Electronic version of an article published in **Haasis, K.; Heinzl, A.; Klumpp, D. (Hrsg): Aktuelle Trends in der Softwareforschung. Tagungsband zum doIT Softwareforschungstag 2007 in Mannheim**

Ein Beitrag zur Berücksichtigung von Compliance in der Softwareentwicklung – Die SIKOSA-Methodik

Dipl.-Kfm. techn. Daniel Weiß Universität Hohenheim
Prof. Dr. Stefan Kirn Universität Hohenheim
Prof. Dr. Günter Müller Universität Freiburg
Dipl.-Inf. Maike Gilliot Universität Freiburg
Dipl.-Inf. (FH) Lutz Lowis Universität Freiburg
Prof. Dr. Barbara Paech Universität Heidelberg
Dr. Andrea Herrmann Universität Heidelberg
MSc Carsten Binnig Universität Heidelberg
Dipl.-Inf. Timea Illes-Seifert Universität Heidelberg
Prof. Dr. Donald Kossmann ETH Zürich

Zusammenfassung

Das Ziel Compliance-konformer Unternehmenssoftware (USW) ist heutzutage von zentraler Bedeutung. Dieses wird jedoch vielfach nicht erreicht, wofür ursächlich zahlreiche Methodenbrüche entlang des Softwareentwicklungsprozesses mit verantwortlich gemacht werden. Vor diesem Hintergrund leistet die SIKOSA-Methodik zur integrierten Spezifikation von Geschäftsprozess- und testbaren USW-Anforderungen einen Beitrag zur Erstellung bzw. Weiterentwicklung valider USW.

1 Motivation

Gesetzliche nationale und internationale Vorschriften, Richtlinien oder auch unternehmensinterne Vorgaben stellen immer neue Anforderungen an die Transparenz von Unternehmenssoftware (USW). Dies gilt so zum Beispiel für Sarbanes-Oxley (SOX), Basel II, KonTragG oder Solvency II. (IT-)Entscheider stehen damit vor der Herausforderung sicherzustellen zu müssen, dass die zur Geschäftsprozessunterstützung neu zu beschaffende oder sich bereits im Einsatz befindende USW nachweislich, also testbar, diesen Auflagen entspricht (engl. compliance). Eine Nicht-Einhaltung kann nicht nur empfindliche Strafen seitens der Kapitalgeber nach sich ziehen, sondern im Fall von Basel II (Internationale Konvergenz der Eigenkapitalmessung und der Eigenkapitalanforderungen) beispielsweise auch den Zugang zu Fremdkapital versperren [Kno07; Can06].

Die Auflagen, oder besser Anforderungen, können damit nicht ohne Konsequenzen für den Softwareentwicklungsprozess als Gegenstand bleiben. Obwohl die Softwareentwicklung in allen ihren Phasen über ein umfangreiches Methodenrepertoire verfügt, stellen aus der Perspektive der Automatisierbarkeit des Softwaretest insbesondere Methodenbrüche nach wie vor ein drängendes, zu lösendes Problem dar. Die SIKOSA-Methodik adressiert daher die Phasen der Softwareentwicklung, die an der Schnittstelle zwischen Nutzern und Erstellern der USW liegen. Sie legt damit das Hauptaugenmerk

auf die Phasen der Anforderungsanalyse und -spezifikation sowie des Tests, die der Anforderungsphase ablauflogisch zwar erst deutlich später nachfolgt, funktional jedoch unmittelbar auf ihr basiert, da Anforderungen den Maßstab für den späteren Test darstellen. Die SIKOSA-Methodik richtet sich dabei an den zwei Säulen der Geschäftsprozessqualität und Sicherheitsanforderungen aus entlang derer die Methodenbrüche bei den Phasenübergänge reduziert werden (vgl. Abb.1). SIKOSA stellt folgende Methoden bereit:

- *ProQAM:* Methode zur qualitätsorientierten Analyse und Spezifikation von Geschäftsprozessen (Universität Hohenheim).
- *TORE:* Methode zur Analyse und Spezifikation funktionaler Anforderungen (Universität Heidelberg).
- *MOQARE:* Methode zur Erhebung und Priorisierung von nicht-funktionalen Anforderungen (synonym Qualitätsanforderungen) integriert mit funktionalen Anforderungen (Universität Heidelberg).
- *PAT3:* Mustersammlung zur Ableitung von Testspezifikationen aus Anforderungen (Anwendungsfälle) (Universität Heidelberg).
- *RQP:* Technologie zur Erzeugung betriebswirtschaftlich relevanter / sinnvoller Testdaten (Universität Zürich).
- *VEP-Sicherheitsmonitor:* Werkzeug zum Sicherheitsmonitoring von USW (Universität Freiburg).

Das SIKOSA-Metamodell, Details zu den verwendeten Methoden sowie das sich über diese Methoden erstreckende Vorgehensmodell findet sich in [Wei07; Kir06].

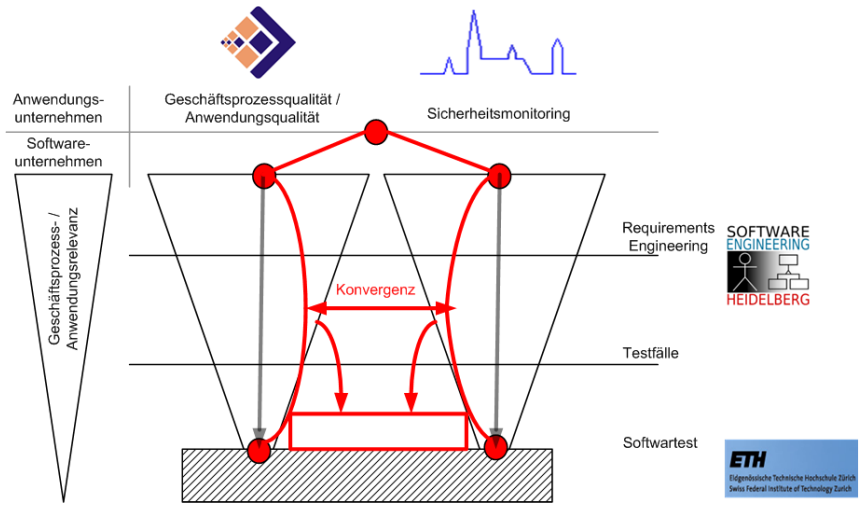

Abb. 1.: Lösungsansatz SIKOSA-Methodik

Durch eine Metamodell-basierte Integration der Einzelmethoden und dadurch erzielte Konvergenz werden folgende Beiträge geleistet:

1. Verbesserte Validierungsmöglichkeit der Compliance-Anforderungen aufgrund einer durchgängigen Geschäftsprozess- und USW-Anforderungsanalyse.
2. Verbesserte Testbarkeit bei Software-Neuentwicklungen und Änderungen sich bereits im Einsatz befindender Software unter Berücksichtigung der Geschäftsprozesssemantik, und damit auch eine verbesserte Automatisierbarkeit des Softwaretests.
3. Überprüfbare Durchsetzung von spezifizierten Sicherheitsanforderungen .

Der Lösungsansatz bringt damit nicht nur die Automatisierbarkeit des Softwaretests substantiell voran, sondern erhöht zugleich auch die Qualität und Wirtschaftlichkeit der gesamten Softwareentwicklung, da Missverständnisse, Auslassungen und Widersprüche minimiert werden und somit die mögliche Fehleranzahl, die sich bis zur Implementierung fortpflanzen könnte, gering hält.

Der Beitrag ist wie folgt gegliedert: Kapitel zwei diskutiert verwandte Arbeiten. In Kapitel drei werden ProQAM, TORE und MOQARE als Methoden zur Lösung der identifizierten Problemstellung vorgestellt. In Kapitel vier werden diese beispielhaft zur Anwendung gebracht. Kapitel fünf behandelt zwei ausgewählte Lösungsansätze zur Validierung von Sicherheitsanforderungen: Der (automatisierte) Systemtest überprüft die Einhaltung der funktionalen Anforderungen. Der Sicherheitsmonitor zielt auf die Erkennung von Sicherheitsverletzungen zur Laufzeit ab. Kapitel sechs fasst die Ergebnisse zusammen.

2 Verwandte Arbeiten

Die Entwicklung von USW verlangt nach einer integrierten Analyse der Geschäftsprozess- und USW-Anforderungen, differenziert nach funktionalen und nichtfunktionale Anforderungen. Jedoch liefert die Literatur zu diesem an und für sich nahe liegenden Vorgehen, USW-Anforderungen aus Geschäftsprozessanforderungen abzuleiten, nur wenige Lösungsvorschläge [Kor06a; Oes03; Brü01]. Als zu überwindende Hürden erweisen sich regelmäßig unter anderem unterschiedliche Notationen, Semantiken von Modellierungskonstrukten oder Qualitätsmodelle für Geschäftsprozesse und USW. Insbesondere eine Semantik erhaltende Überführung von Geschäftsprozess - in IT-Qualitätsattribute erweist sich als problematisch. Soll ein Geschäftsprozess beispielsweise eine geforderte Wertschöpfung (als Beispiel eines Geschäftsprozess-Qualitätsattributes) erzielen, leiten sich daraus unter anderem Anforderungen bezüglich des Antwortzeitverhaltens an die USW oder auch Vorgaben für die akzeptierbaren Kosten (als Beispiele für IT-Qualitätsattribute) der USW-Lösung ab.

Zwar stellen [Kor06b; Nor04; Fer01; Eri00; Sch00; Dan99] Ansätze zur Integration der Geschäftsprozess- und USW-Anforderungsanalyse vor, jedoch verweisen neuere

Arbeiten [Kor06a; Brü01] auf weiter bestehende Schwächen bezüglich der Methodenintegration. Auch fokussieren diese Arbeiten primär funktionale Anforderungen; nicht-funktionale Anforderungen hingegen werden häufig nicht betrachtet, obwohl sie im Hinblick auf gestiegene Kundenansprüche für Softwareanbieter eine hohe Relevanz besitzen. Diese ergibt sich aus der Tatsache, dass hoch-standardisierte USW zunehmend austauschbar wird. Der Kundenwettbewerb kann damit nicht mehr allein durch den Funktionsumfang der Softwarelösung gewonnen werden kann, sondern vielmehr durch deren Qualität. Die SIKOSA-Methodik adressiert daher insbesondere nicht-funktionale Anforderungen.

Parallel dazu haben sich in den letzten Jahren automatisierte (Regressions-)Tests zur Qualitätssicherung bei Softwareänderungen weitgehend durchgesetzt [Gra05; Liu99]. Für datenbankbasierte USW stehen jedoch bisher keine vergleichbaren Lösungen zur Verfügung, obwohl der Großteil der USW gerade datenbankgestützt arbeitet.

Für den Test derartiger Systeme mussten bisher zuvor Testdatenbanken generiert werden. In dynamischen Anwendungen ist dies kaum realitätsnah möglich. Weiter sind auch methodische Probleme noch ungelöst, da oftmals lediglich das Datenbankschema adressiert wird [Cha04; Neu93] oder auf Basis einer statistischen Verteilung zufällige Daten generiert werden [Bru05; Gra94]. Letztere reichen jedoch nicht aus, um eine große Zahl kritischer Pfade der Anwendung abzudecken. In der Folge kann die Aussagekraft des Tests leiden und beeinträchtigt die Validität.

3 Integrierte Analyse und Spezifikation von Geschäftsprozess- und USW-Anforderungen

3.1 ProQAM – Process-oriented Questionnaire for Analyzing and Modeling Scenarios

ProQAM ist eine Fragebogen-gestützte Methode zur funktionalen und nicht-funktionalen Analyse und Spezifikation von Geschäftsprozessen. Sie basiert auf Metamodellen der Aufbauorganisation, Ablauforganisation und Geschäftsprozessqualität. Zur Beschreibung der Ablauforganisation dient das Metamodell der eEPK in Anlehnung an [Sch02], ergänzt um Rollen, Stellen und Stellentypen [Bie06].

