Management des Testprozesses von Anfang an

Das W-Modell

Andreas Spillner

Hochschule Bremen
Zentrum für Informatik und Medientechnologien
Flughafenallee 10, 28199 Bremen

Abstract: In software development, 30 to 40 % of all software activities are testing related. That is why it is critical to launch testing activities at the beginning of the project rather than after the coding is completed. While new software development models such as the Rational Unified Process and eXtreme Programming (XP) continue to be popular with practitioners, the V-model has gained particularly wide acceptance. Based on the V-model, this paper describes a model that shows how the tasks for testing relate to the tasks in the development model. This model – the W-model – further clarifies the priority of the tasks and the dependence between the development and testing activities. Although as simple as the V-model, the W-model makes the importance of testing and the ordering of the individual testing activities clear. It also clarifies that testing and debugging are not the same thing.

1 Prozessmodelle der Softwareentwicklung

Es gibt eine ganze Reihe von unterschiedlichen Prozessmodellen, die dazu beitragen eine strukturierte und steuerbare Softwareentwicklung durchzuführen. Beispiele hierfür sind das Wasserfall-Modell [Bo73], das Allgemeine V-Modell[1] [Bo79] und das Vorgehensmodell des Bundes und der Länder [Ve99]. Aktuell diskutiert werden der Rational Unified Process [Ja99] und eXtreme Programming [Be00].

Alle Modelle definieren eine Systematik zur geordneten Projektabwicklung. Meist werden so genannte Phasen bzw. Arbeitsabschnitte festgelegt, die mit einem definierten Ergebnis in Form eines Dokumentes abzuschließen sind. Der Abschluss, oft auch als Meilenstein bezeichnet, ist dann erreicht, wenn die erstellten Dokumente den geforderten Qualitätskriterien genügen, andernfalls sind Korrekturarbeiten vorzunehmen. In den Modellen werden Rollen beschrieben, denen bestimmte anfallende Aufgaben in der Softwareentwicklung zugeordnet werden und die von den an der Projektentwicklung beteiligten Personen auszufüllen sind. Teilweise werden auch einzusetzenden Methoden und Verfahren in den jeweiligen Phasen beschrieben. Mit Hilfe der Modelle kann eine

[1] Das V-Modell wird mit dem Zusatz „Allgemeines“ versehen, um es nicht mit dem „Vorgehensmodell des Bundes und der Länder“ zu verwechseln, da dieses Vorgehensmodell in der Literatur auch häufig kurz als „V-Modell“ bezeichnet wird.

detaillierte Projektplanung der Ressourcen, wie Zeit, Personal, und benötigte Infrastruktur usw. erfolgen. Die Entwicklungsmodelle definieren die für alle Beteiligten gemeinsame und verbindliche Sicht der logischen und zeitlichen Struktur eines Projekts.

Qualitätssicherungsmaßnahmen und speziell das Testen findet sich in jedem der Vorgehensmodelle wieder, allerdings mit sehr unterschiedlicher Bedeutung und mit unterschiedlichem Umfang. Im Folgenden wird aus Sicht des Testens auf das Wasserfall- und das Allgemeine V-Modell kurz eingegangen. Anschließend wird eine Erweiterung des Allgemeinen V-Modells vorgestellt, die neben dem Prozess der Softwareentwicklung einen zweiten parallelen Prozess vorsieht, der alle Testaktivitäten umfasst und mit dem Start des Projekts beginnt. In dem so erweiterten Modell werden die Bedeutung, der Zeitpunkt der Durchführung und der verbundene Aufwand der einzelnen Testaktivitäten deutlich. Testen und Debugging werden als getrennte Aufgaben mit ihren zyklischen Abhängigkeiten dargestellt. Am Ende des Beitrags wird eine Bewertung des Modells vorgenommen.

2 Vorgehensmodelle der Softwareentwicklung

2.1 Das Wasserfall-Modell

Ausgangspunkt aller gängigen Prozessmodelle ist das Wasserfall-Modell aus den 70er Jahren ([Bo73]). Der Entwicklungsprozess wird in einzelne Phasen eingeteilt, die jeweils abgeschlossen sein sollen, bevor mit den Arbeiten in der nächsten Phase begonnen wird. Am Ende jeder Phase wird eine Validierung der Phasenergebnisse vorgenommen. Nacharbeiten sind bei unzureichender Qualität der Dokumente vorzunehmen. Eine verbreitete Phaseneinteilung ist die folgende: Anforderungsdefinition, Systementwurf, Komponentenspezifikation, Programmierung, Test und Inbetriebnahme.

