

Einsatz von Modellen für das risikominimierende, anforderungsbasierte Testen von Softwaresystemen (ranTEST)

Thomas Rinke^{*}, Thomas Bauer⁺, Andreas Metzger^{*},
Christopher Robinson-Mallett⁺, Robert Eschbach⁺, Klaus Pohl^{*,#}

[*] Software Systems Engineering Universität Duisburg-Essen Schützenbahn 70 45127 Essen {thomas.rinke, andre- as.metzger, klaus.pohl}@sse.uni-due.de	⁺ Fraunhofer IESE Fraunhofer-Platz 1 67661 Kaiserslautern {bauer, mallett, eschbach}@iese.fraunhofer.de	[#] Lero (The Irish Software Engineering Research Center) University of Limerick Limerick, Ireland pohl@lero.ie
--	--	--

Kurzfassung

Testen ist die stichprobenartige Ausführung von Software zum Zweck der Fehlerfindung oder zum Nachweis bestimmter Qualitätseigenschaften. Selbst dann, wenn möglichst repräsentative und fehlersensitive Testfälle ausgewählt wurden, kann es im Feldeinsatz des Systems zu Fehlverhalten kommen. Die Häufigkeit und die Folgeschwere des Auftretens von Fehlverhalten während des Betrachtungszeitraums bestimmen das mit dem Einsatz einer Software verbundene Risiko. Risikobasierte Testansätze berücksichtigen das mit dem Einsatz des Softwaresystems verbundene Risiko der Nichterfüllung der Qualitätsanforderungen. Existierende Ansätze zur Risikobestimmung nutzen Expertenschätzungen zur Bestimmung der Häufigkeit und Schwere von Softwarefehlverhalten. Eine Änderung der Schadenswahrscheinlichkeit während der Entwicklung (z.B. wenn Defekte in der Software korrigiert werden) erfordert eine Neubestimmung der Schätzwerte. Um diese kontinuierliche Neubestimmung des Risikos zu ermöglichen, wird im ranTEST-Projekt die Entwicklung eines neuartigen Risikomodells vorgeschlagen. Das Risiko wird über verschiedene Indikatoren quantifiziert und auf spezielle Qualitätseigenschaften bezogen. Insbesondere werden Testergebnisse noch während der Entwicklung genutzt, um die Risikobewertung anzupassen. Dadurch wird es möglich eine entwicklungsbegleitende Risikoeinschätzung durchzuführen und die Planung von konstruktiven und analytischen Qualitätssicherungsmaßnahmen anzupassen.

1. Einleitung

Der erschöpfende Test von Softwaresystemen ist aufgrund der sehr großen Menge möglicher Eingaben und Benutzungen nur in trivialen Fällen möglich. Mit Hilfe analytischer Testverfahren kann Software daher nur stichprobenartig überprüft werden. Auch wenn dieser Überprüfung idealerweise mittels repräsentativer, fehlersensitiver, reproduzierbarer und wirtschaftlicher Testfälle erfolgt, können dynamische Qualitätssicherungstechniken die Korrektheit einer Software nicht beweisen (siehe z.B. [Li02]). Ein Beweis vertraglich vereinbarter Qualitätsei-

genschaften ist ebenso problematisch. Die Auslieferung von Software ist daher stets mit der Gefahr verbunden, dass diese fehlerbehaftet ist und potentiell zu Fehlverhalten führt.

Wenn ein Fehlverhalten wichtige Qualitätseigenschaften verletzt, kann das Softwaresystem seinen Wert für den Benutzer verlieren und beträchtlichen Schaden anrichten. Für viele Softwaresysteme ist beispielsweise die Performanz ein entscheidendes Qualitätsattribut, da eine inhaltlich korrekte aber verspätete Antwort des Systems weniger Wert besitzt (z.B. bei Börsenwerten) oder ein Sicherheitsrisiko darstellen kann (z.B. bei der Steuerungs-Software eines Airbags). Die Berücksichtigung verschiedenartiger Qualitätsanforderungen ist besonders wichtig, da diese stark von verschiedenen Nutzungsumgebungen abhängen. Beispielsweise macht es einen großen Unterschied für die Performanz, ob 10 oder 10.000 Anwender das System gleichzeitig benutzen.

