Electronic version of an article published as **Maalej W, Bruegge B (Hrsg) Software Enineering 2008 - Workshopband. Fachtagung des GI-Fachbereichs Softwaretechnik (SE 2008), 18.-22. Februar 2008 in München, Germany, LNI P-122, pp. 457-463**

Qualitätsrisiken und deren Abhängigkeiten

Andrea Herrmann

Software Engineering Group, Institut für Informatik
Universität Heidelberg
Im Neuenheimer Feld 326
69120 Heidelberg
andrea.herrmann@informatik.uni-heidelberg.de

Abstract: Es hat sich bewährt, Softwarequalität anhand von Qualitätsrisiken zu beschreiben, d.h. anhand derjeniger Ereignisse oder Zustände, die man durch die Qualitätseigenschaft vermeiden möchte. In dieser Arbeit werden Qualitätsattributen jeweils ein unerwarteter Zustand zugeordnet sowie deren kausale Abhängigkeiten modelliert. Diese Abhängigkeiten können u.a. die Qualitätsvorhersage unterstützen.

1 Einleitung: Qualität versus Qualitätsrisiken

Softwarequalität kann man definieren als das Nicht-Vorhandensein oder die Minimierung von Risiken. Insbesondere wird dies in sicherheitsrelevanten Bereichen so gehandhabt [SO00], [SO01], [Fi03]. An der Universität Heidelberg wenden wir dieses Prinzip erfolgreich auf alle Arten nicht-funktionaler Anforderungen (NFA) an. Interessanterweise beschreiben die Definitionen aller Qualitätsattribute (QA) viel öfter einen unerwünschten Zustand des Systems[1] als den erwünschten Zustand [He07b]. Die gängigen NFA-Metriken messen meist die Auftretenshäufigkeit eines unerwünschten Zustands, manchmal den daraus entstehenden Schaden[2] [He07b].

Wir haben ausgehend vom Standard ISO 9126 [IS91] eine QA-Taxonomie entwickelt, in der jedes QA genau einem unerwünschten Zustand entspricht[3] [He07b]. Im Folgenden wird modelliert, wie diese unerwünschten Zustände kausal zusammenhängen und damit auch die QA und die Erfüllung von NFA. Dies soll uns helfen, Risiken zu quantifizieren, um darauf aufbauend NFA risikobasiert zu priorisieren wie in [He07a] dargestellt.

---

[1] Wobei hier zu dem „System“ nicht nur Hardware und Software zählen, sondern auch Benutzer und sonstiges Personal (Entwickler, Administrator, usw.), Entwicklungs- und Testwerkzeuge, die umgebende physische Welt (z.B. Gebäude) sowie Schnittstellen zu anderen Systemen.

[2] Oft besteht der Schaden in einer erhöhten Wahrscheinlichkeit eines nachfolgenden unerwünschten Zustands.

[3] Im Vergleich zur zweiten Ebene des ISO 9126 Standards wird in unserer Taxonomie das Attribut „Sicherheit“ untergliedert (in Betriebssicherheit, Vertraulichkeit und Überlebensfähigkeit), „Usability“ wird in andere Faktoren zerlegt (Produktivität, Bedienbarkeit und Zufriedenheit), „Portierbarkeit“ wird nicht näher unterteilt (was dann sinnvoll wäre, wenn man die einzelnen Schritte der Portierung betrachten möchte, was bisher jedoch nicht unser Schwerpunkt war).

Es werden zunächst in Abschnitt 2 die vier Typen unerwünschter Zustände dargestellt und in Abschnitt 3 die Zusammenhänge zwischen den unerwünschten Zuständen. Abschnitt 4 gibt einen Ausblick darauf, wo und wie wir diese Abhängigkeiten nutzen möchten und Abschnitt 5 fasst diesen Beitrag zusammen.

