Rückwärts- und Vorwärtsgerichtete Verfolgung von Fehlern für die modellbasierte Entwicklung eingebetteter Systeme

Stefan Miller
DaimlerChrysler Forschung und Technologie
Postfach 2360, 89013 Ulm
Stefan.Miller@DaimlerChrysler.com

Abstract: In diesem Beitrag wird die Notwendigkeit eines systematischen Prozesses zur Verfolgung von Fehlern begründet. Darüber hinaus werden für den Fall der modellbasierten Entwicklung eingebetteter Systeme zwei mögliche Prozesse, *rückwärts-* und *vorwärtsgerichtete Fehlerverfolgung*, zur Diskussion gestellt. Schließlich werden ihre Vor- und Nachteile aus Sicht der Praxis anhand eines Modells zur Fehlerfortpflanzung erläutert.

1 Einleitung

Über das Testen von Software hinaus ist es wichtig, dass die durch den Tester gefundenen Fehler systematisch verfolgt und korrigiert werden. Ziel ist es, Fehlerursachen zu lokalisieren und zu korrigieren, Folgefehler zu vermeiden und weitere, noch nicht entdeckte Fehler aufzudecken.

Im Gegensatz zu den Testprozessen, die mittlerweile in vielen Entwicklungsbereichen akzeptiert sind, existieren meistens keine systematischen Prozesse, um Fehler zu verfolgen und zu korrigieren. Es gibt zwar zahlreiche Dokumentationssysteme, wie zum Beispiel Mercury Quality Center [KSRR05], mit denen sich Fehler verwalten und dokumentieren lassen, dabei wird jedoch nur deren Korrektur überwacht, nicht aber unterstützt. Eine Ausnahme stellen Methoden dar, die verwendet werden, um die Ursachen der Fehler zu finden und die Fehler zukünftig zu vermeiden. Eine dieser Methoden ist Root Cause Analysis [LPS00]. Dabei wird versucht, die Wirkungskette eines Systems oder eines Vorgehens abzubilden, um bei unerwünschten Auswirkungen die Ursachen rückverfolgen und eliminieren zu können. Von einer Prozessunterstützung zur Fehlerkorrektur, die Aufwände und Aufgaben von Tester und Entwickler festlegt und koordiniert, kann aber nicht gesprochen werden. Root Cause Analysis kann aber innerhalb eines Prozesses zur Fehlerverfolgung verwendet werden, um die Entwickler bei der Ursachenverfolgung zu unterstützen.

Des Weiteren gibt es Forschungsansätze, die Fehlerdaten nutzen, um Aussagen über Entwicklungsprozesse selbst zu treffen und diese zu verbessern. Als Beispiele [FDK05] seien hier die Orthogonal Defect Classification (ODC), das HP-Schema oder die Fehlerflussmodelle von Freimut et al. genannt. Alle diese Arbeiten beschäftigen sich jedoch nicht mit der Korrektur und dem Umgang einzelner Fehler.

2 Rückwärts- und Vorwärtsgerichtete Fehlerverfolgung

Existiert heute in der Praxis ein Prozess zur Fehlerverfolgung, so sieht er in den meisten Fällen wie `P1` in Abbildung 1 beschrieben aus: Der Tester teilt seine gefundenen Fehler dem zuständigen Verantwortlichen in der Entwicklung mit. Dieser begibt sich nun auf die Fehlersuche und korrigiert, soweit er kann, den Fehler im Implementierungsmodell.

