Risikobasierte Ableitung und Priorisierung von Testfällen für den modellbasierten Systemtest[1]

Thomas Bauer[+], Heiko Stallbaum*, Andreas Metzger*, Robert Eschbach[+]

[+] Fraunhofer IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern
thomas.bauer, robert.eschbach@iese.fraunhofer.de

* Software Systems Engineering
Universität Duisburg-Essen
Schützenbahn 70, 45127 Essen
heiko.stallbaum, andreas.metzger@sse.uni-due.de

Abstract: Der erschöpfende Test von Softwaresystemen ist aufgrund komplexer Schnittstellen und Umgebungsbedingungen in der Realität kaum möglich. Zusätzlich schränkt das vorhandene Testbudget die Anzahl der durchführbaren und auswertbaren Testfälle ein. Deshalb sind eine Auswahl zweckmäßiger Testfälle und deren Priorisierung für die Reihenfolge der Testausführung notwendig. Ein viel versprechender Ansatz zur Priorisierung von Testfällen ist das risikobasierte Testen. Ein risikobasierter Testplan stellt sicher, dass Teile einer Software umso intensiver getestet werden, je höher das Risiko ist, dass diese Teile beim Einsatz der Software zu einem nicht vernachlässigbaren Schaden führen. Eine direkte Priorisierung von Testfällen auf Basis dieses Risikos ist bei komplexen Softwaresystemen jedoch aufwendig und jede Aktualisierung der Risikobewertung erfordert eine Neupriorisierung der gesamten Testfallmenge. Dieser Beitrag beschreibt einen risikobasierten und modellbasierten Testansatz, der eine weitgehend automatische Generierung und Priorisierung von Testfällen ermöglicht. Ausgangspunkt sind annotierte UML-Diagramme aus der Anforderungs- und Entwurfsphase. Aus den Diagrammen wird ein Testmodell erstellt und mit Risikoinformationen angereichert. Aus dem angereicherten Testmodell werden automatisch Testfälle abgeleitet, priorisiert, ausgeführt und ausgewertet. Die Ergebnisse aus einer industriellen Anwendung des Lösungsansatzes sind beschrieben.

1 Motivation

Testen ist der in der Praxis vorherrschende Ansatz zur Aufdeckung von Defekten in Software. Ein erschöpfender Test von Software ist jedoch aufgrund der sehr großen Menge möglicher Eingaben nur in trivialen Fällen möglich. Deshalb kann eine Software nur stichprobenartig, im Idealfall mittels repräsentativer, fehlersensitiver und

[1] Teile dieser Arbeit wurde mit Mitteln des Bundesministeriums für Bildung und Forschung (BMBF) im Rahmen der Forschungsoffensive „Software Engineering 2006“ unter dem Förderkennzeichen 01 IS E09 B (ranTEST) gefördert. Weitere Informationen sind unter http://www.rantest.de zu finden.

reproduzierbarer Testfälle, getestet werden. Die Korrektheit einer Software kann durch Tests nicht bewiesen werden [Di72]. Daher verbleibt bei Software, die getestet wurde, stets die Gefahr, dass diese fehlerbehaftet ist und während der Nutzung potentiell zu einem Fehlverhalten führen kann.

Die Wahrscheinlichkeit und das Schadensausmaß eines solchen Fehlverhaltens bestimmen das mit der Nutzung der Software verbundene Risiko – das Produktrisiko. Um dieses Risiko zu minimieren, sollten kritische Teile einer Software intensiver als weniger kritische Teile getestet werden. Diese Idee wird als *risikobasiertes Testen* bezeichnet (siehe z.B. [Ba99], [Am00], [Pi04], [Aa06]). Die kritischen Teile enthalten hierbei entweder mit hoher Wahrscheinlichkeit Defekte oder das Fehlverhalten dieser Teile führt zu einem hohen Schadenausmaß. In einem *risikobasierten Testplan* kommen deshalb zuerst solche Testfälle zur Ausführung, die kritische Teile der Software überprüfen. Die risikobasierte Priorisierung von Testfällen stellt sicher, dass kritische Teile einer Software intensiver getestet werden als weniger kritische Teile und dass das Risiko, das mit der Nutzung der Software verbunden ist, minimiert wird.

Entscheidend für den Erfolg bei der risikobasierten Priorisierung von Testfällen sind zwei Punkte: Zum einen müssen die mit dem Einsatz einer Software verbundenen Risiken genau und nachvollziehbar eingeschätzt werden. Zum anderen müssen Testfälle auf dieser Basis so priorisiert werden, dass der resultierende Testplan die bewerteten Risiken adäquat abdeckt. Eine mangelhafte Priorisierung kann zu einer schlechten Testabdeckung und Verteilung des Testaufwands führen. Bisherige risikobasierte Testansätze geben keine hinreichende Anleitung zur Ableitung und Priorisierung von Testfällen.

In diesem Beitrag wird der ranTEST-Ansatz (risko- und anforderungsbasiertes TESTen) zur systematischen und weitgehend automatischen Ableitung und Priorisierung von Testfällen vorgeschlagen. Das zentrale Element des Ansatzes ist ein *risikoorientiertes Testmodell*. Es stellt eine Adaption von zustandsbasierten Testmodellen dar, wie sie beim statistischen Testen verwendet werden. Elemente des Testmodells werden durch Risiken bewertet, die auf Anwendungsfallebene ermittelt wurden. Über den Einsatz des Testmodells und der damit verbundenen Möglichkeit, Testfälle automatisch abzuleiten, zu priorisieren, auszuführen und auszuwerten, wird es möglich, komplexe Softwaresysteme effizient zu testen. Der Vorteil des Ansatzes liegt zudem darin, dass auf eine Änderung der Risiken während der Entwicklung (z.B. durch eine Fehlerkorrektur) durch eine automatische Neupriorisierung der Testfälle mit Hilfe des aktualisierten risikoorientierten Testmodells reagieren werden kann.

