begriffe in der Datenbank einfließen.

8.2.3 Abstraktionsebene bestimmen

Wie in Kapitel 6 beschrieben, wird mit RAM (Requirements Abstraction Modell) einer Anforderung eine Abstraktionsebene zugeordnet. Diese Zuordnung basiert auf eine Reihe von Fragen, die der Anforderungsanalyst zu beantworten hat. Um eine Anforderung in HeRA einer Abstraktionsebene zuzuweisen, klickt man auf den Link Abstraktionsebene bestimmen in der „Security Critic View“. Daraufhin erscheinen die Fragen, die der Anforderungsanalyst beantworten soll:

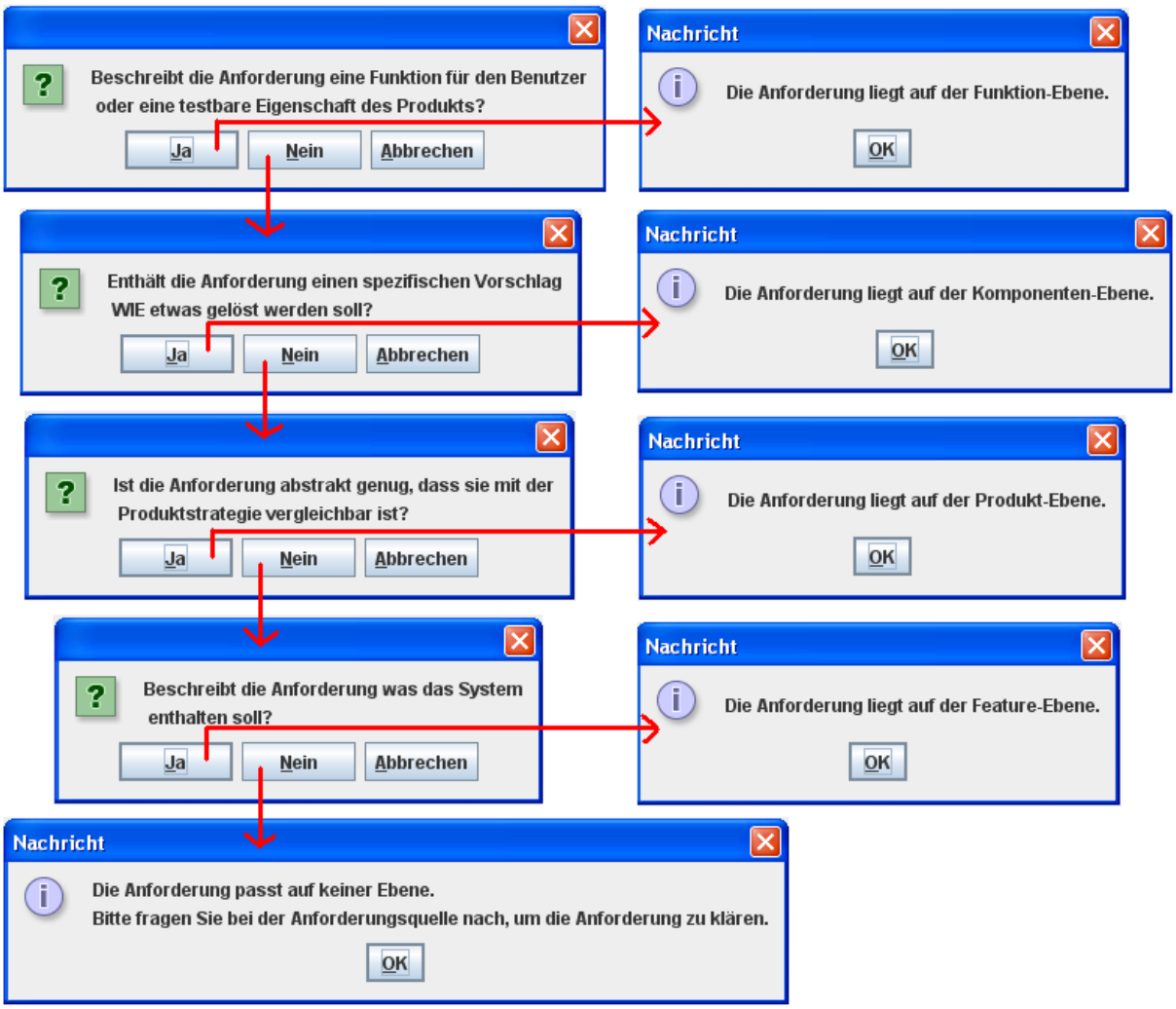


Abbildung 8.8: Abstraktionsebene bestimmen mit HeRA

Wird auf „Nein“ geklickt, so erscheint die nächste Frage. Beim Klicken auf „Ja“, wird die Anforderung einer Abstraktionsebene zugeordnet (siehe Diagramm 6.1). Das Anforderungsattribut „Abstraction Level“ wird automatisch mit dem entsprechend Wert gefüllt.