2 Unerwünschte Zustände

Softwarequalität wird beschrieben durch ein Spektrum an Softwareeigenschaften, die QA. NFA sind Instanzen von QA und beschreiben eine konkrete Anforderung an eine bestimmte Software. Welchem QA wir welchen unerwünschten Zustand zuordnen, ist in den Abbildungen 1-4 dargestellt. Wir unterscheiden vier Typen unerwünschter Zustände, die sich jeweils im Hinblick auf die sie quantifizierenden Metriken ähneln [He07b]:

- Typ D: Defekte (englisch: fault, defect oder bug) unterscheiden wir danach, von welcher erwünschten Eigenschaft das System abweicht (Abbildung 1).

- Typ F: Eine Fehlfunktion (englisch: failure) bedeutet, dass das Verhalten des Systems vom erwarteten Verhalten abweicht. Eine Fehlfunktion tritt auf, wenn ein Teil des Systems ausgeführt wird, der einen Defekt enthält [Oh84]. In Abbildung 2 sind die Fehlfunktionen dargestellt. Diese werden nach ihrer Ursache unterschieden.

- Typ A: Es entsteht unnötig oder inakzeptabel hoher Aufwand, Kosten oder Ausfallzeit bei Benutzung (Typ Ab) oder Wartung (Typ Aw). Die unerwünschten Zustände des Typs Aw werden nach ihrer Ursache (der durchgeführten Tätigkeit) unterschieden (Abbildung 3), des Typs Ab nach den Folgen (Abbildung 2).

- Typ S: sonstiges, wie Sicherheitsverletzungen (Abbildung 4), Benutzeraktion wird nicht erfolgreich durchgeführt oder Benutzer ist unzufrieden (Abbildung 2).

3. Kausale Abhängigkeiten der unerwünschten Zustände

Die unerwünschten Zustände verschiedener QA hängen kausal miteinander zusammen. Hierbei kann ein unerwünschter Zustand mehrere verschiedene nach sich ziehen oder ein erwünschter Zustand mehrere mögliche Vorgängerzustände haben. Aktivitäten führen von einem unerwünschten Zustand zum nächsten. Um die Abhängigkeiten der unerwünschten Zustände untereinander sowie die Aktivitäten darzustellen, ist ein UML-Aktivitätendiagramm geeignet, aber auch jede andere Notation, die Aktivitäten und Zustände unterscheidet und gerichtet verknüpft.

Wegen seiner Größe kann das eine UML-Aktivitätendiagramm aller QA hier nicht als Ganzes abgebildet werden. Darum zeigen die Abbildungen 1 bis 4 jeweils einzelne Swimlanes dieses Diagramms, einschließlich Querverbindungen zu unerwünschten Zuständen in anderen Swimlanes. Nicht dargestellt sind positive Alternativverläufe, z.B. wenn ein Defekt nicht zu einer Fehlfunktion führt, wenn eine Schwachstelle nicht für einen Angriff ausgenutzt wird. Diese werden bei einer quantitativen Betrachtung berücksichtigt, indem man von einem Zustand zum nächsten nur mit einer gewissen Übergangswahrscheinlichkeit von <100% gelangt.