Um das Risiko, das mit einer Verletzung von Qualitätsanforderungen einhergeht, zu minimieren, sollten die kritischen Elemente eines Softwaresystems – Komponenten aber auch Anwendungsfälle (Use-Cases) – intensiver als weniger kritische Elemente getestet werden. Diese Vorgehensweise wird als risikobasiertes Testen bezeichnet. Die Einschätzung des Risikos eines Elements kann zur Bestimmung des Testaufwands für den Test dieses Elements dienen. Diese Aufwandsbetrachtung schlägt sich direkt auf Art und Anzahl der abzuleitenden Testfälle nieder. Es ist für den Erfolg dieser Herangehensweise entscheidend, das Risiko möglichst genau und nachvollziehbar einzuschätzen. Unpräzise Schätzungen können zu einer mangelhaften Testabdeckung und Verteilung des Testaufwands führen.

Das Vorgehen im ranTEST-Projekt verfolgt einen anforderungsbasierten Ansatz zum Test eines Softwaresystems. Deshalb liegt der Fokus des Projekts auf der Erarbeitung von Lösungen zur Beurteilung von Risiken, zur Dokumentation von Qualitätsanforderungen und zur Verteilung von Testaufwand ausgehend von Anwendungsfällen. Anwendungsfälle werden vermehrt in der Praxis zur Spezifikation des Verhaltens von Softwaresystemen eingesetzt. Die risikobewerteten Anwendungsfälle dienen auch als Ausgangspunkt für die Testfallableitung in ranTEST.

Bisherige Ansätze des risikobasierten Testens nutzen im Wesentlichen eine Bestimmung des Risikos auf Basis von Expertenschätzungen. Dieses Vorgehen ist aufwändig, da einerseits die Erfahrung zunächst gewonnen werden muss und andererseits unklar ist, unter welchen Umständen eine Anpassung der Risikobewertung vorgenommen werden muss. Das Risikomanagement ist kein einmaliger Vorgang, der nur zu Projektbeginn durchgeführt werden muss (vgl. z.B. [Wa04], [Pi04]). Vielmehr ändern sich die Risiken während des Projektverlaufs (z.B. wenn Defekte in der Software korrigiert werden) und müssen daher regelmäßig neu eingeschätzt werden. Insbesondere sollte auch eine kontinuierliche Risikobewertung nach Projektabschluss während des Betriebs der Software erfolgen.

Dieser Beitrag skizziert die Lösungsidee, welche im ranTEST-Projekt zur Lösung der oben genannten Probleme angegangen wird. Diese Lösungen basieren auf der Entwicklung eines Testmodells auf Basis eines Risikomodells. Das Risikomodell soll das Expertenwissen explizit machen und eine weitgehend automatische Risikobestimmung erlauben. Das Testmodell soll die Informationen über das Risiko nutzen und somit jederzeit eine an die vorgefundene Situation angepasste Testfallplanung und Testfallableitung ermöglichen.

In Abschnitt 2 werden zunächst der Risiko-Begriff und existierende Ansätze zum risikobasierten Testen näher betrachtet, bevor die Idee eines modellbasierten Ansatzes zum risikobasierten Testen in den Abschnitten 4 und 4 näher erörtert wird. Abschnitt 5 beschäftigt sich mit dem Nutzen von Risiko- und Testmodell. Abschnitt 6 fasst die Erkenntnisse zur Entwick-

lung des Testmodells zusammen und gibt einen Ausblick auf die geplanten Arbeiten im ran-TEST-Projekt.

2. Risiko und risikobasiertes Testen

Risiken werden in Projektrisiken und Produktrisiken unterteilt (vgl. z.B. [Pi04]). Projektrisiken sind Risiken, welche die erfolgreiche Durchführung des Projekts gefährden. So besteht beispielsweise in einem Projekt immer die Gefahr, dass nicht ausreichend qualifizierte Mitarbeiter zur Verfügung stehen. Produktrisiken hingegen sind Risiken, die sich durch den Einsatz des Produkts ergeben, wie z.B. das Risiko falscher/ungenauer Berechnungen eines Programms. Produktrisiken werden besonders durch nicht entdeckte Fehler beeinflusst. Mit Hilfe von Qualitätssicherungsmaßnahmen und anschließender Fehlerbehebung können die Produktrisiken verringert werden.