8.2.4 Verfeinerungshinweise anzeigen

Wird eine Anforderung als sicherheitskritisch klassifiziert und liegt diese auf einer abstrakten Ebene, so kann der Anforderungsanalyst Unterstützung von HeRA erhalten, um diese Anforderung zu verfeinern. Beim Anklicken auf den Link Verfeinerungshinweise anzeigen in der „Security Critic View“ werden abhängig vom UMLsec-Stereotyp der Anforderung eine Menge von Hinweise bzw. Fragen angezeigt. Diese Hinweise bzw. Fragen helfen dem Anforderungsanalyst, die Anforderung detaillierter zu beschreiben. Der Anforderungsanalyst kann die

Hinweise bewerten, ob sie für ihn hilfreich oder nicht hilfreich waren. Für diesen Zweck stehen hinter jeden Hinweis zwei Links: hilfreich und nicht hilfreich. Hinweise, die als hilfreich bewertet wurden, werden als erstes in der Liste angezeigt. Die Bewertung der Hinweise („Anzahl als hilfreich markiert“ minus „Anzahl als nicht hilfreich markiert“) steht in Klammern direkt hinter jedem Hinweis. Abbildung 8.6 zeigt einige Hinweise zur Verfeinerung einer sicherheitskritischen Anforderung mit dem UMLsec-Stereotyp „secrecy“.

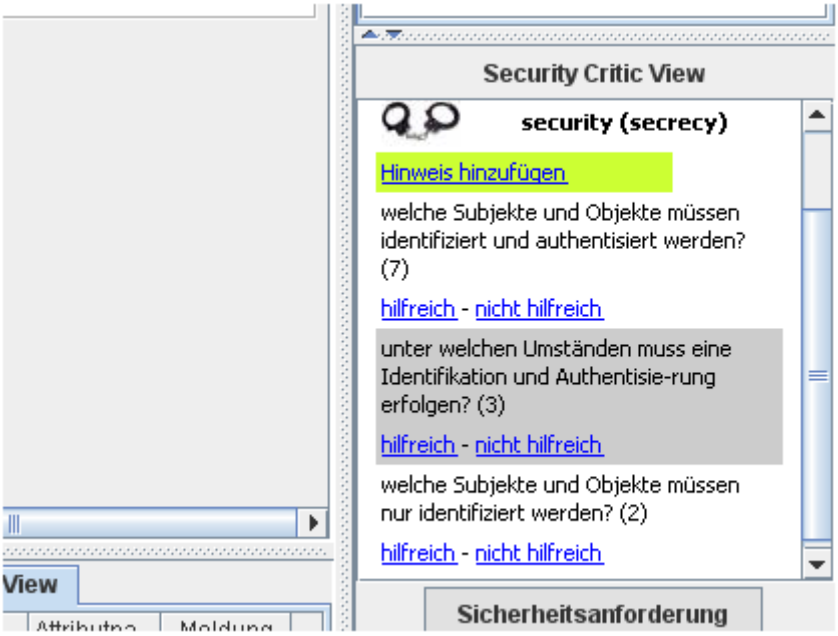


Abbildung 8.9: Hinweise zur Verfeinerung einer sicherheitskritischen Anforderung

Weiterhin besteht die Möglichkeit, neue Hinweise hinzuzufügen. Dafür muss auf den Link Hinweis hinzufügen angeklickt werden. Daraufhin erscheint folgendes Dialogfenster:

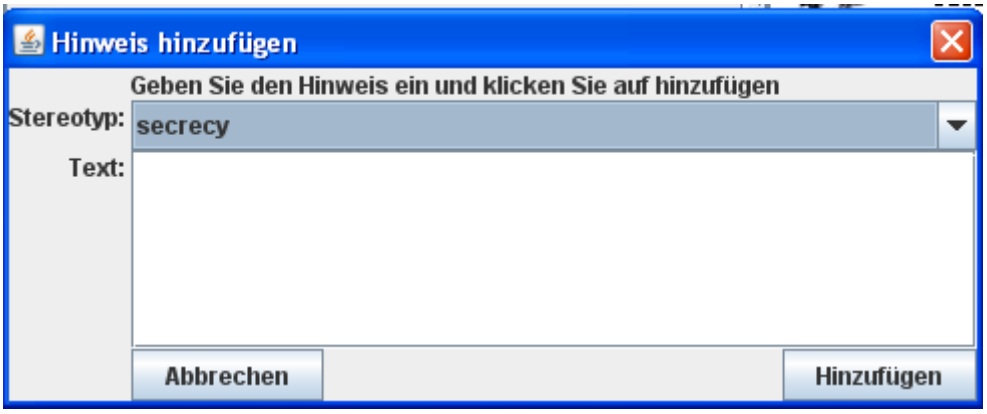


Abbildung 8.10: Hinweis Hinzufügen

Nach Eingabe des Hinweises muss auf den Button „Hinzufügen“ geklickt werden. Der Hinweis wird dem selektierten UMLsec-Stereotyp zugeordnet.