Ähnlich zu den Metamodellen zur Aufbau- und Ablauforganisation basiert das Metamodell zur Beschreibung der Geschäftsprozessqualität ebenfalls auf Funktionen und Geschäftszielen (Sach- und Formalzielen). Auf Basis dreier Klassen von Qualitätsattributen - strukturellen, formalen und inhaltlichen Attributen - werden nicht-funktionale Anforderungen an die einzelne Funktion bzw. den gesamten Prozess formuliert. Die Auswahl des Qualitätsattributes bzw. die Formulierung der nicht-funktionalen Anforderung ist unmittelbar durch die Geschäftsziele beeinflusst. Die Nichterfüllung einer nicht-funktionalen Anforderung führt zu einem Qualitätsmangel der Funktion bzw. des Geschäftsprozesses und birgt damit ein Risiko für das Unternehmen in sich (Unternehmensrisiko). Das Prozessrisiko wird grundsätzlich aus

dem potenziellen Unternehmensschaden und dessen Eintrittswahrscheinlichkeit ermittelt.

Während strukturelle Qualitätsattribute wie zum Beispiel die Komplexität oder der Modularisierungsgrad sich lediglich mittelbar auf die in den Geschäftsprozessen verwendete USW auswirken, haben die formalen und inhaltlichen Qualitätsattribute unmittelbaren Einfluss auf die Auswahl der Qualitätsattribute der im Geschäftsprozess verwendeten USW. Die formalen und inhaltlichen Qualitätsattribute leiten sich aus dem Zweck (z.B. Entwicklung, Beschaffung, Produktion) der Geschäftsprozesse ab und adressieren unter anderem deren Effizienz, Flexibilität oder Integrität. Letzteres beschreibt das Verhältnis aus exklusiv genutzten Informationsobjekten und -trägern in einem Geschäftsprozess zur Gesamtzahl an Informationsobjekten und -trägern. Damit besitzt sie unmittelbaren Einfluss auf die Integrität und Sicherheit der verwendeten USW, die sich bis in den automatisierten Test der Unternehmenssoftware hinein durchschlägt.

3.2 TORE – Task Oriented Requirements Engineering

TORE als Methode des Requirements Engineering leitet funktionale Anforderungen aus Geschäftsprozessen ab [Pae03] und kennt vier Ebenen der Analyse. Durch Diskussion mit den Interessengruppen auf der Interaktionsebene leitet TORE die Anforderungen an die Benutzungsschnittstelle aus den ProQAM-Spezifikationen, d.h. insbesondere den Aufgaben der Nutzer und den Systemverantwortlichkeiten ab. Auf der Systemebene werden daraus funktionale Anforderungen an den Anwendungskern und die grafische Benutzeroberfläche spezifiziert. Das zentrale Konzept der Schnittstelle zur Analyse der nicht-funktionalen Anforderungen und zur Herleitung der Testfallspezifikationen stellen Anwendungsfälle dar.

3.3 MOQARE – Misuse Oriented Quality Requirements Engineering

MOQARE dient zur Konkretisierung von Qualitätszielen sowie zur Herleitung realisierbarer und testbarer nicht-funktionaler Anforderungen bzw. Unterstützung der Qualitätsziele [Her06, Her05].

Die MOQARE-Analyse beginnt mit den Geschäftszielen, die durch einen Geschäftsschaden bedroht werden können. Dieser beschreibt qualitativ den Schaden, während das Unternehmensrisiko ihn quantifiziert. Der Geschäftsschaden wird wiederum durch einen Qualitätsmangel der USW verursacht. MOQARE betrachtet nicht-funktionale Anforderungen in Form von Qualitätszielen, die jeweils eine Kombination aus einem Wert und seinem IT-Qualitätsattribut darstellen. Ein Wert ist dabei jedes zu schützende Teil der USW und beinhaltet alle durch ProQAM erfassten Elemente. Erfüllt der Wert nicht das IT-Qualitätsattribut, wird von einem Qualitäts-mangel gesprochen. Zu beachten ist, dass die Qualitätsmängel und -ziele (auf USW bezogene) IT-Qualitätsattribute verwenden, wie sie durch die ISO 9126 beschrieben werden. Eine Bedrohung ist eine Aktion, die das Qualitätsziel bedroht und zugleich die

Ursache des Qualitätsmangels darstellt. Verursachende Fehlnutzer können sein: Personen (Hacker, Benutzer, Administrator, usw.) andere Systeme oder Naturgewalten.

Sicherheit ist dabei ein Spezialfall, bei dem der Fehlnutzer entweder (1) ein Eindringling oder (2) ein Benutzer ist, der eine für ihn nicht vorgesehene Funktion ausführt und dabei ein schädliches Ziel verfolgt. Ebenso kann (3) ein nachlässiger Benutzer die Sicherheitsregeln verletzen. Alle anderen IT-Qualitätsattribute der ISO 9126 werden durch (4) normale Systembenutzer oder -entwickler bedroht. Oftmals wird eine Bedrohung durch eine Schwachstelle erleichtert, ermöglicht oder gar provoziert.

Eine Fehlnutzung beschreibt dabei das gesamte Fehlnutzungsszenario inklusive Fehlnutzer, Schwachstelle, Bedrohung und dessen Folgen (Qualitätsmangel). Sie wird in Form und Granularität eines Fehlnutzungsanwendungsfalls dokumentiert. Um den Bedrohungen und Fehlnutzungen zu begegnen und die Qualitätsziele (bzw. die Geschäftsziele) zu schützen, definieren wir Gegenmaßnahmen, die meist neue Anforderungen darstellen. Gegenmaßnahmen können eine Fehlnutzung entdecken, verhindern oder abschwächen.