Das Risiko ist abhängig von den Einflussgrößen Schadenswahrscheinlichkeit (oder Schadenshäufigkeit) und Schadensausmaß. Vielfach wird das Risiko vereinfacht als Produkt dieser Einflussgrößen dargestellt ([Aa06], [Wa04]). Das Risiko R ist damit besonders hoch, wenn die Wahrscheinlichkeit des Auftretens P und/oder das Schadensausmaß D hoch sind. Es kann festgehalten werden: $R = f(P, D)$.

Existierende Ansätze des risikobasierten Testens nutzen vielfach Experten um das Schadensausmaß und die Schadenswahrscheinlichkeit zu schätzen ([Aa06], [Pi04]). Dazu werden beide Faktoren i.d.R. auf einer Skala von 1 bis 3 bewertet um Teststrategien auf Basis der Ergebnisse abzuleiten. Bei der Bewertung werden implizit typische Einflussfaktoren auf das Schadensausmaß und die Schadenswahrscheinlichkeit berücksichtigt. So ist die Schadenswahrscheinlichkeit unter anderem von der Nutzungshäufigkeit, der Anzahl der Schnittstellen oder der Änderungshäufigkeit der Anforderungen abhängig. Das Schadensausmaß wird unter anderem durch den Aufwand für die Fehlerbehebung, Vertragsstrafen oder Einnahmeausfälle durch Nichtverfügbarkeit eines Dienstes beeinflusst ([Aa04]).

Allerdings bleibt unklar, welchen Einfluss die einzelnen Faktoren jeweils im Detail besitzen. Viele der Faktoren, die für die Einschätzung genutzt werden, sind nur schwer zu quantifizieren oder werden durch die Projektorganisation selbst beeinflusst. So lässt sich der Einfluss einer neuen Entwicklungsumgebung schlecht in Zahlen ausdrücken und die Entscheidung, erfahrene Entwickler für problembehaftete Komponenten zu nutzen, ist bereits eine Entscheidung zur Risikobehandlung.

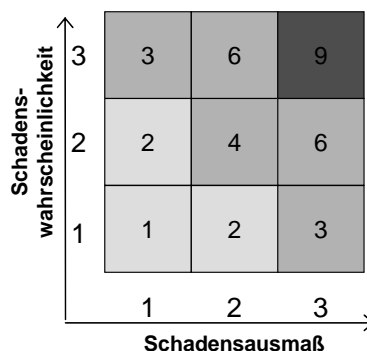


Abbildung 1: Eine Risikomatrix

Abbildung 1 zeigt eine einfache Risikomatrix (ein sehr rudimentäres Risikomodell) mit drei Bewertungsmöglichkeiten für die Faktoren Schadenswahrscheinlichkeit und Schadensmaß. Das Risiko wird hier als Produkt beider Maße berechnet, d.h. $f(P, D) = P \cdot D$. Die ermittelten Risikowerte stehen in den jeweiligen Matrixfeldern. Je höher dieser Wert ist, desto höher ist auch das Risiko. Auf Basis dieser Matrix lassen sich verschiedene Teststrategien ableiten. So sollten Komponenten mit einer Risikobewertung von 6 oder 9 intensiver getestet werden als Komponenten mit einer Risikobewertung von 2 oder 3. Es ist auch möglich, Komponenten mit einer Risikobewertung von beispielsweise 1 gar nicht zu testen, da das Risiko als vertretbar eingestuft wird. Welche Teststrategie jedoch für welche Risikobewertung gewählt wird, ist für jedes Projekt in Abhängigkeit von der jeweiligen Situation zu entscheiden.