8.2.5 UMLsec-Diagramm erstellen

Wie in Abschnitt 7.2 erläutert, werden die Anforderungsanalysten und Software-Designer bei der Modellierung der Sicherheitsanforderungen von HeRA unterstützt. Beim Anklicken auf den Link UMLsec-Diagramm erstellen in der „Security Critc View“ wird der Anwender im ersten Schritt gefordert, einen UMLsec-Stereotyp auszuwählen:

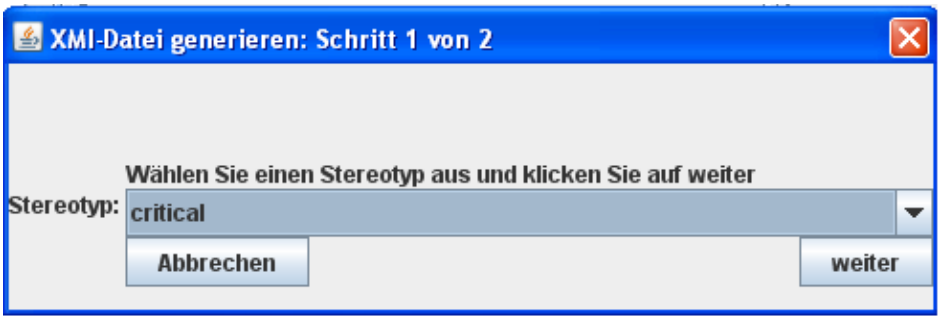


Abbildung 8.11: XMI-Datei generieren, Schritt 1 von 2

Im zweiten schritt werden abhängig vom ausgewählten UMLsec-Stereotyp die zugehörigen Eigenschaftswerte zum Ausfüllen angezeigt:

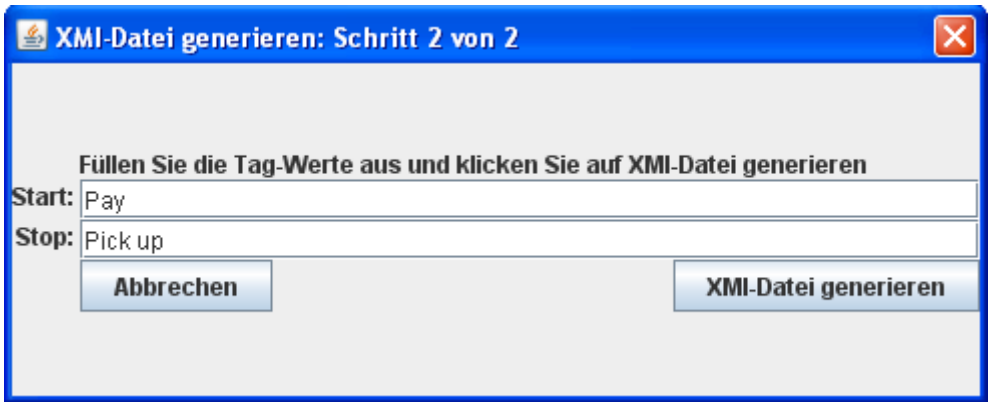


Abbildung 8.12: XMI-Datei generieren, Schritt 2 von 2

Beim Klicken auf den Button „XMI-Datei generieren“ wird eine XMI-Datei generiert und einen Hinweis über den Speichertort im Dateisystem angezeigt:

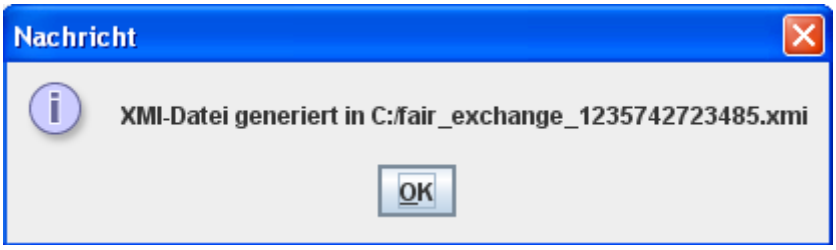


Abbildung 8.13: Speicherort der generierten XMI-Datei

Im Folgenden ist ein Ausschnitt aus der XMI-Datei, der den gesetzten Wert „Pay“ zeigt:

```
<XMI.content>
  <UML:Model xmi.id="a1" name="model 1" isSpecification="false" isRoot="fa
    <UML:ModelElement.taggedValue>
      <UML:TaggedValue xmi.id="a2" isSpecification="false">
        <UML:TaggedValue.dataValue>10--106--114-112-c23942:f3ca438e9f:-80(
        <UML:TaggedValue.type>
          <UML:TagDefinition xmi.idref="a3" />
        </UML:TaggedValue.type>
      </UML:TaggedValue>
      <UML:TaggedValue xmi.id="a4" isSpecification="false">
        <UML:TaggedValue.dataValue>Pay</UML:TaggedValue.dataValue>
        <UML:TaggedValue.type>
          <UML:TagDefinition xmi.idref="a5" />
        </UML:TaggedValue.type>
      </UML:TaggedValue>
```

Abbildung 8.14: Ausschnitt aus einer XMI-Datei

Der Software-Designer kann die generierte XMI-Datei mit einem geeigneten Editor (beispielsweise ArgoUML) öffnen:

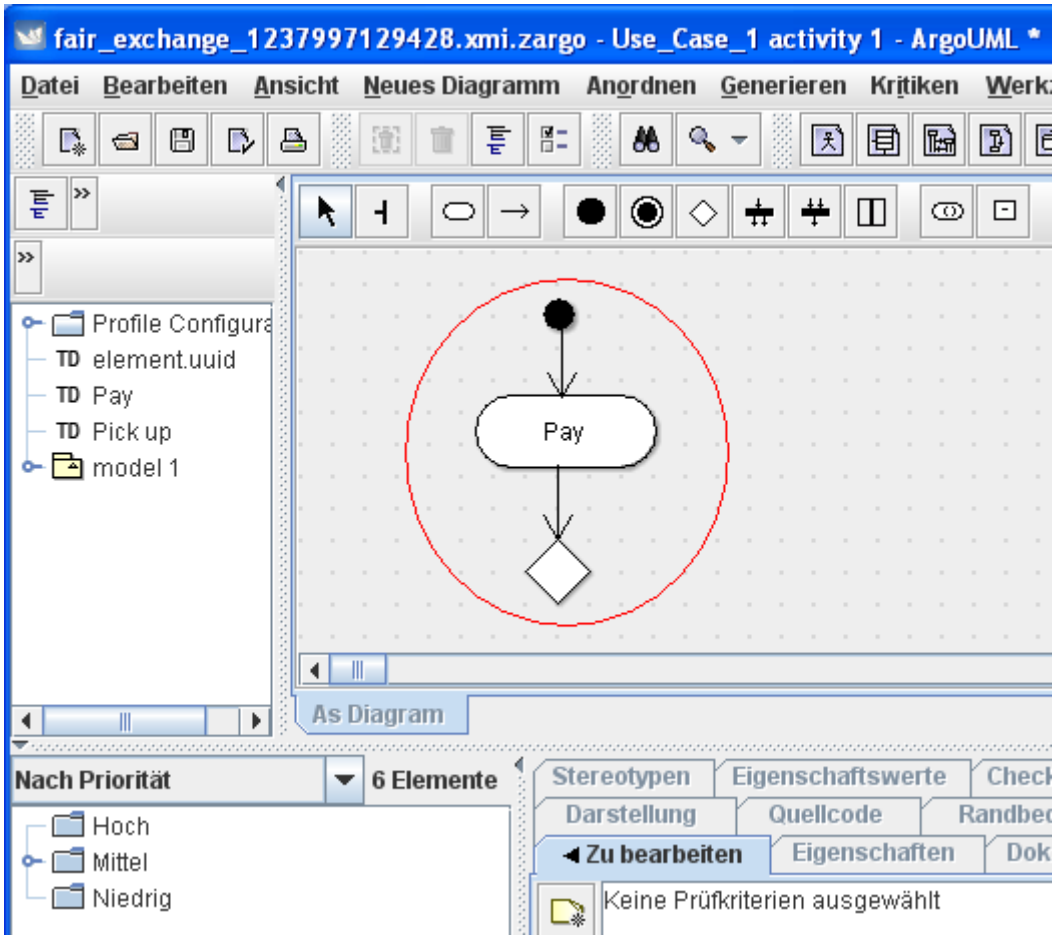


Abbildung 8.15: XMI-Datei im ArgoUML

Die Eigenschaftswerte (Start = Pay, Stop = Pick up) werden im ArgoUML nicht im Diagramm angezeigt. Das UML-Diagramm (der rot umkreiste Teil im obigen Bild) muss manuell von einem Software-Designer erstellt werden.

8.2.6 Datenbank erweitern

Weiterhin kann die Datenbank gleichzeitig mit mehreren Anforderungen erweitert werden. In diesem Fall muss der Button „Datenbank erweitern“ angeklickt werden.

Die Anforderungen müssen in einer .txt-Datei zeilenweise gespeichert sein. Zusätzlich muss am Ende jeder Zeile – getrennt durch ein Semikolon – angegeben werden, ob es sich um eine Sicherheitsanforderung handelt (sec) oder nicht (nonsec) und wenn ja von welchem UMLsec-Stereotyp. Im Folgenden ein Beispiel für eine Datei mit Sicherheitsanforderungen und Nicht-Sicherheitsanforderung:

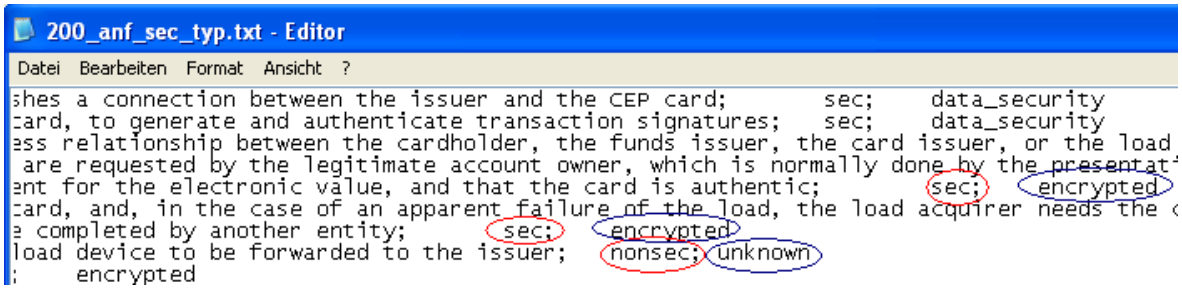


Abbildung 8.16: Anforderungen für die Datenbankerweiterung

8.3 Beispielszenario mit HeRA

Die Interaktionen zwischen dem Anforderungsanalyst und HeRA sowie die Aktionen des Software-Designers werden durch folgendes Beispielszenario zusammengefasst:

Nr.	Akteur	Aktion
1.	Anforderungsanalyst	Erfasst eine Anforderung in HeRA (in der Konstruktionskomponente)
2.	Anforderungsanalyst	Wählt „Critic View“ in HeRA (in der Argumentationskomponente)
3	HeRA	Analysiert die Anforderung mit Hilfe der Klassifizierungsheuristiken und zeigt an, ob es sich um eine sicherheitskritische Anforderung handelt (und das zugehörige UMLsec-Stereotyp): WENN nein, Ende. WENN ja, weiter mit Schritt 4
4	Anforderungsanalyst	Überprüft, ob die Klassifizierung von HeRA zutrifft: WENN nein: klickt auf „Als Nicht-Sicherheitsanforderung bestätigen“, Ende. WENN ja: klickt auf „Als Sicherheitsanforderung bestätigen“, weiter mit Schritt 5
5	Anforderungsanalyst	Klickt auf „Abstraktionsebene bestimmen“
6	HeRA	Stellt dem Anforderungsanalysten Fragen: Anforderungsanalyst antwortet mit „nein“: nächste Frage Anforderungsanalyst antwortet mit „ja“: Abstraktionsebene festlegen
7	Anforderungsanalyst	Anforderung auf einer hohen Abstraktionsebene: Klickt „Verfeinerungshinweise anzeigen“, weiter mit Schritt 8. Sonst: Weiter mit Schritt 10
8	HeRA	Zeigt abhängig vom UMLsec-Stereotyp passende Verfeinerungshinweise
9	Anforderungsanalyst	Liest Verfeinerungshinweise und erfasst neue detaillierte Anforderungen in HeRA
9a	Anforderungsanalyst	(optional) Bewertet die Verfeinerungshinweise als „hilfreich“ oder „nicht hilfreich“
9b	Anforderungsanalyst	(optional) Klickt auf den Link „Hinweis hinzufügen“ und verfasst einen neuen Verfeinerungshinweis
10	Anforderungsanalyst	Klickt auf „UMLsec-Diagramm erstellen“
11	HeRA	Zeigt dem Anforderungsanalysten Dialogfenster zur Eingabe von Eigenschaftswerten, abhängig von UMLsec-Stereotyp

12	Anforderungsanalyst	Gibt Werte ein und klickt auf „XMI-Datei generieren“
13	HeRA	Generiert eine XMI-Datei (UMLsec-Diagramm mit Eigenschafswerten)
14	Software-Designer	Öffnet XMI-Datei mit einem geeigneten UML-Editor und vervollständigt das UMLsec-Diagramm manuell
15	Software-Designer	Überprüft die Richtigkeit mit dem UMLsec-Tool

Tabelle 8.1: Beispielsszenario mit HeRA

Weiterhin kann die Begriffsdatenbank zu einem beliebigen Zeitpunkt mit zuvor klassifizierten Anforderungen erweitert werden (siehe Abschnitt 8.2.6).

9. Verwandte Arbeiten

In diesem Kapitel wird ein Blick auf andere Arbeiten geworfen, die ähnliche Verfahren wie dieser Arbeit behandelt haben.

9.1 Stemming-Verfahren

Im Abschnitt 4.3 wurde der Porter-Stemmer für das Zurückführen der Wörter auf ihren Wortstamm verwendet. Ein Problem des Porter-Stemmers ist, dass beim Stemming möglicherweise die unterschiedliche Semantik zweier Begriffe nicht erhalten bleibt. Dieses Problem wird als „overstemming“ bezeichnet. In [31] wurde ein Stemming-Verfahren vorgestellt, dass das Overstemming-Problem berücksichtigt, sodass beim Stemming die semantische Bedeutung nicht verloren geht. Dieses Stemming-Verfahren wurde in dieser Arbeit nicht weiter verfolgt, weil der Fokus auf den dynamischen Klassifizierungsverfahren gelegt wurde (siehe Abschnitt 5.4: Vorteile der dynamischen Klassifizierungsverfahren).

9.2 Ähnlichkeitsbestimmung

In [15] wurde das Vektormodel (Vector Space Model) für die Bestimmung der Ähnlichkeit von zwei Dokumenten vorgestellt. Das Vektormodell ist ein Verfahren aus dem Bereich Information Retrieval zur Berechnung der Ähnlichkeit zwischen einer Suchanfrage und einem Dokument aus einer Dokumentenmenge.

Die Dokumente werden durch Indexterme repräsentiert. Die Indexterme sind die Wörter mit mittlerer Häufigkeit im Dokument. Wird eine Suchanfrage gestellt, so wird ein Dokumentenvektor (document vector) d erstellt, der die Häufigkeiten der Indexterme in der Suchanfrage enthält. Die Indexterme der Suchanfrage werden ebenfalls als Vektor dargestellt (query vector) q und die Ähnlichkeit (similarity) sim zwischen den beiden Vektoren berechnet:

$$\mathrm{sim}(d,q) = \frac{d_1 \cdot q_1 + \ldots + d_n \cdot q_n}{\sqrt{d_1^2 + \ldots + d_n^2} \cdot \sqrt{q_1^2 + \ldots + q_n^2}}$$

Die Formel leitet sich aus dem Kosinus des Winkels zwischen den beiden Vektoren her. Je kleiner der Winkel, desto ähnlicher sind die Vektoren.