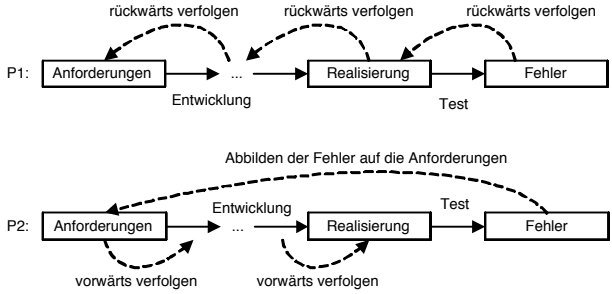

Abbildung 1: Rückwärts- (`P1`) und Vorwärtsgerichteter Prozess (`P2`) zur Fehlerverfolgung

In `P2` wird eine Vorgehensweise vorgestellt, die dazu führt, dass entdeckte Fehler zunächst dem Bereich gemeldet werden, der für die Beschreibung und Dokumentation der Anforderungen zuständig ist. Dessen Aufgabe ist es nun, den Fehler schrittweise an alle weiteren beteiligten Entwickler durch den Entwicklungsprozess durchzureichen, so dass der Fehler letztlich vollständig in allen Phasen der Entwicklung korrigiert werden kann.

3 Vergleich der beiden Prozesse

Die beiden im vorangegangenen Abschnitt vorgestellten Prozesse zur Fehlerverfolgung haben unterschiedliche Vor- und Nachteile.

Der rückwärtsgerichtete Prozess lebt stärker von den Fähigkeiten und der Zuverlässigkeit der einzelnen Entwickler, während der vorwärtsgerichtete Prozess etwas systematischer anwendbar ist. Zudem schreibt der vorwärtsgerichtete Prozess die Aktivitäten der Entwickler stärker vor und schränkt somit auch deren Flexibilität ein. Dennoch ist vor allem in großen Projekten eine systematische Verfolgung und Korrektur der Fehler notwendig. In kleineren Projekten kann das zu entstehende Produkt und dessen Schwachstellen besser überschaut und beurteilt werden. Hier können die Nachteile der rückwärtsgerichteten Verfolgung überwiegend durch eine enge Zusammenarbeit von Tester und Entwickler ausgeglichen werden. In größeren Projekten ist dies dagegen nicht so einfach möglich.

In Tabelle 1 werden die Vor- und Nachteile des vorwärtsgerichteten Prozesses im Vergleich zum rückwärtsgerichteten Prozess dargestellt. Die Spalten Aufwand und Qualität geben eine Bewertung der Vor- und Nachteile an. Positiv gekennzeichnete Einträge sind vorteilhafter für den vorwärtsgerichteten Prozess, negativ gekennzeichnete entsprechend

schlechter. Die Einträge in der Spalte Aufwand geben an, wieviel Arbeitsaufwand in die Korrektur der Fehler und des Systems insgesamt gesteckt werden muss. Unter Qualität wird verstanden, wie vollständig die Fehler korrigiert werden.

Merkmal	Benennung des Merkmals	Aufwand	Qualität
M1	zielgerichtetere Fehlerkorrektur	+	o
M2	unberechtigte Fehlermeldungen früher erkennbar	+	o
M3	Ausrichtung auf die Erfassung mehrerer Ursachen	++	+
M4	sichergestellte Korrektur bis in die Anforderungen	++	++
M5	Fehler und Anforderungen zwingend vergleichbar	- -	o
M6	stärkere Einbeziehung des Testers	-	o
M7	in Realisierung injizierte Fehler vernachlässigt	-	-

++ starker Vorteil, + Vorteil, o ausgeglichen, - Nachteil, - - starker Nachteil

Tabelle 1: Vor- und Nachteile des vorwärtsgerichteten Prozesses im Vergleich zum rückwärtsgerichteten Prozess

In der nachfolgenden Aufzählung werden die Merkmale M1 bis M7 der Tabelle 1 näher erläutert und begründet. Das Modell zur Fortpflanzung der Fehler in Abbildung 2 wird in dieser Aufzählung der Vor- und Nachteile verwendet, um sie besser veranschaulichen und erklären zu können.