Der folgende Beitrag ist wie folgt strukturiert. Abschnitt 2 stellt existierende Ansätze zur Ableitung und Priorisierung von Testfällen vor. In Abschnitt 3 wird der ranTEST-Ansatz näher erörtert. Die Ergebnisse aus einer industriellen Anwendung des Ansatzes werden in Abschnitt 4 beschrieben.

2 Stand der Technik

In diesem Abschnitt werden zunächst existierende risikobasierte Techniken zur Ableitung und Priorisierung von Testfällen diskutiert. Anschließend werden statistische Testtechniken vorgestellt, welche die Grundlage für die automatische Testfallableitung im ranTEST-Ansatz darstellen.

2.1 Risikobasiertes Testen

Es existieren einige Arbeiten, welche vorschlagen das Risiko explizit bei der Ableitung und Priorisierung von Testfällen zu berücksichtigen. So führt Bach verschiedene Heuristiken für die Risikoidentifikation und -bewertung ein und regt an, die gewonnenen Risiken im Testprozess zu nutzen [Ba99]. Allerdings bleibt offen, wie konkrete Testfälle tatsächlich risikobasiert abgeleitet und priorisiert werden können. Auch Rosenberg et al. adressieren die Risikobewertung im Kontext des risikobasierten Testens ohne auf die konkrete Ableitung und Priorisierung von Testfällen einzugehen [RSG99]. Van der Aalst [Aa06] bzw. Amland [Am00] schlagen die Berechnung einer sogenannten Risk Score für jeden Prozess bzw. Risk Exposure für jedes Modul der zu testenden Software vor. Auf Basis der Ergebnisse kann entschieden werden, was als nächstes zu testen ist. Die konkrete Ableitung von Testfällen wird hierbei jedoch nicht betrachtet. Pinkster et al. [Pi04] und Schaefer [Sc05] verbinden Risiken mit Prioritäten für das Testen. Eine detaillierte Technik zur Ableitung der Testfälle entsprechend der Priorisierung wird jedoch nicht beschrieben.

Chen und Probert schlagen einen risikobasierten Priorisierungsansatz für den Regressionstest vor [CP03]. Darüber hinaus können auch die Priorisierungsansätze von Srikanth und Williams [SW05] und Elbaum et al. [EMR01] als risikobasiert angesehen werden. Eine initiale Ableitung von Testfällen wird von diesen Ansätzen jedoch nicht betrachtet, da sie lediglich den Regressionstest fokussieren.

Zusammenfassend ist festzustellen, dass die existierenden Ansätze für das risikobasierte Testen keine zufriedenstellende Anleitung für die Erstellung von Testfällen geben. Es wird entweder davon ausgegangen, dass solche Testfälle schon existieren, oder aber es wird sich darauf beschränkt Risiken zu identifizieren und zu bewerten ohne auf die Bestimmung konkreter Testfälle einzugehen.

2.2 Statistisches Testen

Das statistische Testen wurde ursprünglich vom Software Quality Research Lab der Universität Tennessee entwickelt [Pr99]. Die Technik wurde erfolgreich in mehreren Industrieprojekten eingesetzt um große Softwaresysteme zu systematisch zu testen (siehe [PT98], [Ba07]). Beim statistischen Testen erfolgt die Generierung, Ausführung und Auswertung der Testfälle automatisch. Sie orientiert sich an der Häufigkeit des Auftretens einer bestimmten Eingabe.

Das zentrale Element des statistischen Testens ist das *zustandsbasierte Testmodell*, das die verfügbare und benötigte Information über die zu testende Software (*Testobjekt*), die *Testumgebung* und die Benutzung des Testobjekts (*Benutzungsmodell*) zusammenfasst. Der Aufbau des Testmodells ist eine kreative, schwer zu automatisierende Aufgabe. Er erfolgt mit Hilfe einer systematischen Inspektion der Anforderungen durch die *sequenzbasierte Spezifikation* (*SBS*). Hierbei werden systematisch die möglichen Systembenutzungen in Form von Eingabesequenzen ermittelt und untersucht. Die Eingabesequenzen werden dann als Pfade im Testmodell abgebildet. Die Transitionen des Testmodells entsprechen diskreten Eingaben der Eingabesequenzen. Sie werden mit *Transitionsgewichten* annotiert, die die Häufigkeit des Auftretens einer Eingabe beschreiben.

3 Der ranTEST-Ansatz

Abbildung 1 gibt einen Überblick die Aktivitäten und Entwicklungsartefakte des ranTEST-Ansatzes zur risikobasierten Generierung und Priorisierung von Testfällen. Der Ansatz ist anforderungsbasiert, d.h. die in Anwendungsfällen dokumentierten Anforderungen an eine Software sind der Ausgangspunkt des Lösungsansatzes. Risiken, die mit der Nutzung der Software einhergehen, stellen eine weitere wesentliche Eingabe in den Ansatz dar. Sie werden durch Risikoanalysen ermittelt und an Anwendungsfällen annotiert.