Das Problem bei der Berechnung der Ähnlichkeit zwischen Vektoren mit dieser Formel ist, dass die Länge der Vektoren unberücksichtigt bleibt.

Demzufolge müssen in dieser Arbeit die Sicherheits- bzw. Nicht-Sicherheitsanforderungen als zwei Vektoren aufgefasst werden. Diese Vektoren sind um ein vielfaches länger als die Vektorlänge einer einzigen zu klassifizierenden Anforderung. Aus diesem Grund stellt sich heraus, dass das Vektormodell für diese Arbeit ungeeignet ist.

In [15] wird eine Lösung vorgeschlagen, die die Länge der Dokumente berücksichtigt. Diese Lösung basiert auf das TFxIDF-Verfahren, das in Abschnitt 5.3 bereits vorgestellt wurde. Deshalb wurde die Klassifizierung mit Hilfe des Vektormodells nicht weiter verfolgt.

10. Zusammenfassung und Ausblick

In diesem Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst und auf die Erweiterungsmöglichkeiten in zukünftigen Arbeiten eingegangen.

10.1 Aufdeckung von sicherheitskritischen Anforderungen

In dieser Arbeit wurden Heuristiken zum Aufdecken von sicherheitskritischen Anforderungen entwickelt. Es hat sich gezeigt, dass die statischen Klassifizierungsheuristiken, die auf die Suche nach bestimmten Schlüsselwörtern und Synonymen basieren, sehr aufwendige manuelle Vorbereitung benötigen. Es müssen zunächst eine Vielzahl von Anforderungsspezifikationen auf Sicherheitsanforderungen analysiert werden um ausreichend viele relevante Begriffe zu definieren. Da die Formulierung von Anforderungen, und von natürlichsprachigen Texten im Allgemeinen, keine Grenzen gesetzt sind, ist ein Ansatz mit vordefinierter Menge von Schlüsselwörtern sehr pragmatisch.

Aus diesem Grund wurden weitere Verfahren aus dem Bereich Information Retrieval und Data Mining zur Bestimmung der Ähnlichkeit zwischen eine Menge von Dokumenten untersucht.

In dieser Arbeit als dynamische Klassifizierungsverfahren bezeichnet, wurden konkret der Bayes-Filter und das TFxIDF-Verfahren untersucht und angewendet. Diese Verfahren haben bessere Ergebnisse als die statischen Verfahren geliefert, ohne den Aufwand manuell feste Schlüsselwörter zu definieren. Auffällig war, dass sowohl der Bayes-Filter als auch das TFxIDF-Verfahren sehr ähnliche Ergebnisse erzielt haben. Dies lässt sich dadurch erklären, dass beide Verfahren auf dam gleichen Prinzip basieren, nämlich die Wahrscheinlichkeitsberechnung auf Basis von Worthäufigkeiten.

Ein weiterer Vorteil der dynamischen Verfahren ist, dass die Begriffsdatenbank, auf die sie basieren, leicht zu erweitern ist. Die Erweiterung hat jedoch nicht so viel zur Verbesserung der Ergebnisse beigetragen wie vorher angenommen und erwartet worden ist. Dies könnte daran gelegen haben, dass die Anforderungen, mit denen die Datenbank erweitert wurde, aus der gleichen Domäne stammen und dadurch kein zusätzliches Wissen gewonnen wurde.

Ein Problem ist, manuell eine möglichst große Menge von Sicherheitsanforderungen als Basis für die Berechnung zu identifizieren. Diese Identifizierung sollte idealerweise von einem Sicherheitsexperten vorgenommen werden.

Ein weiteres Ziel war, die aufgedeckten sicherheitskritischen Anforderungen einem UMLsec-Stereotyp zuzuweisen. Die statischen und dynamischen Klassifizierungsheuristiken können einen UMLsec-Stereotyp vorschlagen. Es konnten jedoch auf Grund von fehlenden Testdaten keine sinnvollen Bewertungen gemacht werden.

Ausblick:

In zukünftigen Arbeiten könnten Kombinationen aus statischen und dynamischen Klassifizierungsheuristiken untersucht werden. Beispielsweise können schlüsselwortbasierte Heuristiken so erweitert werden, dass die Menge der Schlüsselwörter durch den Benutzer zur Laufzeit angepasst werden kann. Eine weitere Idee ist, in auf Wahrscheinlichkeitsrechnung basierte Verfahren bestimmte Schlüsselwörter stärker zu gewichten. Zusätzlich könnten die dynamischen Klassifizierungsverfahren auch Synonyme berücksichtigen.