Wie eine Beurteilung der erzielten Testergebnisse in Bezug auf die Risikobewertung aussieht, ist bisher unbeantwortet. Die Testergebnisse können dafür sprechen, dass die Risikobeurteilung gut bzw. schlecht war. Im letzten Fall wird eine Neubewertung der Risiken nötig, die auf dem gleichen Vorgehen beruht und somit von der Verfügbarkeit von Experten abhängig ist.

3. Der Nutzen von Risikomodell und Testmodell

Dieser Artikel beschreibt das als Ergebnis des ranTEST-Projekts angestrebte modellbasierte Vorgehen zum risiko- und anforderungsbasierten Testen mit dem Fokus auf Qualitätsanforderungen (vgl. Abbildung 2). Das Risiko einzelner Anwendungsfälle soll dabei weitgehend automatisch ausgehend von wesentlichen, quantifizierbaren Einflussfaktoren abgeschätzt werden. Dafür werden diese Faktoren zunächst identifiziert, messbar beschrieben und zueinander und zu dem Risiko in Beziehung gesetzt. Ziel ist ein möglichst vollständiges Risikomodell zu erhalten, das zuverlässige Schätzungen des Risikos erlaubt.

Das Risikomodell liefert einen entscheidenden Input bei der Ausgestaltung eines Testmodells. Aus den Anwendungsfällen und deren Szenarien wird ein Testmodell erstellt, das durch Nachvollziehbarkeitsinformationen um die Risikoaspekte ergänzt werden kann.

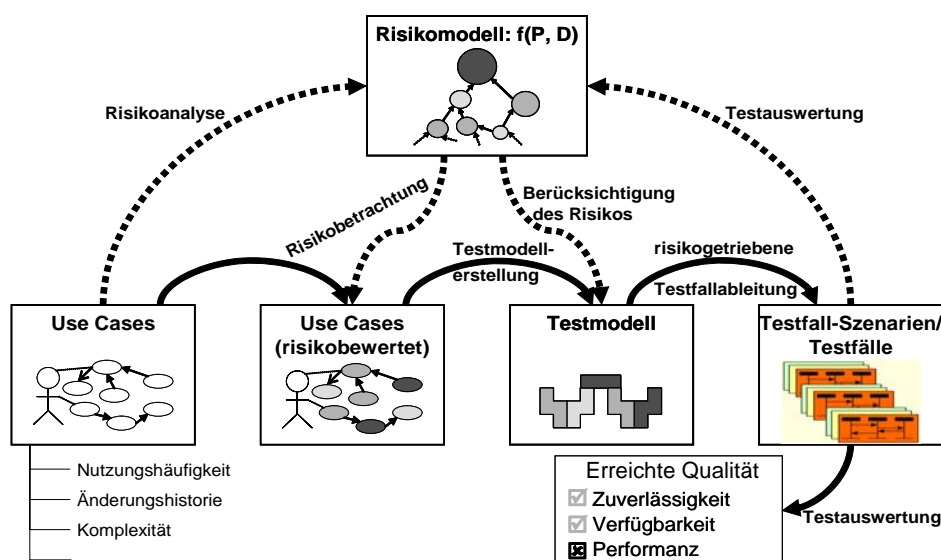


Abbildung 2: Einsatz von Risiko- und Testmodell für die risikominimierende Testfallableitung

Aus diesem Testmodell können gezielt Testfälle abgeleitet und priorisiert werden. Durch die Auswertung der durchgeführten Tests ist zum einen eine Einschätzung der erreichten Qualität möglich, die Grundlage für die Auslieferung des Softwaresystems sein kann. Zum anderen hat die Testauswertung Auswirkung auf die Parameter des Risikomodells. Diese Informationen werden in das Testmodell übernommen, um weitere Tests risikominimierend abzuleiten.

Durch ein solches Vorgehen (vgl. Abbildung 2) wird es möglich, eine kontinuierliche Risikobestimmung während der Entwicklung durchzuführen, ohne dabei permanent auf Expertenschätzungen zurückgreifen zu müssen. Neben dem Einsatz während der Systementwicklung sehen wir einen signifikanten Beitrag des ranTEST-Ansatzes im Bereich des kontinuierlichen Monitorings von Softwaresystemen. Dieses Monitoring kann man sich als einen kontinuierlichen Test von Software in einer sich stetig ändernden Umgebung vorstellen, weshalb hier insbesondere eine kontinuierliche Neubestimmung des Risikos relevant ist.