Solche Gegenmaßnahmen können unter anderem neue funktionale Anforderungen (z.B. Anwendungsfall), erweiterte oder neue Ausnahmeszenarien von Anwendungsfällen, Architekturanforderungen oder weitere nicht-funktionale Anforderungen sein.

4 Anwendungsbeispiel

Zur weiteren Erläuterung der SIKOSA-Methodik und Anwendung des entwickelten Metamodells verwenden wir das exemplarische Szenario eines idealtypischen industriellen Beschaffungsprozesses [Sch97].

Prozessanalyse – ProQAM

Die Prozessanalyse mit ProQAM beinhaltet die Analyse und Spezifikation der Geschäftsziele, -schäden und -prozesse, der nicht-funktionalen Anforderungen an diese Prozesse sowie die Struktur der Aufbauorganisation.

Ausgehend von den Geschäftszielen kann beispielsweise als eine *unternehmensinterne* zu erfüllende *Qualitätsanforderung* an einen Geschäftsprozess eine *Wertschöpfung durch den Einsatz von USW in Hohe von x €* spezifiziert werden. Die Wertschöpfung ergibt sich dabei aus weiteren Geschäftsunterzielen:

- *Durchschnittliche Zeitersparnis* bei der Geschäftsprozessausführung von > x Minuten/Geschäftsprozessinstanz
- *Durchschnittliche Kostenersparnis* durch
    - reduzierte Wartungskosten in Höhe von > 20.000 €/Monat
    - gesteigerte Datenqualität (Datenbereinigungsmaßnahmen) > 15.000 €/-Monat

Jedes Geschäftsziel wird durch mindestens einen Geschäftsschaden bedroht, so zum Beispiel durch gestiegene Kosten für Hardwareersatzteile oder Softwarelizenzen.

Analyse der funktionalen und nicht-funktionalen USW-Anforderungen

Das Ergebnis einer MOQARE-Analyse wird in einem Fehlnutzungsbaum dokumentiert. Dabei werden aber nur diejenigen Bedrohungen und Gegenmaßnahmen spezifiziert, welche für die konkrete USW-Lösung als besonders relevant erachtet werden (vgl. Abb. 2 und Abb. 3).

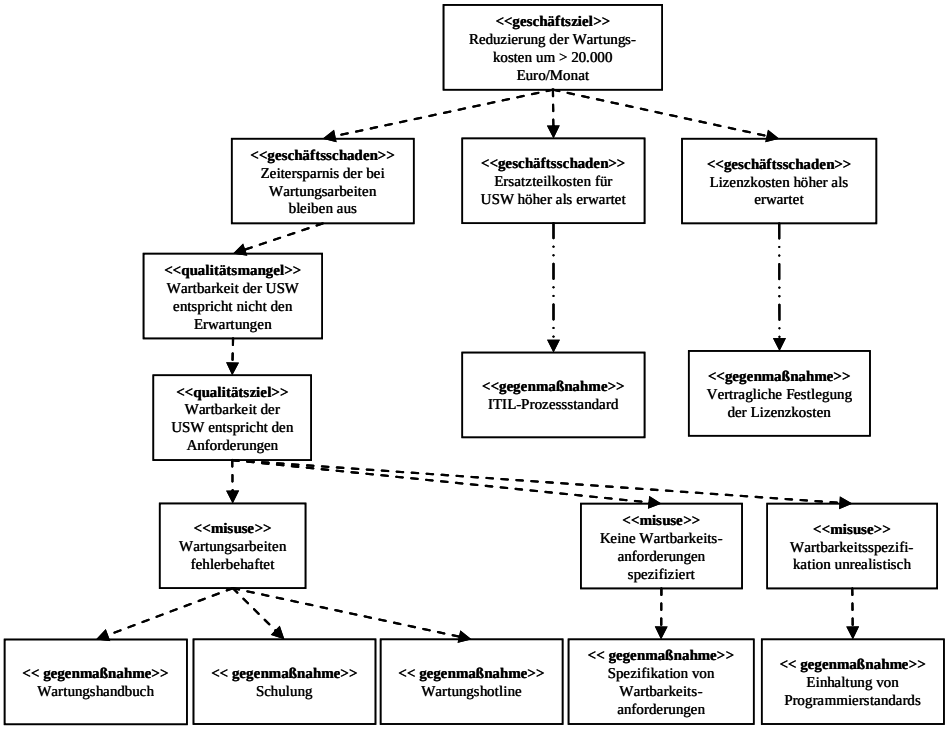

Abb. 2.: Fehlnutzungsbaum für das Geschäftsziel Reduzierung von Wartungskosten

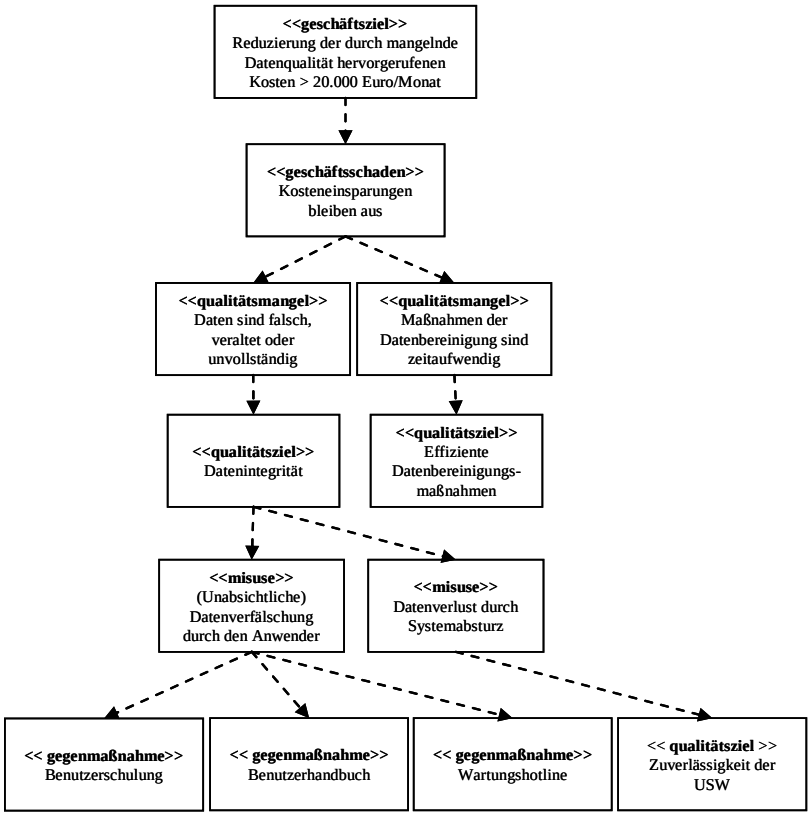

Abb. 3.: Fehlnutzungsbaum für das Geschäftsziel Reduzierung der durch mangelnde Datenqualität hervorgerufenen Kosten > 20.000 Euro / Monat