Die wichtigsten Kennzeichen des Wasserfall-Modells sind:

- Es ist ein sehr einfaches und einsichtiges Modell, das die einzelnen Aktivitäten der Softwareentwicklung klar voneinander trennt.
- Ein Übergang ist nur zur nächstfolgenden Phase möglich, nach dem eine Prüfung der Ergebnisse vorgenommen wurde.
- Bei nicht zufrieden stellendem Ergebnis der Prüfung ist ein Rückschritt nur auf die direkt vorhergehende Phase möglich.

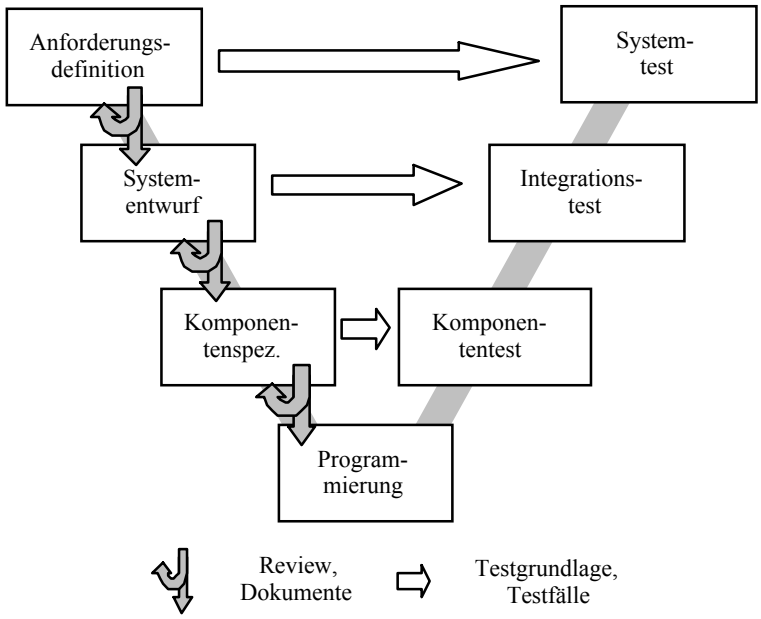
Nach dem Modell wird mit den Überlegungen zum Testen erst begonnen, wenn die Implementierung abgeschlossen ist. Testen ist somit den „späten Phasen“ der Softwareentwicklung zuzuordnen. Eine verhängnisvolle Sicht, die in vielen Projekten dazu geführt hat, dass aus Zeitgründen kein ausreichender Test mehr durchgeführt wurde, es liegt ja ein ausführbares System vor und der Kunde wird die Fehler schon finden und melden.

2.2 Das Allgemeine V-Modell

Neben weiteren Überlegungen zu Prozessmodellen, z. B. dem Spiral-Modell oder anderen iterativen Modellen, wurde Ende der 70er Jahre das Allgemeine V-Modell vorgestellt ([Bo79]). Ähnlich wie bei dem Wasserfall-Modell existieren inzwischen auch vom Allgemeinem V-Modell zahlreiche unterschiedliche Varianten. Benennung und Anzahl der

Phasen variieren je nach Literaturquelle oder Interpretation der anwendenden Firma. Die Grundidee des Allgemeinen V-Modells ist, dass Entwicklungsarbeiten und Testarbeiten zueinander korrespondierende, gleichberechtigte Tätigkeiten sind. Bildlich dargestellt wird dies durch die zwei Äste eines „V"s (s. Bild 1).

Bild 1: Eine Version des Allgemeinen V-Modells



Der linke Ast steht für die immer detaillierter werdenden Entwicklungsschritte, in deren Verlauf das vom Kunden gewünschte System schrittweise entworfen und schließlich programmiert wird. Der rechte Ast steht für Integrations- und Testarbeiten, in deren Verlauf elementare Programmbausteine sukzessive zu größeren Teilsystemen integriert und jeweils auf ihre korrekte Funktionalität hin geprüft werden. Integration und Test enden mit der Abnahmeprüfung des entstandenen Gesamtsystems.