Als zentrales Element des vorgeschlagenen Ansatzes wird ein *risikoorientiertes Testmodell* eingeführt, das auf Anwendungsfallebene ermittelte Produktrisiken explizit berücksichtigt und auf Testmodellelemente abbildet. Es stellt eine Adaption von zustandsbasierten Testmodellen dar, wie sie beim statistischen Testen verwendet werden. Während das Testmodell beim statistischen Testen das Benutzungsmodell eines Testobjekts enthält, beschreibt es im Kontext des hier vorgeschlagenen Ansatzes das *Risikoprofil* des Testobjekts. Zur Erstellung dieses risikoorientierten Testmodells ist die vorherige systematische Verfeinerung der risikobewerteten Anwendungsfälle notwendig.

Das risikoorientierte Testmodell ermöglicht schließlich die automatische und risikobasierte Ableitung und Priorisierung von Testfällen und mithin die Erstellung eines risikobasierten Testplans. Unser Vorgehen verfolgt somit einen durchgehenden risikobasierten Ansatz, beginnend mit risikobewerteten Anforderungen über ein risikoorientiertes Testmodell bis hin zum risikobasierten Testplan.

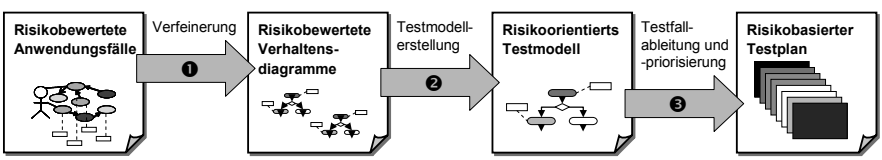

Abbildung 1: Überblick über den ranTEST-Ansatz

Im Folgenden werden die Aktivitäten 1 bis 3 aus Abbildung 1 erläutert und es wird dargestellt, welche Entwicklungsartefakte hierbei genutzt bzw. erzeugt werden.

3.1 Die Basis: Risikobewertete Anwendungsfälle

Ein *Anwendungsfall* spezifiziert Aktionsfolgen einschließlich Alternativ- und Ausnahmesequenzen, die ein System bei der Interaktion mit anderen Objekten ausführt, um einen Mehrwert zu erbringen (vgl. [RJB05], [Po07]). Anwendungsfälle werden im ranTEST-Ansatz in der *Unified Modelling Language* (*UML*, [RJB05], [OMG04], [OMG05]) auf abstrakter Ebene durch Anwendungsfalldiagramme beschrieben. Der Einsatz von Anwendungsfällen zur Dokumentation von Anforderungen ist heute gängige Praxis und bildet daher den Ausgangspunkt unseres Ansatzes.

Die Kenntnis der Risiken, die mit dem Einsatz einer Software einhergehen, ist eine wesentliche Grundlage des vorgeschlagenen Lösungsansatzes. Produktrisiken sind dementsprechend im Rahmen einer Risikoanalyse zu ermitteln und zu bewerten. Die Realisierung einer genauen und nachvollziehbaren Risikoanalyse steht mit Verweis auf die entsprechende Literatur (z.B. [Pi04], [Wa04]) nicht im Fokus dieses Beitrages. Das *Risiko* kann grob durch zwei Einflussfaktoren beschrieben werden: Durch die Wahrscheinlichkeit des Auftretens einer Gefahr, die einen Schaden verursacht (*Schadenswahrscheinlichkeit*), sowie durch den Schweregrad des Schadens (*Schadensausmaß*) [IEC00]. Um Risiken zu quantifizieren kann das Produkt der Eintrittswahrscheinlichkeit eines Risikoereignisses und des Schadensausmaßes des Ereignisses genutzt werden (vgl. z.B. [Ba99], [RSG99], [Am00], [Aa06]). Die Schätzung des Schadensausmaßes erfolgt durch Experten unter Kenntnis des Nutzungskontextes der Software und der Domäne. Die Quantifizierung der Schadenswahrscheinlichkeit bleibt Experten überlassen (vgl. z.B. [Ba99], [Pi04], [Aa06]) oder kann aus Metriken abgeleitet werden (vgl. z.B. [RSG99], [No00], [SM07]).

Die Resultate der Risikoanalyse müssen explizit dokumentiert und in einem ausreichenden Umfang formalisiert werden. Als formale Notation wird im ranTEST-Ansatz das *UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms* (*UML QoS-Profil*, [OMG06]) genutzt und eine *QoSCharacteristic* Risiko definiert. Die entsprechenden Kommentare formulieren Einschränkungen in der *Object Constraint Language* (*OCL*) und werden an Anwendungsfälle geheftet.

3.2 Schritt 1: Verfeinerung der risikobewerteten Anwendungsfälle

Durch die Verfeinerung der risikobewerteten Anwendungsfälle wird das Verhalten eines jedes Anwendungsfalls und dessen Risiken präzisiert. Wir schlagen einen Verfeinerungsansatz vor, der sich in zwei Aktivitäten gliedert: Die Verhaltensmodellierung und die Risikopropagierung.

3.2.1 Verhaltensmodellierung

Jeder Anwendungsfall kann mehrere *Szenarien* beinhalten. Ein Szenario beschreibt ein konkretes Beispiel für die Erfüllung bzw. Nichterfüllung eines Ziels [Po07]. Zur

Modellierung des Verhaltens eines Anwendungsfalls werden seine Haupt-, Alternativ- und Ausnahmeszenarien sowie deren Bedingungen herangezogen.

Als formale Notation für die Verhaltensmodellierung schlagen wir *Verhaltensdiagramme* der UML vor. Sie liefern eine gute Grundlage für die Verhaltensspezifikation auf verschiedenen Detaillierungsstufen. Die Darstellung des Verhaltens eines Anwendungsfalles in *Aktivitätsdiagrammen*, *Zustandsdiagrammen* oder *Sequenzdiagramme* kann so gewählt werden, dass die Hauptverhaltensmerkmale des Testobjektes herausgestellt werden.