4. Auf dem Weg zu einem Risikomodell

Ziel unseres risikobasierten Testansatzes ist es, auf Grundlage jederzeit verfügbarer Informationsquellen eine kontinuierliche Anpassung der Eingaben in das Risikomodell vorzunehmen und damit eine von Experten unabhängige Risikoschätzung zu erreichen. Als Informationsquellen können vorhandene (oder explizit für das risikobasierte Testen definierte) Entwicklungsartefakte genutzt werden.

Zu den Informationsquellen können zum Beispiel Informationen zu Anforderungen (vgl. z.B. [MD99], [SW05]), Kennzahlen des Designs (vgl. [Wo00]) oder Informationen aus Software-Archiven (vgl. [DZZ06]) genutzt werden. Verallgemeinert stellen die Änderungshistorie, die Nutzungshäufigkeit und die Komplexität wichtige Anhaltspunkte für das Risiko dar. Es bleibt zu untersuchen, wie Qualitätsanforderungen die Risikobewertung beeinflussen, welche Wechselwirkungen existieren und wie sie ausreichend detailliert dargestellt werden können.

Basierend auf der initialen Risikoeinschätzung werden Testfälle geplant und ausgeführt, um Rückschlüsse auf das tatsächliche Risiko zu ziehen. Bei der Aufdeckung von Fehlern in einer Komponente steigt die Wahrscheinlichkeit, dass weitere Fehler in dieser Komponente enthalten sind (siehe [My01], [BB01]). Die Testergebnisse können somit genutzt werden um die Risikoeinschätzung anzupassen, wodurch Testen ein Hilfsmittel für die Risikoforschung (vgl. [Wa00]) ist. Dieses Vorgehen kann auch im laufenden Betrieb der Software genutzt werden, in dem auf Basis von im Feld gefundenen Fehlern oder von Fehlern in einem kontinuierlichen Monitoring die Anpassung der Risikobewertung erfolgt.

Durch die auf Basis der Tests durchgeführte Fehlerbehebung kann die Schadenswahrscheinlichkeit verringert werden. Somit berücksichtigt unser Ansatz auch den Aspekt der Risikominderung (vgl. [Wa00]). Das Schadensausmaß bleibt, sofern keine anderen Maßnahmen zur Absicherung gegenüber dem Schaden getroffen werden, unverändert. Dass z.B. eine Vertragsstrafe gezahlt werden muss, wenn ein Dienst nicht verfügbar ist, hängt nicht von der Wahrscheinlichkeit ab, dass dieser Dienst ausfällt. Bei unserem Lösungsansatz konzentrieren wir uns daher auf die modellbasierte Bestimmung der Schadenswahrscheinlichkeit. Die Abschätzung des Schadensausmaßes kann zu Beginn der Entwicklung – wie gehabt – durch Experten erfolgen.

Eine wichtige Qualitätseigenschaft, die in unserem Ansatz u.a. betrachtet werden soll, ist die Zuverlässigkeit. Software-Zuverlässigkeit ist definiert als die Wahrscheinlichkeit des fehler-

freien Einsatzes von Software innerhalb eines bestimmten Zeitraums in einer bestimmten Umgebung [IEEE610]. Zuverlässigkeitsmodelle ermöglichen die Bestimmung oder Vorhersage der Zuverlässigkeit eines Softwaresystems (vgl. [Lyu96], [LN95]). Mit Hilfe der Zuverlässigkeitsanalysen und -prognosen können dann auch Produktrisiken identifiziert werden. Es wird zwischen den statischen, maßbasierten Modelle und den dynamischen, so genannten Zuverlässigkeitswachstumsmodellen unterschieden. Die Zuverlässigkeitsvorhersagen werden dann entweder als Fehlerinhalt bzw. Fehlerdichte eines Systems oder als Verteilungsfunktion der zukünftig zu erwartenden Fehlverhalten bzw. Ausfälle angegeben.