10.2 Verfeinerung von sicherheitskritischen Anforderungen

Ein weiteres Ziel dieser Arbeit war, den Anforderungsanalysten bei der Verfeinerung der sicherheitskritischen Anforderungen zu unterstützen, falls diese auf einer hohen Abstraktionsebene liegen. Für die Bestimmung der Abstraktionsebene hat sich das Requirements Abstraction Model (RAM) hervorragend geeignet, da es auf Basis einfacher Fragen schnell einer Anforderung eine Abstraktionsebene zuordnet. Das große Problem war, den Anforderungsanalysten sinnvolle Hinweise anzubieten, die zur Erhebung von weiteren Informationen beitragen. Die Verfeinerungshinweise sollten auf den UMLsec-Stereotyp der Anforderung zugeschnitten sein.

Die erste Idee war, solche Hinweise aus dem Common Criteria abzuleiten. Da die Richtlinien im Common Criteria sehr umfangreich und nicht explizit spezifiziert sind, hat diese Idee zu keinem Erfolg geführt. Alternativ wurde auf die Deutschen IT-Sicherheitskriterien zurückgegriffen. Sie bieten einige explizite Fragen, welche Daten erhoben werden sollten, um eine bestimmte Sicherheitsfunktion zu realisieren. Diese Fragen wurden als Hinweise für die Verfeinerung von sicherheitskritischen Anforderungen für bestimmte UMLsec-Stereotypen verwendet. Diese Hinweise decken jedoch nicht alle in Tabelle 2.1 vorgeführten UMLsec-Stereotypen auf. Deshalb wurde bei der Implementierung in HeRA den Nutzern die Möglichkeit angeboten, weitere Hinweise eintragen zu können, sodass sich diese Wissensbasis im Laufe der Zeit erweitert.

Es ist darauf aufmerksam zu machen, dass für ein UMLsec-Stereotyp die gleichen Verfeinerungshinweise angezeigt werden, unabhängig davon, auf welcher Abstraktionsebene die zu verfeinernde Anforderung liegt.

Ausblick:

In weiteren Arbeiten könnten Heuristiken entwickelt werden, die Verfeinerungshinweise bezogen auf einer bestimmten Abstraktionsebene vorschlagen.

10.3 Modellierung von Sicherheitsanforderungen

Das nächste Ziel dieser Arbeit war, eine Unterstützung bei der Erstellung von UMLsec-Diagrammen anzubieten. Da zur Modellierung einer Anforderung eine Vielzahl an Möglichkeiten existiert, hat sich die automatische Generierung von UMLsec-Diagrammen in dieser Arbeit darauf beschränkt, Diagramme mit UMLsec-Tags-Informationen zu erstellen. Die weitere Modellierung muss vom Software-Designer manuell vorgenommen werden.

Ausblick:

In weiteren Arbeiten könnte versucht werden, die aus der Anforderungsverfeinerung gewonnen Informationen automatisch in die UMLsec-Modellierung einzubinden.

Weiterhin könnten bei der Modellierung Security Patterns [32] beachtet werden. Security Patterns beschreiben bewährte Sicherheitslösungen und weisen eine Struktur auf, die mit herkömmlichen (Design) Patterns vergleichbar ist.

10.4 Informationsflüsse im Sicherheitsentwicklungsprozess

In dieser Arbeit wurde ein Prozess definiert, um die Aktivitäten bei der Aufdeckung, Verfeinerung und Modellierung von Sicherheitsanforderungen und deren Zusammenhang darzustellen. Der Prozess legt fest, wer an welcher Stelle mit welchem Tool welche Aktivitäten durchführen soll. Um dieser Prozess zu veranschaulichen, wurde FLOW verwendet, weil es eine einfache und überschaubare Notation zur Modellierung von Informationsflüssen anbietet.

Literaturverzeichnis

[1] R. Richardson. 2003 CSI/FBI computer crime and security survey. Technical report, Computer Security Institute, San Francisco, Mai 2003.

[2] D. Damian, L. Izquierdo, J. Singer, and I. Kwan. Awareness in the Wild: Why Communication Breakdowns Occur. In Proceedings of Second Intl Conference on Global Software Engineering, pages 81–90, Munich, Germany, 2007.

[3] P. Ryan, S. Scheinder, M. Goldsmith, G. Lowe and B. Roscoe. The Modelling and Analysis of Security Protocols: the CSP Approach. Addison-Wesley, 2001.

[4] Knauss, E. (2007), 'Einsatz computergestützter Kritiken für Anforderungen', GI Softwaretechnik-Trends 27(1).

[5] Knauss, E. & Flohr, T. (2007), Managing Requirement Engineering Processes by Adapted Quality Gateways and critique-based RE-Tools, in 'Proceedings of Workshop on Measuring Requirements for Project and Product Success'.

[6] Schneider, K. (2008), Improving Feedback on Requirements through Heuristics, in 'Proceedings of 4th World Congress for Software Quality (4WCSQ)'.

[7] E. Knauss, D. Lübke, and S. Meyer, "Feedback-Driven Requirements Engineering: The Heuristic Requirements Assistant", Formal Research Demonstration at 31st IEEE International Conference on Software Engineering, Vancouver, Canada, 2009.

[8] K., Requirements Engineering, Grundlagen, Prinzipien, Techniken. Dpunkt.verlag 2008.