3.2.2 Risikopropagierung

Die Propagierung der Risiken von Anwendungsfällen auf deren Verhaltensdiagramme besteht im Wesentlichen darin, Risiken eines Anwendungsfalls systematisch an dessen Verhaltensbeschreibung weitergegeben, d.h., die in Kommentaren annotierten Risiken im Verhaltensdiagramm zu dokumentieren. Die Genauigkeit, mit der Risiken annotiert werden, kann in Abhängigkeit von projektspezifischen Anforderungen und Rahmenbedingungen (z.B. Budget und Ressourcen) variieren. Wir schlagen ein iteratives Vorgehen vor, bei dem die Genauigkeit der Risikopropagierung schrittweise erhöht wird bis die gewünschte *Propagierungstiefe* erreicht ist. Zur Klassifikation der Propagierungstiefe werden folgende *Propagierungsebenen* samt Formalismus zur Dokumentation der Risiken vorgeschlagen:

1. *Verhaltensebene*: Bei der Risikopropagierung auf Verhaltensebene werden alle Risiko-Annotationen eines Anwendungsfalls in dessen Verhaltensdiagramm übernommen. Diese Risiken gelten für das Verhalten des Anwendungsfalls im Ganzen. Dies wird im Verhaltensdiagramm dadurch dokumentiert, dass die Annotation (UML-Kommentar) mit keinem Modellelement verbunden ist (Abbildung 2, Kommentar 1).

2. *Szenarioebene*: Ein Szenario ist einen Pfad in einem Verhaltensdiagramm, der durch Angabe aller Bedingungen beschrieben werden kann, die zutreffen müssen, damit das Szenario eintritt. Risiken werden einem einzelnen Szenario im Verhaltensdiagramm zugeordnet, indem die entsprechenden Kommentare der Verhaltensebene erweitert werden. Dazu wird das Schlüsselwort *implies* (log. Implikation) der OCL genutzt, welches es erlaubt, Bedingungen zu formulieren: Die Aussage auf der linken Seite fordert die Erfüllung aller Bedingungen, die zutreffen müssen, damit ein Szenario eintritt. Die Aussage auf der rechten Seite dokumentiert das für dieses Szenario geltende Risiko (Abbildung 2, Kommentar 2).

3. *Aktivierungs-, Aktivitäts- oder Zustandsebene*: Risiko-Annotationen werden bei Propagierung auf Aktivierungsebene mit den sie betreffenden Aktivierungen verbunden (Abbildung 2, Kommentar 3). Werden Objekte aktiviert, so führen sie Aktivitäten aus, die Zustandsänderungen herbeiführen können. Die Risikopropagierung auf Aktivierungsebene in Sequenzdiagrammen entspricht der Propagierung auf Aktivitäten in Aktivitätsdiagrammen (Aktivitätsebene) bzw. auf Zustände in Zustandsdiagrammen (Zustandsebene).

4. *Nachrichten- oder Transitionsebene*: Bei der Risikopropagierung auf Nachrichtenebene werden Risken an zugehörige Nachrichten im Sequenzdiagramm annotiert (Abbildung 2, Kommentar 4). Für Aktivitäts- und Zustandsdiagramme treten anstelle der Nachrichten Transitionen (Transitionsebene).

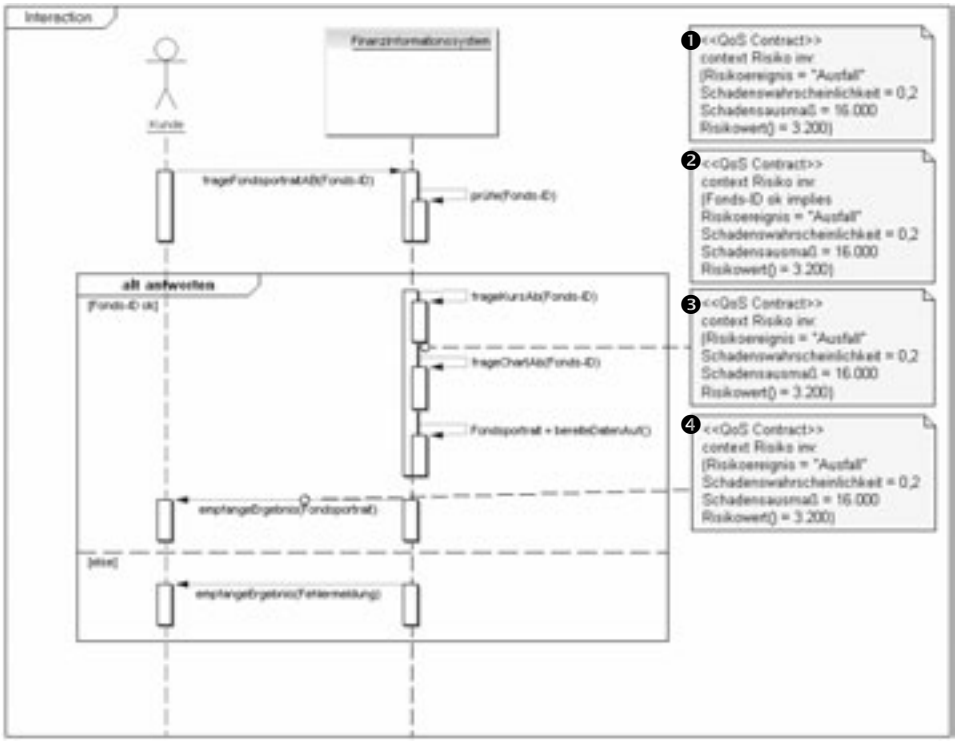

Abbildung 2: Dokumentation von Risiken in unterschiedlicher Detaillierung

Bei der Risikopropagierung kann es zu Konkurrenzsituationen kommen, die ihren Ausgangspunkt in unterschiedlichen *Propagierungsquellen* haben. Diese Quellen sind Risiken, denen ähnliche Risikoereignisse auf einer höheren Propagierungsebene zugrunde liegen. Aus ihnen resultiert eine *Propagierungskonkurrenz*, wenn sie ein und dasselbe Modellelement eines Verhaltensdiagramms betreffen. Zur Auflösung von Propagierungskonkurrenzen können folgende Strategien angewendet werden:

1. *Verfeinernde Risikopropagierung*: Propagierungskonkurrenzen können durch Verfeinerung aufgelöst werden, indem die beteiligten Risiken bei der Propagierung so präzisiert werden, dass sie keine gemeinsamen Modellelemente mehr betreffen. Betrifft ein Risiko z.B. mehrere Modellelemente einer Propagierungsebene in unterschiedlicher Ausprägung (oder manche Modellelemente gar nicht), so kann

eine Verfeinerung durch Annotation mehrerer Kommentare mit unterschiedlicher Ausprägung des Risikos an den verschiedenen Modellelementen dargestellt werden.

2. *Vereinigende Risikopropagierung*: Zur Auflösung von Propagierungskonkurrenzen durch Vereinigung werden die konkurrierenden Risiken bei der Propagierung so kombiniert, dass das resultierende Risiko deren Aussagen abdeckt. Mit Bezug zur Notation im UML QoS-Profil bedeutet das, dass die Invarianten der konkurrierenden Kommentare vereinigt werden. Dies kann z.B. durch Benennung des Gesamtrisikoereignisses, der Quantifizierung seiner Schadenwahrscheinlichkeit (siehe Abschnitt 3.1) und der Addition des Schadensausmaßes über alle Invarianten geschehen.

3.3 Schritt 2: Erstellung eines risikoorientierten Testmodells

Im Folgenden wird das risikoorientierte Testmodell des ranTEST-Ansatzes beschrieben, das die zuvor identifizierten, bewerteten und propagierten Risiken explizit berücksichtigt. Als Basis dienen zustandsbasierte Testmodelle, wie sie beim statistischen Testen verwendet werden (siehe Abschnitt 2.2). Sie enthalten Transitionsgewichte, die das *Benutzungsmodell des Testobjektes* darstellen und zur Steuerung der Testfallgenerierung genutzt werden.

Der Aufbau des Testmodells erfolgt durch die *sequenzbasierte Spezifikation (SBS)* auf Basis der vorliegenden Verhaltensdiagramme. Durch systematische Inspektion der Anforderungen an das Testobjekt wird die Systemumgebung mit den möglichen Systemeingaben und -ausgaben identifiziert. Danach werden die gültigen Eingabesequenzen (entspr. Benutzungsszenarien) schrittweise untersucht. Die Menge aller Eingabesequenzen beschreibt das Testmodell als Zustandsautomaten.

Unsere Idee besteht darin, die Transitionen bezüglich ihrer *Risikoabdeckung* zu bewerten und damit das *Risikoprofil* des Testobjekts zu beschreiben. Bei der automatisierten Testfallableitung werden dadurch risikoreiche Transitionen öfter in den Testfällen vorkommen als risikoarme.

Je nach gewünschter *Propagierungstiefe* wurden in Abschnitt 3.2 das Verhalten der Anwendungsfälle, ihre Szenarien, einzelne Aktivierungen (bzw. Aktivitäten oder Zustände) oder Nachrichten (bzw. Transitionen) in den Verhaltensdiagrammen bezüglich ihres Risikos bewertet. Sie entsprechen im risikoorientierten Testmodell Teilmodellen, Pfaden, Benutzungszuständen oder einzelnen Transitionen.

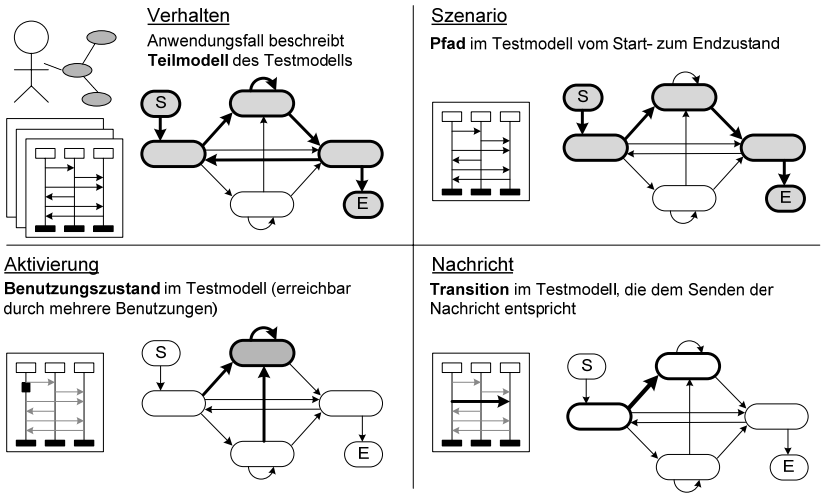

Abbildung 3: Risikopropagierung im Testmodell

Die Abbildung 3 zeigt beispielhaft, wie Risikobewertungen aus Verhaltensdiagrammen ins Testmodell übernommen werden. Auf Szenarioebene werden die Risikobewertungen der Pfade im Verhaltensdiagramm auf Pfade im Testmodell abgebildet. Aktivierungen (bzw. Aktivitäten oder Zustände) entsprechen Benutzungszuständen, die durch unterschiedliche Pfade erreicht werden. Ihre Risikobewertungen werden auf die Gewichte der eingehenden Transitionen abgebildet. Nachrichten (bzw. Transitionen) im Verhaltensdiagramm werden auf Transitionen im Testmodell abgebildet, wobei nur Nachrichten berücksichtigt werden, die von der Umgebung an das Testobjekt gesendet werden. Der systeminterne Nachrichtenfluss spielt für den Aufbau des risikoorientierten Testmodells keine Rolle, da es ein anforderungsbasierter Testansatz ist.