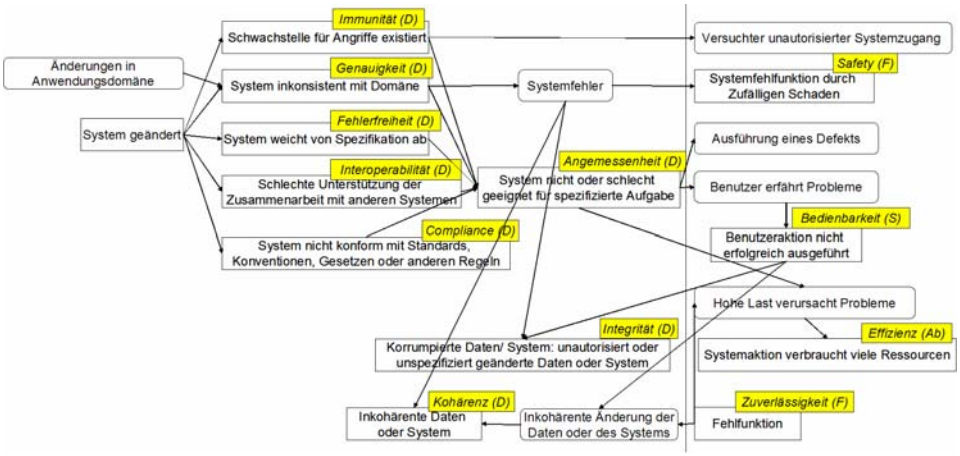

Abbildung 1: Szenario „intended use: software & hardware“; Typ D (Defekte)

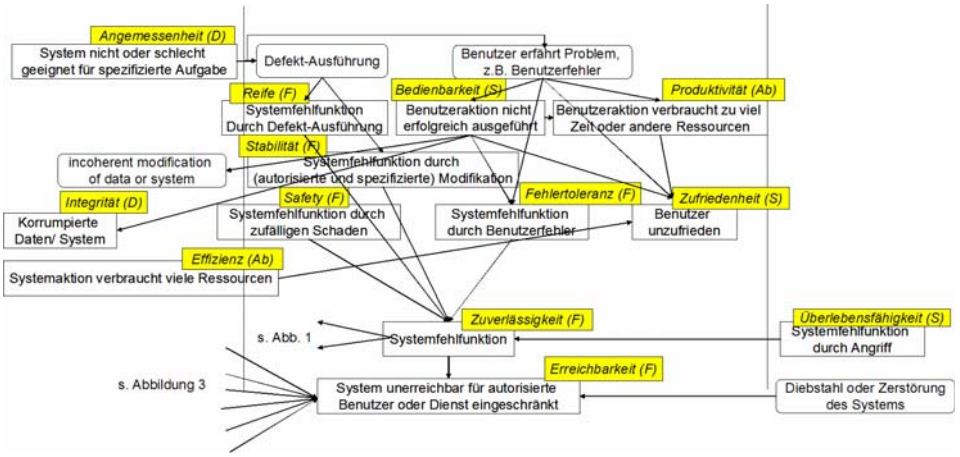

Abbildung 2: Szenario „intended use: user“; Typ F (Fehlfunktionen) und Typ Ab (zu hoher Aufwand oder Kosten während der Benutzung)

Das Aktivitätendiagramm hat vier Swimlanes von links nach rechts: „fault correction, maintenance and enhancement“, „intended use: software & hardware“, „intended use: user“ und „security-relevant misuse“. Diese beschreiben die drei Hauptszenarien, wobei das Szenario „intended use“ der Übersicht halber in die Perspektiven der Software und Hardware einerseits und des Benutzers andererseits aufgeteilt ist. Diese drei Szenarien lassen sich wie folgt beschreiben:

1. **Intended use** (siehe Abbildung 1 und Abbildung 2): Während das System wie vorgesehen benutzt wird (Swimlane “intended use: user”), führen die Defekte (Swimlane “intended use: software and hardware”) zu Fehlfunktionen und Benutzerproblemen (Benutzer im weitesten Sinne, einschließlich Wartungspersonal, Entwicklern, usw.), was zu Defekten in Bezug auf Integrität, Effizienz, Kohärenz oder Zuverlässigkeit führt.
2. **Fault correction, maintenance and enhancement** (siehe Abbildung 3): Diese Aktivitäten führen u.a. dazu, dass das System für autorisierte Benutzer zumindest zeitweise nicht erreichbar ist oder der Dienst eingeschränkt. Außerdem sind dadurch Änderungen am System entstanden, die eventuell neue Defekte enthalten.
3. **Security-relevant misuse** (Abbildung 4): Das Sicherheitsszenario kennt Benutzer, die von der vorgesehenen Benutzung abweichen, während im Szenario “intended use” das System vom vorgesehenen Verhalten abweicht.