[9] http://www.softwarekompetenz.de

[10] Schäling B., Der moderne Softwareentwicklungsprozess mit UML, http://www.highscore.de/uml/, 2005

[11] Harald S., UML 2 für Studenten, Pearson Studium, 2005.

[12] Jan Jürgens, Secure Systems Development with UML, Springer Verlag, 2004.

[13] Russell, S.J., and Norvig, P., Artificial Intelligence a Modern Approach. Prentice Hall, Upper Saddle River, New Jersey, 1995.

[14] Baeza-Yates R., Ribeiro-Neto B, Modern Information Retrieval, New York, ACM Press, Addison-Wesley, 1999.

[15] http://kontext.fraunhofer.de/haenelt/kurs/Referate/SchmidtStefan_WS02/vektormodell.pdf

[16] http://mcs.open.ac.uk/jj2924/umlsectool/Applications/ceps/specification/

[17] Christian Gütl et al. Dynamic Background Libraries as an improved Way for Web-Based Learning using HIKS, Proceedings of ICCE98, 1998.

[18] M.F. Porter: An algorithm for suffix stripping, Juli 1980.

[19] Russel S., Norvig R. Künstliche Intelligenz, ein moderner Ansatz. 2. Auflage, Person Studium 2004.

[20] Ending spam: Bayesian content filtering and the art of statistical language classification / by Jonathan A. Zdziarski, San Francisco : No Starch Press, 2005.

[21] Stefania Costache et al., Detecting Contexts on the Desktop Using Bayesian Networks, L3S Research Center, Leibniz Universität Hannover.

[22] Konrad H., Dynamischer Keyword-Relevanzfilter und Ähnlichkeitsmaß von Dokumentenmittels Keywords, TU Graz.

[23] G. Fischer. Domain-Oriented Design Environments. Automated Software Engineering, 1:177–203, 1994.

[24] M. Sahami, S. Dumais, D. Heckerman, E. Horvitz: A Bayesian approach to filtering junk e-mail, AAAI'98 Workshop on Learning for Text Categorization, 1998.

[25] Kitzmann I., Konzept und Werkzeug zur erfahrungsbasierten Erstellung von Use Cases, Bachelorrarbeit, Leibniz Universität Hannover, 2006.

[26] http://www.se.uni-hannover.de/forschung/flow/

[27] D. T. VERLAG (Herausgeber): dtv Lexikon Band 16. Deutscher Taschenbuch Verlag, 1990.

[28] Englbrecht, Michael, Entwicklung sicherer Software, Spektrum, 2004.

[29] Gorschek T., Wohlin C., „Requirements Abstraction Model“, Requirements Eng (2006) 11: 79-101

[30] Kotonya G., Sommerville, „Requirements Engineering: processes and techniques.“, Wisley, New York, 1998.

[31] Meyer S., Halbautomatische Generierung eines Glossars während der Dokumentation von Anforderungen, Leibniz Universität Hannover, 2007.

[32] Schumacher M., Security Patterns: Integrating Security and Systems Engineering, 2006

Anhang A: Schlüsselwörter und die dazugehörigen Synonyme

UMLsec-Stereotyp	Schlüsselwort	Synonyme
secrecy	access	call | execute | fetch | query | read | request | retrieve | select
	account	ledger
	add	annex | append | apply | attach | copy | create | enclose | include | infix | insert | join | paste | put | write
	authorise	accord | allow | appropriate | approve | concede | empower | grant | license | permit
	data	component | content | details| document | file | page | paper | information | site | statement | text | website
	display	advertise | announce | demonstrate | globalize | look | overview | present | record | report | see | show | view | visualize
	download	import
	identify	audit | certify | check | control | descry | detect | discover | expose | inspect | probe | prove | review | uncover | verify
	logon	login | log-in | log in | log-on | log on
	password	keyword | codeword | PIN
	publish	headline | issue | publicise | release
	register	earn | enrol | enter
	user	actor | administrator | author | cardholder | customer | editor | handler | operator | program | somebody | someone | stakeholder | student | system | worker
integrity	add	siehe secrecy
	access	siehe secrecy
	allow	accord | afford | bestow | concede | confer | grant
	block	close | disable | lock
	data	siehe secrecy
	delete	move | remove
	display	siehe secrecy
	downlaod	siehe secrecy
	edit	change | modify | process | replicate | update
	manage	administrate | configure
	save	memorise | record | store
	upload	export
	user	siehe secrecy
high	account number	acct. No.
	bank account	checking account
	credit card	bank card | charge card | master card | plastic | smart card | visa card
	decrypt	decipher | decode | unscramble
	encrypt	cipher | code | encipher | encode | scramble

	meta data	metadata
	password	siehe secrecy
	personal	private
fair exchange	buy	appoint | aquire | disburse | gain | get | pay | purchase | request | shop
	business	company | concern | enterprise | shop | store | trade
	deliver	consign | dispatch | forward | mail | post | provide | purvey | send | serve | ship | supply | surrender
	good	artifact | article | item | material | offer | product | service | wares
	internet	net | online | web
	lend	borrow | loan | rent
	reserve	book | order
	sell	market | merchandise | retail
	user	acquirer | actor | buyer | client | consumer | customer | handler | operator | purchaser | shopper | somebody | someone | stakeholder | worker