Die konstruktiven Aktivitäten auf der linken Seite sind bereits aus dem Wasserfall-Modell bekannt:

- **Anforderungsdefinition**: Die Wünsche und Anforderung des Auftraggebers oder späteren Systemanwenders werden gesammelt, strukturiert, geprüft und verabschiedet. Zweck und gewünschte Leistungsmerkmale des zu entwickelnden Softwaresystems werden in dieser Phase festgelegt.
- **Systementwurf**: Die technische Realisierung des Systems wird grob entworfen. Die Schnittstellen zur Systemumwelt werden definiert und die Zerlegung des Systems in überschaubarere Teilsysteme, die möglichst unabhängig voneinander entwickelt werden können, wird konzipiert.

- **Komponentenspezifikation**: Für jedes Teilsystem werden Aufgabe, Verhalten, innerer Aufbau und Schnittstelle zu anderen Teilsystemen festgelegt.
- **Programmierung**: Implementierung jedes spezifizierten Bausteins (Modul, Unit, Klasse o.ä.) in einer Programmiersprache.

Entlang dieser Konstruktionsphasen wird das zu entwickelnde Softwaresystem zunehmend detaillierter beschrieben. Dabei wird spätestens vor dem Übergang zur nächsten Aktivität eine Prüfung (Review) der erstellten Dokumente vorgenommen. Die rechte aufsteigende Seite des Allgemeinen V-Modells ordnet jedem Spezifikations- bzw. Konstruktionsschritt eine korrespondierende Testphase[2] zu:

- Komponententest prüft, ob jeder einzelne Software-Baustein separat die Vorgaben der Komponentenspezifikation erfüllt.
- **Integrationstest** überprüft, ob Gruppen von Komponenten wie im Systementwurf spezifiziert zusammenspielen.
- **Systemtest** kontrolliert, ob das System als Ganzes die definierten Kundenanforderungen erfüllt.

Aufgabe jeder Testphase ist die Prüfung, ob die in der Phase Programmierung entstandenen Programmteile diejenigen Anforderungen erfüllen, die auf der jeweiligen Abstraktionsstufe relevant bzw. spezifiziert sind.

Die wichtigsten Kennzeichen des Allgemeinen V-Modells sind:

- Konstruktive und prüfende Entwicklungsaktivitäten sind getrennt aber gleichberechtigt (linke und rechte Seite des „V“s).
- Die rechte Seite kann auch als Folge von Integrationsschritten (Komponente, Teilsysteme, System) gesehen werden.
- Es werden arbeitsteilige Testphasen (Komponenten-, Integrations-, Systemtest) unterschieden, wobei jede Phase „gegen“ ihre korrespondierende Entwicklungsphase testet. Im Bild 1 durch die Pfeile von links nach rechts dargestellt.

Mit dem Allgemeinem V-Modell sind folgende Nachteile verbunden. Die Testaktivitäten scheinen auch in diesem Modell erst nach der Programmierung zu beginnen, vorbereitende Aktivitäten zum Testen werden nicht deutlich. Oft wird das Modell so interpretiert, das zwischen den konstruktiven Aufgaben und den prüfenden Aufgaben keine Querbezüge bestehen bzw. bestehen sollten. Test und Entwicklung werden als strikt voneinander zu trennende Aufgaben gesehen (s.a. [SL02]).

Im nächsten Kapitel wird ein Modell vorgestellt, das diese Nachteile behebt. Die Grundidee dazu ist nicht neu. Viele Erweiterungen des Allgemeinen V-Modells zielen in eine ähnliche Richtung (s. beispielsweise [Vo01]).

3 Das W-Modell

Im W-Modell werden die Testaktivitäten als paralleler Prozess zur Entwicklung der Software betrachtet [Sp00, Sp02]. Bei anderen Erweiterungen des Allgemeinen V-Modells wird meist nur die linke, absteigende Seite des Modells bis zur Programmierung um

[2] oft auch mit Teststufe bezeichnet

die entsprechenden vorbereitenden Testaktivitäten ergänzt. Alle Aktivitäten, die vor der Ausführung der Testfälle in den unterschiedlichen Testphasen durchzuführen sind, können und sollen bereits parallel zu den frühen Entwicklungsphasen bearbeitet werden. Bei der im W-Modell vorgestellten Erweiterung wird auch die rechte, aufsteigende Seite, die Testphasen, um Debug- und Änderungs-Aktivitäten erweitert. Deutlich wird, dass Test und Debugging unterschiedliche Aktivitäten sind und es zyklische Abhängigkeiten zwischen ihnen gibt. Es existiert sozusagen ein zweites, parallel verlaufendes V. Mit dem Namen W-Modell soll dies verdeutlicht werden (s. Bild 2).