5 Sicherheitsmonitoring und Systemtest

Durch Testen und Monitoring lässt sich die Einhaltung der mit MOQARE und TORE erhobenen Anforderungen validieren. Der automatisierte Systemtest dient der Überprüfung der funktionalen Anforderungen, der Sicherheitsmonitor der Erkennung von Sicherheitsverletzungen zur Laufzeit der USW.

5.1 Sicherheitsmonitoring für USW

Softwarehersteller wie SAP, IBM, Microsoft oder Oracle sind bestrebt den Einsatz von auf serviceorientierten Architekturen (SOA) basierender USW zu forcieren. Ein noch ungelöstes Problem stellt dabei aber die Durchsetzung von Sicherheitsanforderungen und Konformitätsprüfung zu neuen Regularien dar. Dazu wird ein Monitor vorge-

schlagen, der unter Berücksichtigung von Schwachstellen die Einhaltung von Sicherheitsanforderungen zur Laufzeit erzwingt.

Sicherheitsfehler in dynamischen Anwendungssystemen

Vereinfachen SOA einerseits die flexible Einbindung von Kommunikationspartnern und Diensten (bspw. beim E-Procurement mit wechselnden Transaktionspartnern), führen sie aber anderseits zeitgleich zu Problemen bzgl. der Geheimhaltung (engl. privacy), ob der eingebundene Dienst mit den Daten und Rechten im Sinne der Richtlinie (engl. policy) umgeht [Woh06]. Derzeitige USW-Lösungen bieten noch keine ausreichenden Mechanismen für diese wiederholt notwendige Sicherheitsüberprüfung. Um dynamische, unternehmensübergreifende Kooperationen zu sichern, müssen die Sicherheitsbeziehungen flexibler gestaltet werden und die Sicherheitsüberprüfung effizient möglich sein. Sicherheitsschwachstellen lassen sich dabei nach dem Zeitpunkt, zu dem sie festgestellt werden, unterscheiden: Vor, während und nach der Abwicklung des Geschäftsprozesses. Der Fehlerzeitpunkt ist insofern von Bedeutung, als vom Zeitpunkt der Fehlerentdeckung die Reaktionsmöglichkeiten auf den Fehler abhängen.

- *Vor der Abwicklung:* Die Methoden der „Vorher-Sicherheit“ (z.B. formale Spezifikation, etc.) gehen davon aus, dass alle möglichen Verhaltensweisen der USW determiniert sind. Sicherheitsrelevante Fehler lassen sich zu diesem Zeitpunkt zum Beispiel mittels geänderten Zugriffsregeln oder zusätzlichen Sicherheitsmechanismen beheben.
- *Während der Abwicklung:* Für die Überprüfung von Sicherheitseigenschaften während der Abwicklung eignen sich Referenzmonitore oder um Sicherheitstypen ergänzte Programmiersprachen (z.B. C#). Dadurch werden Aktionen während ihrer Ausführung beobachtbar und beurteilungsfähig. Dabei werden viele Sicherheitsschwachstellen aber eben erst zur Laufzeit der Anwendung deutlich und machen damit eine erneute Überprüfung der Sicherheitsanforderungen notwendig. Jedoch existiert derzeit noch kein Mechanismus, der zur Laufzeit auftretende Sicherheitsfehler betrachtet und Reaktionsmöglichkeiten auf diese Sicherheitsfehler anbietet.
- *Nach der Abwicklung:* Durch Auditierung „geloggter“ Informationen kann nach der Ausführung des Prozesses überprüft und bewiesen werden, dass die Anforderungen bzgl. Sicherheit und Geheimhaltung eingehalten wurden [Sac06]. Wurden Regeln nicht eingehalten, sind Verfahren aus der Informatik nur noch begrenzt möglich.

Sichere Unternehmensinteraktion durch schwachstellensensitives Monitoring

Zum Umgang mit *während der Abwicklung* entdeckten Sicherheitsfehlern wird ein schwachstellensensitiver Sicherheitsmonitor „Vulnerability Evaluation Point“ (VEP) vorgeschlagen. Dieser betrachtet bei der Auswertung von Zugriffsanfragen nicht nur

die Zugriffskontrollregeln, sondern auch die Schwachstellen der USW. Der VEP-Monitor ist eine Ergänzung zur regelbasierten Zugriffskontrolle woraus sich ein zweistufiger Zugriffskontrollmechanismus ableitet:

Zunächst wird dafür eine eingehende Zugriffsanfrage gemäß der Zugriffskontrollregeln überprüft. Ist die Anfrage gemäß den Zugriffsregeln zulässig, überprüft der VEP-Monitor, ob der gewünschte Zugriff eine bekannte Schwachstelle ausnützen könnte. Dabei greift er auf eine Schwachstellendatenbank zurück, welche die bekannten Schwachstellen der Komponenten des Anwendungssystems enthält. Könnte der angefragte Zugriff eine Schwachstelle ausnutzen und unberechtigte Zugriffe oder Rechte zur Folge haben, lehnt der VEP-Monitor den Zugriff ab, obwohl die Anfrage gemäß den Zugriffsregeln zulässig ist. Somit lassen sich sicherheitskritische Anfragen, die sich während der Abwicklung des Geschäftsprozesses ergeben, entdecken und verbieten. Abbildung 4 beschreibt den Aufbau des schwachstellensensitiven Sicherheitsmonitors.