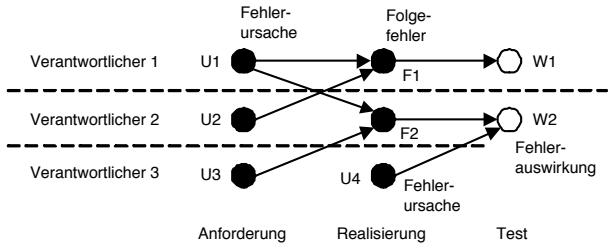

Abbildung 2: Modell für die Fortpflanzung von Fehler

Dargestellt in Abbildung 2 sind Fehlerursachen und deren Auswirkungen über die Erstellung der Anforderungen, Realisierung bis hin zum Testen. Es werden drei verschiedene Bereichsverantwortliche angegeben, die für die Anforderungen und für die Realisierung eines Teilsystems die Verantwortung tragen. Fehlerursachen sind mit `U`, Folgefehler mit `F` und die Auswirkungen der Fehler sind mit `W` gekennzeichnet. Die Pfeile stellen die Fortpflanzung der Fehler während der Entwicklung des Systems dar.

Nachfolgend werden die Merkmale der Tabelle 1 im Kontext der modellbasierten Entwicklung eingebetteter Systeme aus Sicht der Praxis diskutiert.

- *M1: zielgerichtetere Fehlerkorrektur.* Die Fehlerkorrektur in der Entwicklung kann im vorwärtsgerichteten im Vergleich zum rückwärtsgerichteten Prozess gezielter gesteuert werden, da die Korrekturarbeit durch die Abbildung der Fehler auf die An-

forderungen in der Regel bereits durch den Tester zugeordnet werden kann. Unklarheiten über die Zuständigkeit der Fehlerkorrektur können so reduziert werden. In Abbildung 2 hat der `Verantwortliche 2` nichts mit dem Fehler `W2` zu tun, obwohl der Fehler in einem Teilsystem gefunden wurde, der in seinen Zuständigkeitsbereich fällt. Durch die hohe Zahl der Vernetzung in eingebetteten Systemen treten solche Fälle auf. Häufig kann nicht klar festgelegt werden, wer mit der Korrektur des Fehlers betreut werden muss.

- *M2: unberechtigte Fehlermeldungen früher erkennbar.* Durch die Abbildung der Fehler auf die Anforderungen können fälschlicherweise gemeldete Fehler teilweise früher erkannt werden. Häufig ist im Detail unklar wie sich das System verhalten muss, was dazu führt, dass Fehler gemeldet werden, obwohl die Umsetzung korrekt ist. Durch die stärkere Auseinandersetzung mit den Anforderungen können solche Nichtfehler zum Teil erkannt werden.

- *M3: Ausrichtung auf die Erfassung mehrerer Ursachen.* Beim rückwärtsgerichteten Prozess wird häufig im ersten Schritt, falls mehrere Fehlerursachen existieren, nur eine der Ursachen erkannt und korrigiert. Durch die Abbildung der Fehler auf die Anforderungen können mehrere mögliche Ursachen erkannt werden. So sind in Abbildung 2 für das Fehlverhalten `W2` die Ursachen `U1`, `U3` und `U4` verantwortlich. Durch verfolgen bis hin zu `U1` kann ein weiteres Fehlverhalten `W1` entdeckt werden, das die Notwendigkeit der Korrektur von Folgefehler `F1` offen legt.

- *M4: sichergestellte Korrektur bis in die Anforderungen.* Durch die Abbildung der Fehler auf die Anforderungen wird die Fehlerursache in den ersten Arbeitsergebnissen der Entwicklungskette identifiziert und dann schrittweise durch den Entwicklungsprozess verfolgt. Dies stellt sicher, dass die Fehlerursachen in allen Entwicklungsergebnissen eliminiert werden. Insbesondere wenn die Anforderungsdokumente in zukünftigen Projekten wieder verwendet werden, ist eine Korrektur der Fehler unerlässlich. Bei der rückwärtsgerichteten Fehlersuche wird aus Zeitmangel häufig auf eine Verfolgung rückwärts durch die Prozesskette bis zu den Anforderungen verzichtet. Vor allem bei komplexen und großen Projekten ist es äußerst fraglich, ob es sinnvoll ist auf diese Weise Zeit einzusparen. In Abbildung 2 müssen, wenn Fehlerauswirkung `W1` entdeckt wird, `U1`, `U2` und `F1` korrigiert werden. Falls das Teilsystem des `Verantwortlichen 2` noch nicht umgesetzt wurde, könnte bei der weiteren Fortentwicklung des Systems `F2`, und damit auch `W2`, vermieden werden.