5. Risikogetriebene Testfallableitung über ein Testmodell

Die Ableitung von Testfällen zur Überprüfung der Qualitätsanforderungen und zur Absicherung der Risiken sollte nach einem systematischen Ansatz erfolgen. Das modellbasierte Testen ist ein solch systematischer Ansatz, der auf dem Einsatz von Modellen, d.h. Abstraktion, des Systems beruht. Diese Abstraktion ermöglicht eine Konzentration auf die für das Testziel wesentlichen Eigenschaften des Systems. Ein Testmodell stellt die einheitliche Basis für die Testableitung dar, die auf Basis des Testmodells automatisiert erfolgen kann (siehe z.B. [Be90], [Bi99], [Re05]). Die erstellten Testfälle dienen dann zur Überprüfung der Erfüllung der Anforderungen und der Absicherung der Risiken.

In unserem Ansatz wird im Folgenden ein spezielles Testmodell eingeführt, das die zuvor identifizierten Risiken in Betracht zieht. Das Testmodell wird aus den betrachteten Use Cases und deren Szenarien abgeleitet. Daher bietet sich die Beschreibung des Testmodells mit Hilfe von zum Beispiel Zustandsautomaten oder Aktivitätsdiagrammen an. Die abgeleiteten Testszenarien werden auf Basis von Sequenzen z.B. mittels Message Sequence Charts (vgl. [ITU Z120]) beschrieben.

Über Informationen zur Verfolgbarkeit zwischen Use Cases und Risiken sowie zwischen Use Cases und Testmodell können die Risiken, die für die Use Cases identifiziert worden sind, in das Testmodell überführt werden. Somit lassen sich aus dem Testmodell Testfälle ableiten, die das Risiko berücksichtigen. Dies kann zum einen bedeuten, dass die Anzahl der abgeleiteten Testfälle an das Risiko angepasst wird. Eine Anlehnung an die Bestimmung der Anzahl der Testfälle auf Basis der Priorität von Use Cases (vgl. [BBM01]) ist dabei denkbar. Dabei können auch gezielt keine Testfälle abgeleitet werden, denn wo kein Risiko vorhanden ist, muss auch nicht getestet werden (vgl. [Aa06]). Zum anderen kann über das Risiko auch die Art der Testfallableitung festgelegt werden, ob also z.B. verschiedene Techniken der Testfallableitung genutzt werden sollen. Das Testmodell berücksichtigt weiterhin die betrachteten Qualitätsanforderungen und erlaubt eine gezielte Bestimmung der für die jeweilige Anforderung relevanten Testfälle. Somit kann bei der Testauswertung ermittelt werden, ob die Qualitätsanforderungen erfüllt sind und die Risiken des Systems abgesichert wurden.

Über den Einsatz des Testmodells und der damit verbundenen Möglichkeit, Testfälle automatisch abzuleiten, wird es möglich, im Rahmen eines kontinuierlichen Monitorings Testfälle zu erstellen und auszuführen, die die Eigenschaften und aktuelle Bewertung des Systems in Betracht ziehen. Im laufenden Betrieb kann so ein statistischer Test kontinuierlich durchgeführt werden.

Die Testauswertung stellt in unserem Ansatz einen weiteren wichtigen Aspekt dar. So kann auf Basis der Testergebnisse zum einen eine Entscheidung über die Freigabe des Systems getroffen werden. Darüber hinaus können mit Hilfe der Testauswertung die Parameter des

Risikomodells angepasst werden, was ein darauf abgestimmtes weiteres Testvorgehen erlaubt. Dies ist ebenso auf Basis von identifizierten Fehlern, einer Fehlerbehebung und anschließenden Regressionstests möglich.