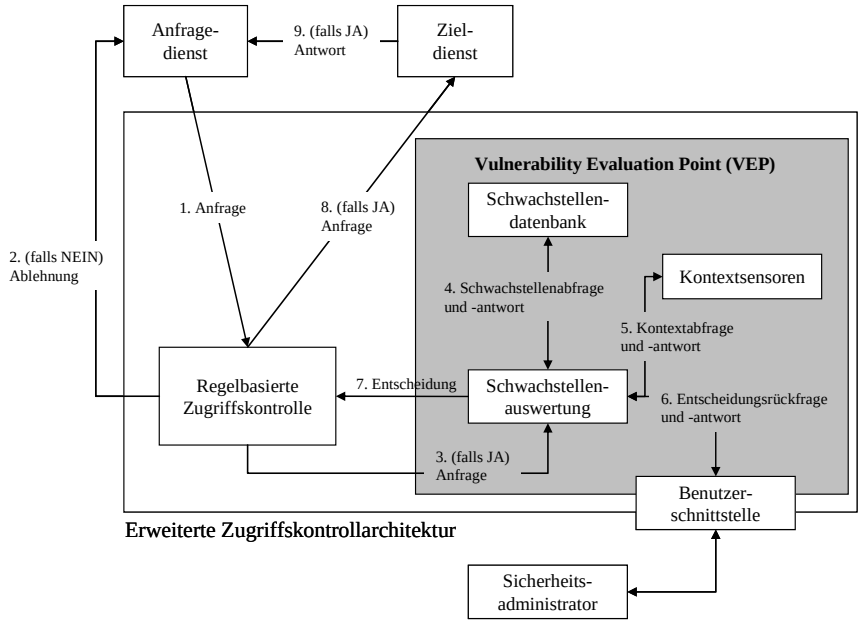

Abb. 4.: Aufbau des Sicherheitsmonitors VEP

5.2 Lösungsansatz für Systemtests

Abbildung 5 gibt einen Überblick über die unterschiedlichen Abstraktionsebenen auf denen Testentscheidungen gefällt werden [Bor07]. Entscheidungen auf niederer Ebene bedingen zunächst eine Entscheidung auf nächst höherer Ebene.

- *Anforderungsebene*: Identifikation der zu testenden Entitäten; Spezifikation testbarer Anforderungen; Prüfung ob die Testbarkeit der Anforderungen vorliegt.
- *Testzielebene*: Festlegung der zu testenden Entitäten und der Methoden zur Definition von Testfällen, Testendekriterien, usw.
- *Logische Testebene*: Detaillierung der definierten Ziele; Definition der logischen Umgebung, der logischen Daten (u.a Eingabedaten, Vor- und Nachbedingung eines Testfalls), der Testschritte sowie der Testsequenzen (Reihenfolge der auszuführenden Testfälle).
- *Ausführungsebene*: Konkretisierung der logischen Testfälle; dabei werden für einen logischen Testfall mehrere konkrete Testfälle definiert. Entscheidungen auf dieser Ebene betreffen die konkreten Umgebungen, die getestet werden müssen, die konkreten Daten sowie die konkreten Testschritte und -sequenzen.
- *Auswertungsebene*: Prüfung, ob ein Testfall erfolgreich war oder nicht, welche Fehler aufgetreten sind, oder ob das Testende erreicht wurde und die Testaktivitäten abgebrochen werden können.

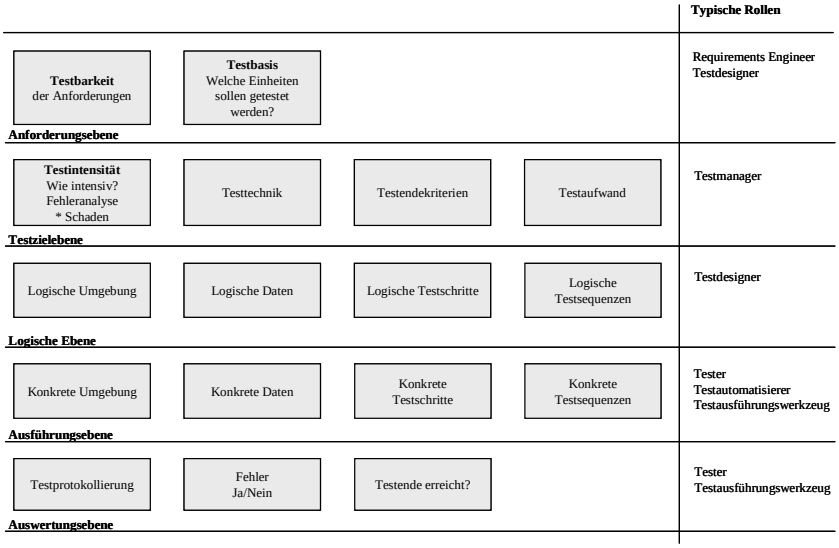

Abb. 5.: Ebenen des Testens

PAT3 – Musterbasierte Testfallermittlung aus Anwendungsfällen

Zur Ermittlung der Testfälle wird der Lösungsansatz PAT3 (Process, Automation, Testability, Transformation, Traceability) verwendet und arbeitet mit Mustern (eng. pattern). Ausgangspunkt aller Muster sind dabei Fehlerklassen, die durch die Muster adressiert werden. Da Testaktivitäten mit dem Ziel ausgeführt werden Fehler zu finden [Mey79], ist es entscheidend zu wissen, welche Fehlerklassen durch welche Muster erkannt werden. Bisherige Ansätze vernachlässigen diesen Aspekt vollständig oder

decken nicht alle Fehlerklassen ab [Ill06]. Die von PAT3 identifizierten fünf Musterkategorien adressieren Probleme an der Schnittstelle von USW- Anforderungen und Softwaretest:

- *Prozessmuster*: Verbesserung der Prozessschnittstelle von USW- Anforderungen und Softwaretest.
- *Automatisierungsmuster*: Unterstützung bei der Auswahl- und beim Anwendungsprozess von Testwerkzeugen.
- *Testbarkeitsmuster*: Unterstützung bei der Formulierung von testbaren Anforderungen.
- *Transformationsmuster*: Unterstützung bei der Ableitung von Testfällen aus Anforderungen,
    - durch Schritt-für-Schritt Anleitungen zur Ableitung von Testfällen aus Anforderungen (*Abbildungsmuster*) vor.
    - durch eine geeignete Darstellung der Testfälle (*Repräsentationsmuster*)
    - sowie durch mögliche Abdeckungskriterien (*Abdeckungsmuster*) vor.
- *Nachvollziehbarkeitsmuster*: Unterstützung bei der Nachvollziehbarkeit zwischen USW-Anforderungen und Testfällen.

Generierung von Testdatenbanken

Da heutige USW praktisch ausschließlich datenbankgestützt arbeitet, kann aber nur getestet werden, wenn im Vorfeld eine Testdatenbank generiert wurde. Das bedeutet oftmals einen hohen manuellen Aufwand, da zwar einerseits Werkzeuge existieren, die aufbauend auf bestehende Datenbankschemata zufällige Testdaten generieren, diese aber andererseits Testdaten generieren, die nicht auf den spezifischen Bedarf der USW abgestimmt sind.

Als Lösungsansatz kann aber das Reverse Query Processing (RQP) dienen [Bin07]. Dieses ermöglicht es, zusätzlich zum Datenbankschema Abhängigkeiten bei der Eingabe als logische Ausdrücke zu berücksichtigen um damit zu betriebswirtschaftlich sinnvollen Testdatenerzeugung zu gelangen. Beispiele hierzu und Informationen zu den Prototypen finden sich in [Bin06]. Technisch gesehen bekommt RQP als Eingabe eine Anfrage, eine Spezifikation von möglichen Anfrageergebnissen und ein Datenbankschema und liefert als Ausgabe eine Testdatenbank. Die Spezifikation der Anfrageergebnisse und der Anfrage dient dazu, für die USW und ihre Anwendung sinnvolle Testdatenbanken zu erzeugen.

RQP basiert auf zwei Kerntechniken, Modellüberprüfung und Datenbankanfragebearbeitung: Die Modellüberprüfung wird verwendet, um neue Daten zu generieren, die bestimmte Bedingungen erfüllen. Das klassische Modell zur Bearbeitung von Datenbankanfragen mittels Parsing, semantischer Analyse, logischer Optimierung und Ausführung durch eine Fließbandverarbeitung wird angewendet und erweitert, um sehr große Datenmengen (bis zu Terrabytes) mit akzeptablen Laufzeiten zu erzeugen.

Ausführung von Tests

Zur Ausführung von Testfällen werden in der Praxis überwiegend Capture&Replay-Werkzeuge verwendet: In der Capture-Phase werden die Testfälle und zu erwarteten Ergebnisse aufgezeichnet, in der Replay-Phase die Testfälle abgespielt und die erzielten Ergebnisse mit den erwarteten Ergebnissen der Capture-Phase verglichen. Eine Deltafunktion wird angewendet, um relevante Differenzen zwischen den erzielten und erwarteten Ergebnissen zu ermitteln. Zur Unterstützung von Capture&Replay-Werkzeugen für USW löst die SIKOSA-Methodik folgende Herausforderungen:

- *Deltafunktionen*: Die Deltafunktionen müssen zwei Anforderungen gerecht werden, die für das Testen von USW besonders wichtig sind: (1) sie sollten nur relevante Änderungen auf Webseiten erkennen (z.B. die Anzeige eines aktuellen Datums ist kein Fehler) (2) Deltafunktionen sollten ein „Gedächtnis“, besitzen, so dass Folgefehler und wiederkehrende Fehler erkannt und protokolliert werden.
- *Datenbankreset*: Capture&Replay-Werkzeuge benötigen Unterstützung zur Wiederherstellung des Testdatenbankzustands nach Ausführung eines Testlaufes. Diese Aufgabe muss zur Zeit noch manuell programmiert werden, was einen hohen Zusatzaufwand bedeutet. Die SIKOSA-Methodik bietet effiziente Algorithmen zum automatisierten Datenbankreset.
- *Parallelisierung*: Um die Abdeckung durch Testen zu maximieren, sind Softwaretests parallel auszuführen, was in der Praxis derzeit wiederum nur manuell geschieht. Die oben genannten Algorithmen für den Datenbankreset wurden für die parallele Ausführung angepasst.

Zusammenfassung

Die SIKOSA-Methodik liefert einen Beitrag zur Berücksichtigung von Compliance bei der Softwareentwicklung und unterstützt (IT-)Entscheider bei der Beschaffung und / oder (Weiter-)Entwicklung von USW. Auf der Basis funktionaler und nicht-funktionaler Geschäftsprozessanforderungen leitet sie testbare funktionale und nicht-funktionalen USW-Anforderungen ab und ermöglicht deren automatisierten Test. Durch die nachweisliche Berücksichtigung der Geschäftsprozesssemantik beim automatisierten Softwaretest kommt sie damit der Anforderung nach „mehr“ Transparenz nach und steigert die Validität der USW.

Zugleich erhöht sie auch die Effizienz des Softwareentwicklungsprozess als solchem, da durch die vertikale Integration Fehler (z.B. Auslassungen, Widersprüche etc.) insgesamt reduziert werden und die mögliche Fehleranzahl, die sich bis zur Implementierung fortpflanzen könnte, gering gehalten wird.

Referenzen

[Bie06] T. Bieser, H. Fürstenau, S. Otto, D. Weiß: Requirements Engineering. In: S. Kirn, O. Herzog, P. Lockemann, O. Spaniol (Hrsg.): Multiagent Engineering. Theory and Applications in Enterprises. Heidelberg 2006, S. 359-381.