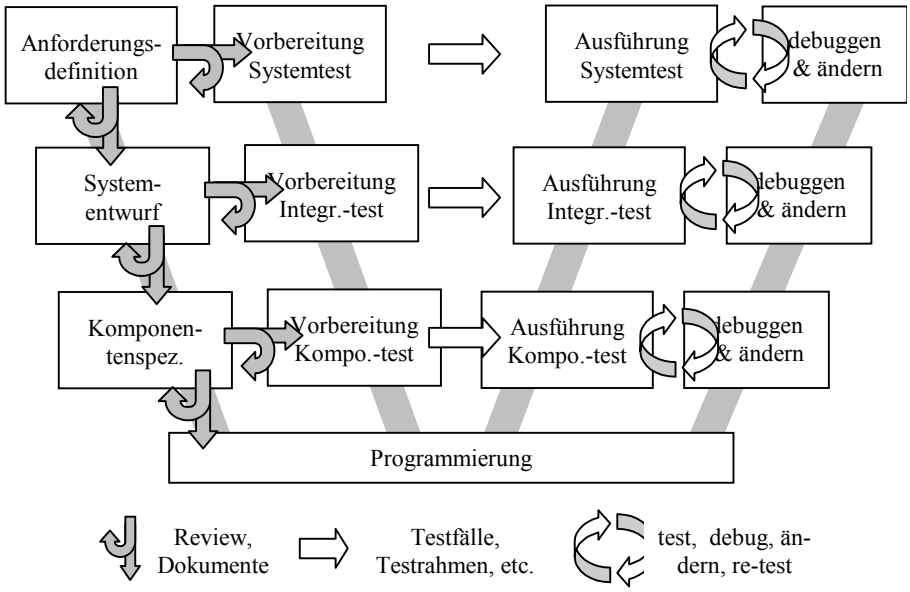
3.1 Grundlegende Konzepte des W-Modells

3.1.1 Testprozess-Management parallel zur Entwicklung

Eine strukturierte Abwicklung einer so umfangreichen Aufgabe wie dem Testen kann nicht planlos erfolgen. Mit der Planung des Testprozesses wird zum Beginn des Softwareentwicklungsprojektes begonnen. Wie bei jeder Planung ist während des Projektfortschritts eine Aktualisierung und Anpassung der Planung vorzunehmen. Eine detaillierte Planung ist für jede Testphase durchzuführen.

Für den gesamten Testprozess sind die benötigten Ressourcen einzuplanen: Wann werden welche Mitarbeiter zur Durchführung der Aufgaben benötigt? Wie viel Zeit ist dafür zu veranschlagen und welche Hilfsmittel und Geräte sind dafür bereitzustellen? Dies sind Fragen die vorab geklärt werden müssen. Werkzeuge zur Unterstützung der Testaktivitäten sind genauso wie entsprechende Hardware-Ressourcen einzuplanen. Die entsprechenden Festlegungen sind im Testplan festzuhalten. Testteams sind zusammenzustellen und mögliche notwendige Schulungen der Mitarbeiter sind vorzusehen. Eine Organisationsstruktur mit dem entsprechenden Management ist ggf. aufzubauen bzw. anzupassen. Zu den Aufgaben des Managements gehört die Verwaltung des Testprozesses, der Testinfrastruktur und der Testprodukte. Die Überwachung des Fortschritts im Testprozess kann auf Grundlage der entsprechenden Berichterstattung der Mitarbeiter erfolgen.

Bild 2: Das W-Modell



Kernaufgabe der Planung ist die Bestimmung der Teststrategie. Ein vollständiger Test, der alle Möglichkeiten der Ausführung des Testobjektes umfasst, ist nicht möglich. Deshalb müssen Prioritäten anhand einer Risikoeinschätzung gesetzt werden. Je nach zu erwartendem Risiko im Fehlerfall sind die Testaktivitäten auf die einzelnen Systemteile zu verteilen. Kritische Systemteile müssen eine erhöhte Aufmerksamkeit im Test erfahren, also intensiver getestet werden. Bei den weniger kritischen Teilen reicht ein weniger umfangreicher Test aus. Ziel der Teststrategie ist die optimale Verteilung der Tests auf die „richtigen“ Stellen des Softwaresystems.