Ein Modell zur Testfallableitung, das seit Jahren in der Praxis eingesetzt wird, ist das so genannte Benutzungsmodell (engl. Usage Model, vgl. [PPL99]). Dieses Modell erlaubt die Erstellung von Testfällen unter Beachtung des operationalen Profils der Software. Es ist möglich verschiedene Benutzungsarten zu modellieren, wie beispielsweise die Benutzung mit Fokus auf Zuverlässigkeit oder Performanz. Die Benutzungsmodelle basieren auf Zustandsautomaten mit Markov-Eigenschaften. Sie ermöglichen die Ableitung von Testfällen in Form von Sequenzen, die sich als gängige Darstellungsform zur Testfallbeschreibung etabliert haben. Die Spezifikationssprache TTCN-3 (Testing and Test Control Notation, vgl. [ITU Z140]) erlaubt die Spezifikation von Testfällen als Sequenzen von Ereignissen und Aktionen, beispielsweise mit der Semantik von Message Sequence Charts. Aktuelle Forschungsergebnisse demonstrieren die Eignung dieser Notation für die Darstellung statistisch gewichteter Sequenzen (vgl. [DZ05]). Die Mächtigkeit der TTCN-3 Core Language (vgl. [ITU Z140] Part 1) kann zur Darstellung von Zustandsautomaten mit Markov-Eigenschaften genutzt werden. Es ist anhand der Fallstudien zu untersuchen, in wie weit sich TTCN-3 zur Beschreibung domänenspezifischer Anforderungen und als Beschreibungssprache für Zustandsautomaten mit Markov-Eigenschaften als praxisgerechte Lösung bewährt.

6. Zusammenfassung und Ausblick

In diesem Beitrag wurde der Ansatz des ranTEST-Projekts zum risikominimierenden, anforderungsbasierten Testen von Softwaresystemen vorgestellt. Auf Basis einer objektiven Bewertung der Häufigkeit und Schwere des Auftretens von Häufigkeit unter expliziter Berücksichtigung spezieller Qualitätseigenschaften wird ein Risikomodell zur Quantifizierung der negativen Einflussfaktoren auf ein Softwaresystem erstellt. Das Risikomodell wird zur Priorisierung von Anforderungen genutzt.

Basierend auf Anforderungen und Risikomodell wird ein mit Prioritäten annotiertes Testmodell vorgeschlagen, das zur systematischen Testfallableitung dient. Durch Testläufe werden die Qualitätseigenschaften des Softwaresystems überprüft und die Risiken abgesichert. Die Testergebnisse liefern Aufschluss über die Genauigkeit der Risikobewertungen und ermöglichen eine nachträgliche Kalibrierung der Risikokennzahlen. So ergibt sich ein iterativer Ansatz mit dem Ziel der permanenten Verbesserung der Qualität des Softwaresystems.

Über existierende Lösungen hinaus, wird durch unseren Ansatz eine besondere Berücksichtigung spezieller Qualitätsanforderungen angestrebt. Insbesondere sollen die derzeit verbreiteten Expertenschätzungen der Risikokenngrößen durch objektive, automatisch generierbare Maße ersetzt werden.

Im ranTEST-Projekt (siehe auch www.rantest.de) werden die in diesem Beitrag skizzierten Lösungen in einer Kooperation zwischen dem Fraunhofer IESE und der Universität Duisburg-Essen als Forschungseinrichtungen sowie der Siemens AG Transportation Systems und market maker Software AG als Industriepartner entwickelt und validiert.

Die Industriepartner werden die entwickelte Technik zum risikominimierenden Testen in ausgewählten Projekten aus den Domänen der eingebetteten Systeme und der Finanzsoftware

anwenden und validieren. Die Vorbereitung der Studien sowie die Auswertung und Zusammenführung der Ergebnisse werden durch die Forschungseinrichtungen durchgeführt.

Die unterschiedlichen Domänen der Industriepartner sollen die breite Anwendbarkeit des Verfahrens zeigen. Interessante Punkte stellen die unterschiedlichen Arten der betrachteten Software und die daraus abgeleiteten fokussierten Risiken und Qualitätseigenschaften dar. Es ist zu erwarten, dass die für die verschiedenen Qualitätsanforderungen und Domänen zu erarbeitenden Risiko- und Testmodelle Unterschiede aufweisen. Eine Verallgemeinerung der Ergebnisse und die Integration der Modelle in ein allgemeines Modell werden angestrebt. In zukünftigen Studien soll auch untersucht werden, ob das zu entwickelnde Verfahren in anderen Domänen und mit anderen Risiken und Qualitätseigenschaften anwendbar ist.