[Bin07] C. Binnig, D. Kossmann, R. Lo: Reverse Query Processing. Generating Meaningful Test Databases. In: ICDE 2007. Istanbul, April 2007.

[Bin06] C. Binnig, D. Kossmann, E. Lo: Testing Database Applications. In: SIGMOD 2006. Chicago 2006, S. 739-741.

[Bor07] L. Borner, T. Illes-Seifert, B. Paech: The Testing Process – A Decision Based Approach, In: ICSEA 2007. Cap Esterel 2007. Im Erscheinen.

[Bru05] N. Bruno, S. Chaudhuri: Flexible database generators. In: VLDB. Trondheim 2005, S. 1097-1107.

[Brü01] H. Brücher, R. Endl: Erweiterung von UML zur geschäftsregelorientierten Prozessmodellierung. Heidelberg 2002.

[Can06] J.C. Cannon, M. Byers: Compliance Deconstructed. In: ACM Queue 4 (2006) 7, S. 30-37.

[Cha04] D. Chays, Y. Deng, P.G. Frankl, S. Dan, F.I. Vokolos, E.J. Weyuker: An AGENDA for testing relational database applications. In: Software Testing, verification and reliability 14 (2004) 1, S. 17-44.

[Dan99] J. Dandl: Objektorientierte Prozeßmodellierung mit der UML und EPK. Arbeitspapiere WI 1999 Nr. 12/99:. http://geb.uni-giessen.de/geb/volltexte/2004/1634/pdf/-Apap_WI_1999_12.pdf, Abruf am 2007-05-21.

[Eri00] H.-E. Eriksson, M. Penker: Business Modelling with UML. New York 2000.

[Fer01] A. Ferdian: Comparison of Event-driven Process Chains and UML Activity Diagram for Denoting Business Processes. http://www.sts.tu-harburg.de/pw-and-m-theses/2001/Ferd01.pdf, Abruf am 2007-05-21.

[Gra94] J. Gray, P. Sundaresan, S. Englert, K. Baclawski, P.J. Weinberger: Quickly generating billion-record synthetic databases. In SIGMOD. Minneapolis 1994, S. 243-252.

[Gra05] D. Graham, M. Fewster: Software Test Automation. Effective Use of Test Execution Tools. Harlow 1999.

[Her06] A. Herrmann, B. Paech: MOQARE = Misuse-oriented Quality Requirements Engineering – Über den Nutzen von Bedrohungsszenarien beim RE von Qualitätsanforderungen. In: Softwaretechnik-Trends 26 (2006) 1, S. 13–14.

[Her05] A. Herrmann, B. Paech: Quality Misuse**.** In: REFSQ - Workshop on Requirements Engineering for Software Quality, Essener Informatik Berichte, 2005, S. 193-199.

[Ill06] T. Illes, B. Paech: How Well Can Use Cases Support Test Case Development - An Analysis Based on a Defect Taxonomy. In: IFIP Working Conference on Software Engineering Techniques. Warsaw 2006.

[Kir06] S. Kirn, B. Paech, G. Müller, D. Kossmann: Die SIKOSA-Methodik. Arbeitsbericht. Hohenheim, Heidelberg, Freiburg, Zürich 2006.

[Kno07] G.F. Knolmayer: Compliance-Nachweise bei Outsourcing von IT-Aufgaben. In: WIRTSCHAFTSINFORMATIK 49 (2007) Sonderheft, S. 98-106.

[Kor06a] B. Korherr, B. List: Aligning Business Processes and Software Connecting the UML 2 Profile for Event Driven Process Chains with Use Cases and Components. In: CAiSE´06. Luxembourg 2006, S 19-22.

[Kor06b] B. Korherr, B. List: A UML 2 Profile for Event Driven Process Chains. http://www.wit.at/people/korherr/publications/confenis2006.pdf, Abruf am 2007-05-21.

[Liu99] Y. Liu: Regression Testing Experiments and Infrastructure. Master Thesis. Oregon State University 1999.

[Mey79] G.J. Meyers: The Art of Software Testing. New York 1979.

[Neu93] A. Neufeld, G. Moerkotte, P.C. Lockemann: Generating consistent test data for a variable set of general consistency constraints. In: VLDB 2 (1993) 2, S. 173-213.

[Nor04] O.S. Noran: Business Modelling: UML vs. IDEF. http://www.cit.gu.edu.au/~noran/Docs/UMLvsIDEF.pdf, Abruf am 2007-05-21.

[Oes03] B. Oestereich, C. Weiss, C. Schröder, T. Weilkiens, A. Lenhard: Objektorientierte Geschäftsprozessmodellierung mit der UML. Heidelberg 2003.

[Pae03] B. Paech, K. Kohler: Task-driven Requirements in object-oriented Development. In: J. Leite, J. Doorn (Hrsg.): Perspectives on Requirements Engineering. o.A. 2003.

[Sac06] S. Sackmann, J. Strüker, R. Accorsi: Personalization in privacy-aware highly dynamic systems. In: Communications of the ACM 49 (2006) 9, S. 32–38.

[Sch02] A.-W. Scheer: ARIS –Vom Geschäftsprozess zum Anwendungssystem. Berlin 2002.

[Sch00] A. Scheer; F. Habermann: Making ERP a success. In: Communications of the ACM 43 (2000) 3, S.57-61.

[Sch97] A.-W. Scheer: Wirtschaftsinformatik. Referenzmodelle für industrielle Geschäftsprozesse. 7. Aufl., Berlin 1997.

[Wei07] D. Weiß, J. Kaack, S. Kirn et al.: Die SIKOSA-Methodik. Unterstützung der industriellen Softwareproduktion durch methodisch integrierte Softwareentwicklungsprozesse. In: WIRTSCHAFTSINFORMATIK 49 (2007) 3, S. 1-11.

[Woh06] S. Wohlgemuth, G. Müller: Privacy with Delegation of Rights by Identity Management. In: ETRICS 2006. Freiburg 2006, S. 175–199.