Es sind die einzusetzenden Testmethoden auszuwählen. Der zu erreichende Deckungsgrad bei der Testausführung ist festzulegen, um ein Kriterium für das Testende zu definieren. Neben den Abdeckungskriterien, die sich auf den Programmtext beziehen (z. B. auf Anweisungen oder Verzweigungen), kann die Erfüllung der Anforderungen des Kunden als Endekriterium festgelegt werden. Es kann gefordert sein, dass alle Funktionen mindestens einmal getestet werden oder beispielsweise 70% der möglichen Transaktionen in einem System einmal zur Ausführung kommen. Selbstverständlich sind auch die Festlegungen der Endekriterien und damit die Intensität der Tests unter Berücksichtigung des im Fehlerfall zu erwartenden Risikos zu treffen.

Da Softwareprojekte oft unter hohem Zeitdruck abgewickelt werden, ist es sinnvoll, den Zeitaspekt in der Planung entsprechend zu berücksichtigen. Die Priorisierung der Tests garantiert, dass die kritischen Softwareteile zuerst getestet werden, falls aus Zeitgründen nicht alle geplanten Tests durchgeführt werden können. Alle Fakten zusammen sind Grundlage für eine erste Abschätzung der zu veranschlagenden Testressourcen. Ausführliche Beschreibungen zum Testprozess finden sich in [Po02] und [SL02].

3.1.2 Reviews unter Beteiligung der Tester

Im W-Modell werden die Tester frühzeitig und nicht erst nach der Programmierung an der Entwicklung beteiligt. Die Ergebnisse der jeweiligen Phasen werden Reviews unterzogen, um deren Qualität zu prüfen und möglichst frühzeitig Fehler und Unstimmigkeiten aufzudecken. In den Reviews wird kontrolliert, ob alle Anforderungen der Phase soweit erfüllt sind und den Qualitätsanforderungen genügen, damit das untersuchte Dokument als Basis für die Arbeiten der nächsten Phase verwendet werden kann. Meist findet eine intensive Diskussion in den Reviewsitzungen statt.

Im W-Modell werden die Reviews zusätzlich dazu genutzt zu diskutieren, wie gut die Dokumente sich als Grundlage für die korrespondierende Testphase eignen. Aus Sicht des Tests werden andere Aspekte betrachtet als aus der Sicht der Entwicklung. Zusätzliche Punkte werden diskutiert und geklärt. Auch die direkten Auswirkungen auf den Test werden in Betracht gezogen. Führt beispielsweise eine vorgeschlagene Systemarchitektur zu einem erheblichen Testaufwand, so soll dies zu diesem frühen Zeitpunkt thematisiert werden. Möglicherweise kann der Testaufwand durch die Wahl einer einfacheren Architektur verringert werden.

Die Dokumente, die den Reviews unterzogen werden, dienen als Grundlage für die in den korrespondierenden Phasen durchzuführenden Tests. Es liegt somit nahe, alle testvorbereitenden Aufgaben bereits zu diesem Zeitpunkt durchzuführen. Die Tester haben sich in der Vorbereitung auf die Reviews mit der Materie vertraut gemacht und das notwendige Wissen zur Testvorbereitung ist somit vorhanden. Alle vorbereitenden Testarbeiten sind noch vor der Implementierung abgeschlossen.

Auch diese Vorbereitung wird weitere Fragen aufwerfen, die zu diesem frühen Zeitpunkt geklärt werden und zur Verbesserung der weiter verwendeten Dokumente beitragen, bevor mit dem unzureichenden Dokument in der nächsten Phase weitergearbeitet wird. Eine Diskussion zwischen Entwickler und Tester ist erwünscht und hilfreich, da die unterschiedliche Sichtweise der beiden Gruppen auf die Dokumente zur Verbesserung der Qualität der Dokumente beiträgt.

