Testbare Anforderungen modellbasiert spezifizieren – Ergebnisse eines Studentenexperiments

Michael Felderer[1], Andrea Herrmann[2]
[1]Institut für Informatik, Universität Innsbruck, A-6020 Innsbruck, michael.felderer@uibk.ac.at
[2]Freiberufliche Trainerin für Software Engineering, D-70372 Stuttgart, herrmann@herrmann-ehrlich.de

Motivation: Warum ist die Testbarkeit von Anforderungen wichtig?

Die Testbarkeit von Anforderungen ist nicht nur ein wichtiges Qualitätskriterium laut IEEE Standard 830-1998 [1] und IREB [2], sondern in der Praxis sollen die Anforderungen ja tatsächlich als Grundlage für das Testen dienen. Beim modellbasierten Testen werden automatisiert oder manuell Testfälle aus den Anforderungsmodellen abgeleitet, aber auch textuelle Anforderungen dienen als Grundlage von Tests, z.B. für den manuellen System- und Abnahmetest. Sich bereits während der Anforderungsspezifikation Gedanken über die Testbarkeit zu machen, ist nicht verfrüht, sondern dient im Gegenteil der Qualitätssicherung der Anforderungen und dem Risikomanagement. Eine nicht testbare Anforderung ist auch keine eindeutige Grundlage für die Implementierung. Folglich erhöht eine nicht testbare Anforderung das Risiko, Code zu entwickeln, der so nicht gebraucht wird.

Wann ist ein Anforderungsmodell testbar?

Dieser Beitrag verfolgt die Frage, wie man die Testbarkeit von Anforderungen definieren und sicherstellen kann. Das Folgende beruht auf unserer Berufs- und Beratungserfahrung. Um empirisch zu explorieren, wie Anforderungsmodelle spezifiziert sein müssen, damit sie testbar sind, haben wir außerdem ein Studentenexperiment durchgeführt

Ein fachliches Modell ist dann testbar, wenn es alle Informationen enthält, welche für die Erstellung der Testfälle nötig sind. Für die Spezifikation der Testfälle verwenden wir – auch in industriellen Projekten – die in Abbildung 1 dargestellte Vorlage [3].

2	**Vorbedingung:** gültige Karte		
Nr.	**Aktion**	**Eingabedaten**	**Erwartetes Ergebnis**
1	Begrüßung		
2	Karte eingeben	Gültige Karte	
3	Karte prüfen		Karte erkannt

Abbildung 1: Testfall-Vorlage mit Anfang eines Testfalls für den Geldautomaten

Das Eckfeld links oben enthält eine Testfall-ID, hier 2, und im anschließenden Feld die Vorbedingung, welche vor Ausführung des Testfalls erfüllt sein muss. Jeder Testschritt entspricht einer nummerierten Zeile einschließlich auszuführender Aktion, Eingabedaten und erwartetem Ergebnis.

Die für das Testen im Anforderungsmodell nötigen Informationen lassen sich drei Kategorien zuordnen: Inhalte, Detailtiefe und Kontrollfluss.

Inhalte: Für die Erstellung eines Testfalls muss ein testbares Modell laut Abbildung 1 dessen Vorbedingung, Aktionen, Eingabedaten und erwartetes Ergebnis entweder ausdrücklich enthalten, oder diese müssen sich für den menschlichen Tester intuitiv schlüssig aus dem Modell ergeben. Was der Tester intuitiv ableiten kann und was nicht, ist bisher nicht erforscht. Wichtig ist jedoch Domänenwissen, d.h. die Kenntnis des Anwendungsbereichs der Software. Ohne solches Vorwissen kann der Testdesigner kaum gültige Schlüsse ziehen.

Detailtiefe: Hilfreich ist es, wenn das Modell dieselbe Detailtiefe wie die Testfälle hat, also insbesondere eine Aktivität (im Aktivitätsdiagramm) bzw. ein Zustandsübergang (im Zustandsdiagramm) genau einem Testschritt entspricht.