3.4 Schritt 3: Risikobasierte Ableitung und Priorisierung von Testfällen

Die *Testfallableitung* aus dem Testmodell erfolgt automatisiert durch *Traversierung* des Zustandsautomaten. Testfälle entsprechen Pfaden im Testmodell vom Start- zum Endzustand. Bei der Testfallableitung werden die *Transitionsgewichte* berücksichtigt, damit werden Transitionen mit hohen Gewichten öfter durch die Testfälle überprüft als Transitionen mit niedrigen Gewichten. Da im vorgeschlagenen Ansatz die Transitionsgewichte Risiken darstellen, ist die Risikobewertung entscheidend für die Häufigkeit der einzelnen Transitionen in der Testfallmenge.

Die Struktur des Testmodells (Zustände, Transitionen) und die Transitionsgewichte haben einen großen Einfluss auf die Testfalllänge. Längere Testfälle haben mit hoher Wahrscheinlichkeit eine bessere Risikoabdeckung als kürzere Testfälle, da sie das Testmodell stärker überdecken und mehr Transitionen traversieren. Je nach verfügbarem Aufwand können beliebig viele Testfälle generiert werden. Bei der Testplanung muss festgelegt werden, wie viele Testfälle abgeleitet, ausgeführt und ausgewertet werden.

Das Ergebnis ist ein *risikobasierter Testplan* mit einer Menge von Testfällen, die gemäß den verfügbaren Ressourcen und Risikobewertungen abgeleitet wurden. Die Testfälle werden nach ihrer Generierung bezüglich ihrer Risikoabdeckung beurteilt. Im vorgestellten Ansatz werden dazu die Risikobewertungen der durch den Testfall traversierten Transitionen aufsummiert. Die Qualität der Testfälle hängt dadurch im großen Maße von der Qualität der Risikobewertungen ab.

4 Industrielle Anwendung

Der ranTEST-Ansatz wurde im Rahmen des Forschungsprojekts ranTEST an einem Finanzsoftwaresystem des Projektpartners MarketMaker in einer initialen Studie angewendet. Das Testobjekt bearbeitet permanent Kundenanfragen zu Kurs- und Depotdaten, die in einer Konfigurationsdatei spezifiziert sind. Diese Anfragen haben einen Zeitstempel und werden periodisch abgearbeitet, beispielsweise täglich oder wöchentlich. Zusätzlich gibt es die Möglichkeit asynchrone Anfragen zu jeder Zeit zu stellen. Diese zusätzlichen Anfragen werden abgearbeitet, sobald Ressourcen von der Abarbeitung der anderen Anfragen frei sind. Intern werden die Anfragen vor der Bearbeitung in eine Warteschlange eingereiht. Das Ergebnis jeder Anfrage ist ein Bericht in einem gewünschten Format, der dem Kunden ausgeliefert wird.

Relevante Einflussfaktoren sind die Konfigurationsdateien mit den periodischen Anfragen, die Menge und Art der zusätzlichen Kundenanfragen und die Umgebungsbedingungen des Testobjekts (Netzwerklast, Hardware). Die Konfigurationen werden im Folgenden in einfach und komplex unterteilt, die Umgebungsbedingungen in kritisch und unkritisch. Das Risikoereignis im Anwendungsbeispiel ist die Nichterfüllung der nicht-funktionalen Qualitätseigenschaft Performanz, d.h., dass Anfragen nicht rechtzeitig beantwortet werden.

Die Anforderungen wurden in Form von Anwendungsfällen spezifiziert. Die Anwendungsfälle wurden bezüglich ihres Risikos für die Performanz des Gesamtsystems bewertet. Performanzkritische Funktionen wurden mit einer höheren Risikozahl versehen. Die bei der Risikobewertung verwendeten Maße von Schadenswahrscheinlichkeit und Schadenausmaß wurden miteinander verrechnet und das Ergebnis auf eine Risikozahl zwischen 0 (kein Risiko) und 100 (hohes Risiko) abgebildet. Risiken von unter 20 wurden von den Systemexperten als vernachlässigbar eingestuft. Alle Anwendungsfälle wurden in Szenarien verfeinert, die auch bezüglich ihres Risikos bewertet wurden. Aus den bewerteten Szenarien wurde ein Testmodell erstellt, das die Stimulation des Testobjekts durch die Umgebung in Form von Nachrichten oder Funktionsaufrufen beschreibt.