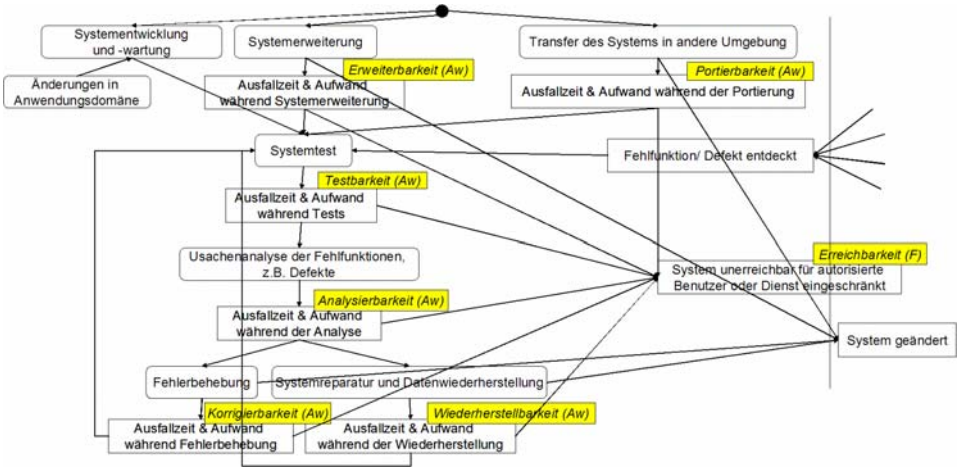

Abbildung 3: Szenario „Fault correction, maintenance and enhancement“, Typ Aw: zu hoher Aufwand oder Kosten während der Wartung

So selbstverständlich die hier dargestellten Abhängigkeiten zwischen unerwünschten Zuständen erscheinen mögen, gab es jedoch bisher keine alle QA umfassende Überlegungen hierzu. Einzelne Arbeiten existieren wie diejenige von Madan et al. [Ma02], welche Übergänge zwischen erwünschten Zuständen in Bezug auf Sicherheit in Zustandsübergangsmodellen darstellt. Singh et al. [SCC01] beschreiben erwünschte Zustände in Sequenzdiagrammen, um Zuverlässigkeitsabweichungen zu analysieren.

Die Granularität der QA und der damit zusammen hängenden unerwünschten Zustände muss jeweils entsprechend dem gewünschten Zweck gewählt werden. Beispielsweise kann man sich vorstellen, Portabilität noch weiter aufzuschlüsseln nach den Aktivitäten, die zur Portierung gehören.

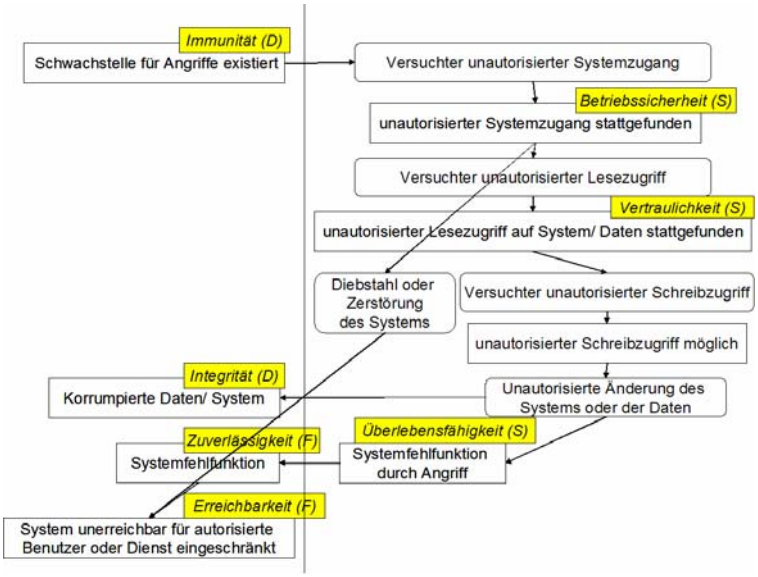

Abbildung 4: Szenario „security-relevant misuse“; Typ S (Sicherheitsverletzungen)