Kontrollfluss: Damit der Tester die für eine Testüberdeckung nötige Anzahl von Testfällen ableiten kann, müssen entweder alle Fehler- und Sonderfälle im Modell enthalten sein, oder er muss von dieser Unvollständigkeit wissen und selbständig prüfen, ob an jeder Stelle alle möglichen Fälle abgedeckt sind. Ggf. muss er noch Verzweigungen und Pfade ins Modell einfügen.

Durchführung des Studentenexperiments

Um konkret herauszufinden, wie viele und welche Fehler beim manuellen Ableiten von Testfällen auftreten, haben wir eine Untersuchung mit insgesamt 86 Studierenden an zwei Hochschulen im deutschsprachigen Raum durchgeführt. Studierende können durchaus repräsentativ sein für echte Testdesigner, da in der Praxis Tests oft von neuen Mitarbeitern (zur Einarbeitung) erstellt werden oder von Key Usern als Abnahmetest, die zwar die Anwendungsdomäne kennen, aber nur eine kurze Einführung in die Testfallerstellung erhalten haben.

Die 86 Teilnehmer/innen sollten jeweils aus einem Aktivitäts- und einem Zustandsdiagramm für einen Getränkeautomaten und einen Geldautomaten Testfälle ableiteten. Es handelte sich um eine ihnen vertraute Anwendungsdomäne. Im Experiment hatten die Modelle dieselbe Detailtiefe wie die abzuleitenden Systemtests. Die Sonderfälle hatten wir vollständig dargestellt. Informationen wie Vorbedingungen, Eingabedaten und erwartete Ergebnisse wurden nur teilweise ausdrücklich in den Modellen dargestellt, um zu prüfen, wie weit diese dann richtig ergänzt werden.

Ergebnisse des Experiments

Um die Vollständigkeit der Ergebnisse zu beurteilen definierten wir, dass alle Informationen, die tatsächlich für einen manuellen Test nötig wären, in dem Testfall enthalten sein müssen. Muss eine Karte eingeschoben oder über die Tasten eine bestimmte Eingabe gemacht werden, so muss dies auch im Testfall als Eingabe stehen. Nimmt der Testfall nur dann den geplanten Verlauf, wenn von Anfang an eine gültige Karte vorliegt oder der Geldbestand im Automaten einen bestimmten Wert hat, muss dies als Vorbedingung angegeben sein. Als Referenz erstellten wir Musterlösungen der Testfälle.

Unsere Testdesigner erstellten 150 Sätze von Testfällen, die insgesamt 342 Testfälle hätten enthalten sollen. Wir zählten darin insgesamt 1816 Fehler.

Es zeigte sich, dass öfter etwas vergessen wird als zu viel dargestellt:

- 724 Mal fehlte ein Inhalt im Testfall,
- 152 Mal war ein Feld mit unnötigem Inhalt befüllt,
- 695 Mal stand ein (richtiger) Inhalt im falschen Vorlagenfeld und
- 194 Mal war ein Feldinhalt ungültig.

Schließlich verglichen wir noch die Fehlerzahlen bei den Aktivitätsdiagrammen und Zustandsdiagrammen, um herauszufinden, welche der beiden Diagrammarten sich bei der Testfallableitung als verständlicher erwies. Es wurden bei der Ableitung von Testfällen aus Aktivitätsdiagrammen (statistisch signifikant) mehr Fehler gemacht als bei der Ableitung aus den Zustandsdiagrammen. Dies ist umso interessanter, als die Teilnehmer bei der Frage, wie verständlich sie das jeweilige Diagramm gefunden hatten, das Aktivitätsdiagramm (statistisch signifikant) als verständlicher bewertet hatten als das Zustandsdiagramm. Eine mögliche Erklärung für diesen scheinbaren Widerspruch könnte sein, dass die scheinbar intuitive Verständlichkeit und geringere Formalität des Aktivitätsdiagramms die Teilnehmer zu Flüchtigkeitsfehlern verleiteten, während das schwerer verständliche Zustandsdiagramm mehr Sorgfalt erzwang. Oder es könnten die weniger formalen Aktivitätsdiagramme auch weniger eindeutig sein als die Zustandsdiagramme.