- *M5: Fehler und Anforderungen zwingend vergleichbar.* Die Abbildung der Fehler auf die Anforderungen erfordert Techniken und Methoden, die dem Tester bereitgestellt werden müssen. Idealerweise liegen die Anforderungen formal und maschinenlesbar vor, so dass der Tester seine entdeckten Fehler ebenfalls formalisieren kann und die Abbildung der Fehler auf die Anforderungen automatisiert vorgenommen wird. Eine wichtige Voraussetzung für eine Formalisierung ist jedoch, dass sie praktikabel und effizient durchgeführt werden kann.

- *M6: stärkere Einbeziehung des Testers.* Der Tester wird im vorwärtsgerichteten Prozess stärker eingebunden. Es müssen daher hier mehr Ressourcen zur Verfügung ge-

stellt werden. In anderen Bereichen können eventuell jedoch Kapazitäten eingespart werden, da die Korrektur der Fehler schneller durchgeführt werden kann.

- *M7: in Realisierung injizierte Fehler vernachlässigt.* Durch eine zu starke Fokussierung auf die Anforderungen könnten während der Realisierungsphase injizierte Fehlerursachen übersehen werden. So könnte Fehlerursache `U4` in Abbildung 2 nicht entdeckt werden, falls sich die Entwickler oder Tester bei der Ursachensuche ausschließlich auf die Anforderungen fixieren.

4 Ausblick

Die in der Tabelle 1 angegebenen Vor- und Nachteile mit den Bewertungen von - - bis ++ spiegeln Erfahrungen aus der Praxis modellbasierter Entwicklung wider. Diese Annahmen sollten durch empirische Untersuchungen bestätigt werden. Wir beabsichtigen in einem Experiment im Umfeld modellbasierter Entwicklung eingebetteter Systeme diese Annahmen zu validieren. Darüber hinaus soll eine Methodik entwickelt und geprüft werden, die den Nachteil M5 des vorwärtsgerichteten Prozsses abschwächt. Der Kontext des modellbasierten Entwickelns kann hier genutzt werden, Fehler und Anforderungen zumindest teilweise zu formalisieren und deren Abbildung aufeinander algorithmisch zu unterstützen. Würden sich die einzelnen Merkmale von Tabelle 1 bestätigen sowie die Methodik zur Abbildung der Fehler auf die Anforderung als praktikabel erweisen, so würde der vorwärtsgerichtete gegenüber dem rückwärtsgerichteten Prozess klare Vorteile hinsichtlich Entwicklungsaufwand und Produktqualität bieten.

5 Danksagung

Mein Dank gilt Bela Mutschler, Ramin Tavakoli und Peter Manhart (DaimlerChrysler) sowie Prof. Dr. Franz Schweiggert (Universität Ulm, Abteilung Angewandte Informationsverarbeitung) für die hilfreichen Verbesserungsvorschläge zu diesem Beitrag.

Literatur

[FDK05] Bernd G. Freimut, Christian Denger und Markus Ketterer. An Industrial Case Study of Implementing and Validating Defect Classification for Process Improvement and Quality Management. In *IEEE METRICS*, Seite 19. IEEE Computer Society, 2005.

[KSRR05] Thomas Konschak, Thomas Sußebach, Peter Rissling und Rainer Rasche. Testprozesse bei der BMW Group. *Hanser Automotive*, Seiten 46–50, Oktober 2005.

[LPS00] Marek Leszak, Dewayne E. Perry und Dieter Stoll. A case study in root cause defect analysis. In *Proceedings of the 22nd International Conference on Software Engineering*, Seiten 428–437. ACM Press, Juni 2000.