4 Nutzung der Abhängigkeiten

Unsere Motivation für die Herleitung dieser QA-Taxonomie und der Darstellung der Abhängigkeiten zwischen unerwünschten Zuständen bestand darin, dass wir Risiken (d.h. Wahrscheinlichkeit und Schaden) unerwünschter Zustände während des Requirements Engineerings (d.h. vor der Implementierung) quantifizieren wollten. Diese dienen uns als Grundlage für die Priorisierung von Software-Anforderungen, welche Qualitätsrisiken verringern [He07a]. Diese Vorhersage hatte sich in unserer Erfahrung als schwierig erwiesen. Existierende Vorhersagemethoden von Qualitätseigenschaften bzw. Qualitätsrisiken setzen zumeist eine Messung an einem existierenden System voraus, das zumindest ähnlich demjenigen ist, das man betrachtet [He07b].

Die Kenntnis der Abhängigkeiten zwischen unerwünschten Zuständen ersetzt zwar keine Messung, stellt aber ein Rahmenwerk dar, das es erlaubt, von der Erfüllung einer NFA auf eine andere zu schließen. Singh et al. [SCC01] beispielsweise ergänzen UML Use Case und Sequenzdiagramme mit Auftretenswahrscheinlichkeiten und Übergangswahrscheinlichkeiten und verwandeln so diese Diagramme in Bayes-Netzwerke. Dasselbe möchten wir auch mit unseren Aktivitätsdiagrammen tun, um ein System zu analysieren. Wenn unerwünschte Zustände aufeinander folgen, hängen auch deren Wahrscheinlichkeiten voneinander ab. So berechnet sich beispielsweise in einem vereinfachten Modell, in dem ein System entweder erreichbar ist oder aufgrund einer Fehlfunktion in Reparatur, die Wahrscheinlichkeit der Erreichbarkeit dieses Systems näherungsweise als $\rho/(\rho + \lambda)$ aus der Wahrscheinlichkeit λ einer Fehlfunktion und der Reparaturrate ρ, d.h. der Anzahl an Fehlfunktionen, die pro Zeiteinheit repariert werden [Vo00]. Ob analoge Abhängigkeiten für die Risiko- und Qualitätsvorhersage für alle QA verwendet werden können oder ob dies wegen der vielfältigen Abhängigkeiten zu kompliziert wird, wäre noch zu untersuchen. Gesucht werden dann: die Häufigkeiten der unerwünschten Zustände, die Übergangswahrscheinlichkeiten zwischen diesen sowie der jeweils zu erwartende Schaden.

Solche Werte stammen von Messungen ähnlichen Systemen oder an Teilsystemen. Die Leichtigkeit, mit der man diese Größen messen, schätzen oder Erfahrungswerte wieder verwenden kann, unterscheidet sich vermutlich für verschiedene QA. Daher ist es nützlich, wenn man von bekannten Werten auf unbekannte schließen kann, so dass sich die Menge der nötigen Werte verringert. Übergangswahrscheinlichkeiten zwischen Qualitätseigenschaften werden z.B. von [DK06] quantifiziert, die empirisch messen, wie in einem konkreten System die Produktivität, Safety und Zufriedenheit u.a. von der Bedienbarkeit, Angemessenheit und Genauigkeit abhängen. Bei diesem Experiment wurden Paare von QA identifiziert, die am stärksten miteinander korrelieren. Diese entsprechen in unserem Modell unerwünschten Zuständen, die direkt aufeinander folgen. Diese Übereinstimmung unabhängig gefundener empirischer Ergebnisse mit den Vorhersagen unseres durch Literaturrecherchen entwickelten Modells ist ermutigend.