7. Literaturverzeichnis

- [Aa06] Van der Aalst, L.: Risk Based Test Strategy Judged. Proceedings 7th International Conference on Software Testing, ICSTEST, Düsseldorf, Mai 2006.
- [BB01] Boehm, B.; Basili, V. R.: Software Defect Reduction Top 10 List. IEEE Computer 34(1), Januar 2001, S. 135-137.
- [BBM01] Basanieri, F.; Bertolino, A.; Marchetti, E.: CoWTeSt: A Cost Weighted Test Strategy. Escom-Scope 2001, London, England, 2.-4. April 2001, S. 387-396.
- [Be90] Beizer, B.: Software Testing Techniques, International Thomson Computer Press, 1990.
- [Bi99] Binder, R. V.: Testing Object Oriented Systems: Models, Patterns and Tools, Addison-Wesley Professional, 1999.
- [DZ05] Dulz, W.; Zhen, F.: MaTeLo - Statistical Usage Testing by Annotated Sequence Diagrams, Markov Chains and TTCN-3, Third International Conference On Quality Software, p. 336, 2005.
- [DZZ06] Diel, S.; Zeller, A.; Zimmermann, T.: Was Software-Archive erzählen. In: (Biel, B.; Book, M.; Gruhn, V. Hrsg.): Software Engineering 2006, Fachtagung des GI-Fachbereichs Softwaretechnik, 28.-31.3.2006 in Leipzig, Gesellschaft für Informatik, Bonn, 2006; S. 39-50.
- [IEEE610] IEEE610.12-1990, IEEE Standard Glossary of Software Engineering Terminology; ANSI/IEEE Standard 610.12-1990, 1990.
- [ITU Z120] Spezifikation des Standards Z.120 für Sequenzdiagramme (MSCs), ITU - International Telecommunication Union, 1999
- [ITU Z140] Spezifikation des Standards Z.140 für TTCN-3, ITU - International Telecommunication Union, 2003
- [Li02] Liggesmeyer P.: Software-Qualität. Spektrum-Verlag, Heidelberg, 2002.
- [LN95] Lakey, P. B.; Neufelder, A. M.: System and Software Reliability Assurance Notebook, Rome Laboratory, 1995.

- [Ly96] Lyu, M. R.: Handbook of Software Reliability Engineering, IEEE and McGraw-Hill, 1996.
- [MD99] Malaiya, Y. K.; Denton, J.: Requirements Volatility and Defect Density. 10th International Symposium of Software Reliability Engineering, November 1999, S. 285-294.
- [My01] Myers, G. J.: Methodisches Testen von Programmen. 7. Auflage. Oldenbourg, München, 2001.
- [Pi04] Pinkster, I. et al.: Successful Test Management – An Integral Approach. Springer, Berlin, 2004.
- [PPL99] Prowell S. J. ; Poore, J. H.; Linger, R. C., Cleanroom Software Engineering, Addison-Wesley Professional, 1999.
- [Re05] Reuys, A. et al.: Model-Based System Testing of Software Product Families. In: Pastor, O.; Falcao e Cunha, J. (Eds.) Proceedings 17th Conference on Advanced Information Systems Engineering, CAiSE 2005 (Porto, Portugal, 13–17 June, 2005). Springer, Heidelberg, 2005, 519–534.
- [SW05] Srikanth, H.; Williams, L.: On the Economics of Requirements-Based Test Case Prioritization. Proceedings of the seventh international workshop on Economics-driven software engineering research (EDSER), S. 1-3, 2005.
- [Wa04] Wallmüller, E.: Risikomanagement für IT- und Software-Projekte: Ein Leitfaden für die Umsetzung in der Praxis. Hanser, München, 2004.
- [Wo00] Wong, W.E. et al.: Applying design metrics to predict fault-proneness: a case study on a large-scale software system. Software – Practice and Experience 30(14), November 2000, S. 1587-1608.