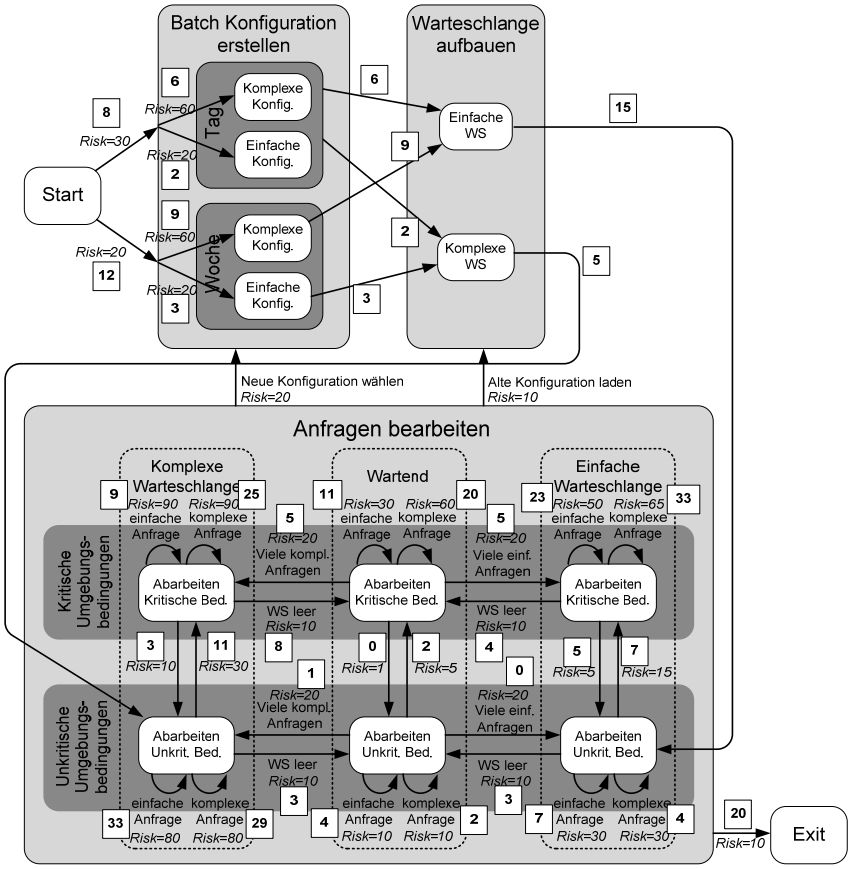

Abbildung 4: Testmodell des Anwendungsbeispiels

Die Abbildung 4 stellt das vereinfachte risikoorientierte Testmodell des Anwendungsbeispiels dar. Es ist ein hierarchischer Zustandsautomat mit einem Start- und Endzustand. Im Zustand *Anfragen bearbeiten* sind die Subzustände nach der Komplexität der Warteschlange (gestrichelte Linie) und nach den Umgebungsbedingungen (dunkelgrauer Hintergrund) gruppiert. Zu jeder Transition ist deren Häufigkeit in den abgeleiteten Testfällen in einem weißen Kästchen aufgeführt. Das risikoorientierte Testmodell wurde zusammen mit Systemexperten von MarketMaker erstellt. Als Ausgangspunkt dienten die Risikobewertungen der Anwendungsfälle, die auf Teilpfade und Transitionen des Testmodells abgebildet wurden. Die Risikobewertung der Transition zum Endzustand dient der Steuerung der Testfalllänge (hier: Risiko = 10). Kleine Werte führen zu längeren Testfällen, da dann die Wahrscheinlichkeit der Transitionsauswahl bei der Testfallableitung kleiner wird. Bei der Erstellung des risikoorientierten Testmodells wurde der Schwerpunkt auf die Bearbeitung und Einreihung von Anfragen gelegt. Die Anwendungsfälle wurden weiter verfeinert und mit Risiken annotiert. Risikoreiche Benutzungen sind vor allem bei komplexen Warteschlangen zu finden, wenn zusätzliche Anfragen gestellt werden.

Aus dem Testmodell wurden 20 Testfälle abgeleitet, die aus insgesamt 346 Testschritten bestehen. Die Häufigkeit der Transitionen in den zufällig abgeleiteten Testfällen ist an den Zahlen in den weißen Kästchen zu erkennen. Die Testfälle können nach ihrer Erstellung bezüglich ihrer Risikoabdeckung bewertet werden, im Beispiel als Summe der Werte der traversierten Transitionen. Die Änderung der Ausführungsreihenfolge der einzelnen Testfälle ist auf Basis der Risikobewertungen möglich.

Der bisherige Ansatz bei MarketMaker umfasste eine manuelle risikoorientierte Testfallerstellung durch Systemexperten. Das Testmodell und die abgeleiteten Testfälle wurden zusammen mit den Systemexperten von MarketMaker diskutiert. Durch die vorgestellte Methode ist es dem Industriepartner möglich, automatisiert Testfälle gemäß dem Risikoprofil der Software abzuleiten und gezielt risikoreiche Benutzungen zu überprüfen. Neben der automatisierten Testfallableitung wurde vom Industriepartner besonders die Änderbarkeit des Testmodells positiv beurteilt. Modifizierte Risikobewertungen können leicht ins Testmodell übernommen werden. Die Anpassung der einzelnen Testfälle entfällt durch die automatische Testfallgenerierung.

Der Aufwand für die Erstellung des Testmodells lag bei 2 Personenwochen, wobei zusätzlicher Aufwand von geschätzten 2-3 Personenwochen für die plattformspezifische Umsetzung (Testskripte an den Transitionen) geplant ist. Die Implementation der Testorakel (erwartete Testergebnisse) auf der Zielplattform wurde als aufwändigster Teil identifiziert. Bei der Modellierung konnte auf detaillierte Anforderungsspezifikationen und Produktdokumentation zurückgegriffen werden, was den Erstellungsprozess beschleunigt hat.