Vorstellbar wäre auch, dass die Kenntnis der Abhängigkeiten zwischen den unerwünschten Zuständen noch auf folgende Weisen beim Requirements Engineering nützlich ist: Besitzt man Schätzwerte über die Wahrscheinlichkeit einiger unerwünschter Zustände, kann man auch realistische Vorgaben für die NFA davon abhängiger QA machen; die Spezifikation nur einer minimalen Menge an NFA wird unterstützt, wenn von direkt voneinander abhängigen QA nur eines betrachtet wird; Impakt-Analyse, d.h. Analyse der Auswirkungen, wenn eine bestimmte Qualitätseigenschaft geändert wird.

5 Zusammenfassung

In diesem Beitrag wird ein Ansatz aufgezeigt, der die Vielfalt der QA strukturiert und Abhängigkeiten zwischen ihnen darstellt, indem man jedem QA einen unerwünschten Zustand zuordnet und deren Abhängigkeiten darstellt.

Wie gut die quantitative Vorhersage von Qualitätsrisiken auf der Grundlage der in Abschnitt 3 dargestellten Abhängigkeiten funktioniert, muss noch untersucht werden. Eine Sammlung von Erfahrungswerten über Häufigkeiten und Schaden unerwünschter Zustände sowie von Übergangswahrscheinlichkeiten an einem bestimmten System wäre ein erster Schritt.

Literaturverzeichnis

[DK06] Dörr, J.; Kerkow, D.: Total control of User Experience in Software Development – a Software Engineering dream? In: Proc. 2nd COST294-MAUSE Int. Open Workshop, 2006; S. 94-99.

[Fi03] Firesmith, D.G.: Analyzing and Specifying Reusable Security Requirements. In: Proc. Requirements for High Assurance Systems Workshop RHAS, Monterey, 2003.

[He07a] Herrmann, A.: Priorisierung von Qualitätsanforderungen auf der Basis von Risikoabschätzungen. Software & Systems Quality Conf. Int., Düsseldorf, 2007.

[He07b] Herrmann, A.: Von Risiken zu Nutzen: Vorhersage von Risiken durch Schätzmethoden. Metrikon, Kaiserslautern, Germany, 2007, S. 179-199.

[IS91] International Standards Organization ISO: International Standard ISO/IEC 9126, Information technology - Software product evaluation - Quality characteristics and guidelines for their use. ISO, International Electrotechnical Commission, Geneva, 1991.

[Ma02] Madan, B.B.; Goševa-Popstojanova, K.; Vaidyanathan, K.; Trivedi, K.S.: Modeling and Quantification of Security Attributes of Software Systems. In: Proc. Int. Conf. On Dependable Systems and Networks DSN, June 2002.

[Oh84] Ohba, M.: Software Reliability Analysis Models. In: IBM J. of Res. and Development 28 (4) 1984; S. 428-443.

[SCC01] Singh, H.; Cortellessa, V.; Cukic, B.; Gunel, E.; Bharadwaj V.: A bayesian approach to reliability prediction and assessment of component based systems. In: Proc. 12th Int. Symposium on Software Reliability Engineering (ISSRE'01), 2001.

[SO00] Sindre, G.; Opdahl, A.L.: Eliciting Security Requirements by Misuse Cases. In: Proc. TOOLS Pacific 2000; S. 120-131.

[SO01] Sindre, G.; Opdahl, A.L.: Templates for Misuse Case Description. In: Proc. REFSQ - Workshop on Requirements Engineering for Software Quality, Foundations for Software Quality, Essener Informatik Beiträge Bd.6, Essen/ Germany, 2001; S. 125-136.

[Vo00] Vouk, M.A.: Software Reliability Engineering: In: 2000 Annual Reliability and Maintainability Symposium.