Schlussfolgerungen

Zusätzlich zu theoretischen Überlegungen zur Testbarkeit von Anforderungsmodellen haben wir die Fehler untersucht, die unerfahrene Testdesigner bei der Testfallableitung tatsächlich machen. Die Testfall-ableitung hat sich dabei als eine Aufgabe erwiesen, die intuitiv nicht leicht korrekt durchzuführen ist. Insbesondere ist es wichtig, dass das Anforderungsmodell möglichst alle nötigen Informationen bereits enthält. Selbst bei Kenntnis der Anwendungsdomäne kann ein unerfahrener Testdesigner nicht unbedingt fehlende Informationen vollständig ergänzen.

Zum Weiterlesen…

In einem wissenschaftlichen Zeitschriftenartikel [4] können bei Interesse eine genauere Beschreibung des Experiments und seiner Ergebnisse nachgelesen werden. Unsere vorherigen Arbeiten zu ähnlichen Themen betreffen die Fehler, die bei der UML-Modellierung [5] und der Inspektion von Anforderungsdokumenten passieren [6], sowie Schwierigkeiten bei der Risikoschätzung [7], [8], [9], [10], [11], [12] und außerdem das Testen mit Fehlertaxonomien [3], [13], [14], [15].

Referenzen

[1] IEEE: IEEE Std 830-1998 Recommended Practice for Software Requirements Specifications, 1998

[2] K. Pohl, C. Rupp: Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB compliant. O'Reilly, 2011

[3] M. Felderer, A. Beer: Using Defect Taxonomies to Improve the Maturity of the System Test Process: Results from an Industrial Case Study. SWQD, 2013

[4] M. Felderer, A. Herrmann: Manual Test Case Derivation from UML Activity Diagrams and State Machines: a Controlled Experiment. Information and Software Technology, 2015, http://dx.doi.org/10.1016/j.infsof.2014.12.005

[5] A. Herrmann: Die häufigsten Modellierungsfehler. Symposium Software-Architektur, 2013, Stuttgart

[6] A. Herrmann: Lernen durch Feedback aus Inspektionen. Softwaretechnik-Trends, 34(1), 2014

[7] A. Herrmann, B. Paech: Practical Challenges of Requirements Prioritization Based on Risk Estimation. Journal of Empirical Software Engineering, 14(6), 2009, 644-674

[8] A. Herrmann: REFSQ 2011 Live Experiment about Risk-Based Requirements Prioritization: The Influence of Wording and Metrics. REFSQ, 2011

[9] A. Herrmann: Information Need of IT Risk Estimation – Qualitative Results from Experiments. RePriCo'11, 2011

[10] A. Herrmann: Risikobasierte Anforderungspriorisierung: Ergebnisse einer Experiment-Reihe. REConf, 2013

[11] M. Felderer, C. Haisjackl, R. Breu: Integrating Manual and Automatic Risk Assessment for Risk-Based Testing. SWQD, 2012

[12] M. Felderer, R. Ramler: Integrating risk-based testing in industrial test processes, Software Quality Journal, 22(3), 2014, 543-575

[13] M. Felderer, A. Beer: Using Defect Taxonomies for Testing Requirements, IEEE Software, 2014

[14] M. Felderer, A. Beer, B. Peischl: On the role of defect taxonomy types for testing requirements: Results of a controlled experiment, SEAA, 2014

[15] M. Felderer, A. Beer: Eine industriell erprobte Methode für den Review und Test von Anforderungen mit Hilfe von Fehlertaxonomien. Softwaretechnik-Trends, 34(1), 2014