5 Zusammenfassung und Ausblick

In diesem Beitrag wurde der ranTEST-Ansatz für das risikobasierte und modellbasierte Testen von Softwaresystemen vorgestellt. Der Ansatz erlaubt eine weitgehend automatische Generierung und Priorisierung von Testfällen auf Basis des Risikos, das mit der Nutzung der Software einhergeht. Ausgangspunkt sind annotierte UML-Diagramme aus der Anforderungs- und Entwurfsphase, welche mit Risikoinformationen angereichert werden. Aus diesen erweiterten Diagrammen wird ein risikoorientiertes Testmodell abgeleitet, in dem die Risikoinformationen entsprechend verfeinert sind. Auf Basis des risikoorientierten Testmodells werden dann automatisch Testfälle durch Nutzung statistischer Testtechniken abgeleitet.

Eine industrielle Anwendung des Ansatzes an einem Beispiel im Bereich der Finanzsoftware hat gezeigt, dass sich der ranTEST-Ansatz für das effiziente risikobasierte Testen der Qualitätseigenschaft Performanz eignet. Die generierte Priorisierung der Testfälle wurde von den Industrieexperten als geeignet eingestuft.

Im Rahmen des ranTEST-Projekts soll der Ansatz für weitere Qualitätseigenschaften, wie z.B. Verfügbarkeit und Zuverlässigkeit, evaluiert werden. Entsprechende Fallstudien in verschiedenen Anwendungsdomänen sind geplant.

Literaturverzeichnis

[Aa06] Van der Aalst, L.: Risk Based Test Strategy Judged. In: Proc. of 7th ICSTEST, SQS, Düsseldorf, Deutschland, 2006.

[Am00] Amland, S.: Risk-based testing: Risk analysis fundamentals and metrics for software testing including a financial application case study. In: Journal of Systems and Software, vol. 53, no. 3, Elsevier Science Inc., 2000. pp. 287-295

[Ba99] Bach, J.: Risk-Based Testing. How to conduct heuristic risk analysis. In: Software Testing & Quality Engineering Magazine, vol. 1, iss. 6, 1999.

[Ba07] Bauer, T.; Beletski, T.; Boehr, F.; Landmann, D., Eschbach, R., Poore, J.; From Requirements to Statistical Testing of Embedded Systems. In: Proc. of SEAS '07 at 29th ICSE, Minneapolis, USA, 2007.

[CP03] Chen, Y.; Probert, R.L.: A Risk-based Regression Test Selection Strategy. In: Proc. of 14th ISSRE, Fast Abstract, Denver, Colorado, USA, 2003. pp. 305-306

[Di72] Dijkstra, E. W.: Notes on structured programming. In: Dahl, O. J.; Dijkstra, E. W.; Hoare, C. A. R. (Eds.): Structured programming, Academic Press, London, 1972.

[EMR01] Elbaum, S.; Malishevsky, A.; Rothermel, G.: Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization. In: Proc. of 23rd ICSE, Toronto, Ontario, Canada, 2001. pp. 329-338

[IEC00] International Electrotechnical Commission (IEC): International Standard IEC 61508 – Functional safety of electrical/electronic/programmable electronic safety-related systems. 2000.

[No00] Nogueira, J.C. et al.: A Formal Risk Assessment Model for Software Evolution. In: Proc. of 2nd EDSER at 22nd ICSE, Limerick, Ireland, 2000.

[OMG04] Object Management Group (OMG): Unified Modeling Language: Superstructure, v2.0. 05.07.2004. Verfügbar unter http://www.omg.org/docs/formal/05-07-04.pdf.

[OMG05] Object Management Group (OMG): Unified Modeling Language Infrastructure, v2.0. 05.07.2005. Verfügbar unter http://www.omg.org/docs/formal/05-07-05.pdf.

[OMG06] Object Managements Group (OMG): UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms. Version 1.0, formal/2006-05-02, 2006.

[Pi04] Pinkster, I. et al.: Successful Test Management – An Integral Approach. Springer, 2004.

[Po07] Pohl, K.: Requirements Engineering – Grundlagen, Prinzipien, Techniken. Dpunkt, 2007.

[Pr99] Prowell, S. J. et al.: Cleanroom Software Engineering: Technology and Process, Addison Wesley, 1999.

[PT98] Poore, J. H.; Trammell, C. J.: Application of Statistical Science to Testing and Evaluating Software Intensive Systems, Statistics, Testing, and Defense Acquisition: New Approaches and Methodological Improvements, National Academy of Science Press, 1998. Reprinted by IEEE in May 1999.

[RJB05] Rumbaugh, J.; Jacobson, I.; Booch, G.: The Unified Modeling Language Reference Manual. Addison-Wesley, 2005.

[RSG99] Rosenberg, L. H.; Stapko, R.; Gallo, A.: Risk-based Object Oriented Testing. 24th Annual Software Engineering Workshop, Goddard Space Flight Center, 1999.

[Sc05] Schaefer, H.: Risk Based Testing, Strategies for Prioritizing Tests against Deadlines. In: Method and Tools, Winter 2005, vol. 13, no. 4, 2005.

[SM07] Stallbaum, H.; Metzger, M.: Employing Requirements Metrics for Automating Early Risk Assessment. In: Proc. of MeReP07, Palma de Mallorca, Spain, 2007.

[SW05] Srikanth, H.; Williams, L.: On the Economics of Requirements-Based Test Case Prioritization. In: Proc. of 7th EDSER at 27th ICSE, St. Louis, Missouri, USA, 2005.

[Wa04] Wallmüller, E.: Risikomanagement für IT- und Software-Projekte. Hanser, 2004.