3.1.3 Testen und Debugging sind unterschiedliche Aufgaben

Die rechte Seite des Allgemeinen V-Modells wird oft als reine Testaktivität gesehen, was allerdings nicht der Fall ist. Um dies zu verdeutlichen, ist im W-Modell auch der rechte Ast des „V“s aufgeteilt worden in Aktivitäten, die von Testern durchzuführen sind und in solche, die von den Entwicklern zu tätigen sind. Da alle vorbereitenden Aufgaben zum Testen parallel zur Entwicklung durchgeführt werden, sind nach der Programmierung nur noch die vorbereiteten Testfälle auszuführen. Nachdem Fehler gefunden werden, sind die Ursachen der Fehler zu lokalisieren (Debugging) und zu beseitigen. Nach deren Beseitigung sind Tests zu wiederholen, um festzustellen, ob die Fehler beseitigt und keine neuen hinzugefügt wurden.

Oft werden Test und Debugging nicht strikt auseinander gehalten, manchmal sogar verwechselt oder das Debugging, die Fehlerursachenfindung, als Testen angesehen. Im W-Modell wird die Unterscheidung hervorgehoben. Die auftretenden Zyklen (testen, debuggen, ändern, erneut testen) und der damit verbundene Aufwand werden in den meisten anderen Prozessmodellen nicht deutlich.

Durch die Trennung in zwei parallele Prozesse wird darüber hinaus klar, dass unterschiedliche Personengruppen beteiligt sind. Die Änderungen am Programmtext sind von den Entwicklern durchzuführen. Testen ist Aufgabe der Tester. Auch hier ist eine enge kooperative Zusammenarbeit zwischen den beiden Gruppen hilfreich.

3.2 Aktivitäten im W-Modell

Im Folgenden werden die einzelnen Aktivitäten erörtert, die im W-Modell im Vergleich zum Allgemeinem V-Modell neu hinzugekommen sind. Dabei wird auch der zeitliche Ablauf deutlich.

3.2.1 Vorbereitung Systemtest

Bei den Reviews der Anforderungsdokumente sind von Seiten der Tester die Aspekte zu betrachten, die für den Systemtest von Bedeutung sind. Es ist darauf zu achten, dass jede Anforderung messbar und damit testbar formuliert wird. Dies trifft besonders für die nichtfunktionalen Anforderungen zu. Formulierungen wie „das System soll leicht bedienbar sein“ oder „schnell reagieren“ sind in dieser Form nicht testbar.

Folgende Fragen sind beispielsweise im Review der Anforderungsdokumente zu klären:

- Sind die Angaben präzise genug, um Systemtestfälle spezifizieren zu können?
- Ist der erforderliche Testaufwand den beteiligten Personen klar? Ggf. lässt sich der Testaufwand durch Änderungen der Anforderungen verringern.
- Sind alle Randbedingungen für den Einsatz des Systems definiert? Diese Randbedingungen müssen auch für den Systemtest gelten.

Sind die Anforderung nur vage formuliert und es lassen sich keine konkreten Testfälle ableitet, dann reichen die Angaben auch nicht aus, um eine konkrete Realisierung der Anforderungen vorzunehmen. Eine Präzisierung in Zusammenarbeit mit dem Kunden ist erforderlich.

Neben der konkreteren Planung und Vorbereitung der Systemtests werden die funktionalen und die nicht funktionalen Systemtestfälle auf Grundlage der Anforderungsdokumente spezifiziert. Durch die Beteiligung an den Reviews sind die Tester in die Materie eingearbeitet. Werden die Testfälle kurz vor deren Ausführung spezifiziert, also kurz vor Abschluss des Projektes, arbeiten sich die Tester erst sehr spät in die Anforderungsdokumente ein. Eine dann stattfindende Diskussion über unzureichende Angaben in den Dokumenten zwischen Testern und Entwicklern ist mühevoll, da die Entwickler meist mit anderen Aufgaben beschäftigt sind.

Eine Priorisierung der Systemtestfälle wird vorgenommen. In vielen Projekten ist trotz ausreichender Planung die Zeit zur Durchführung aller spezifizierten Testfälle nicht vorhanden. Es ist dann eine sinnvolle Auswahl der Testfälle zu treffen, um sicherzustellen, dass trotz der eingeschränkten Anzahl an durchzuführenden Testfällen noch möglichst viele kritische Fehler gefunden werden. Die Priorisierung soll so erfolgen, dass bei einer vorzeitigen Beendigung der Tests zu einem beliebigen Zeitpunkt das bis dahin bestmögliche Testergebnis erreicht wird.

Ist für die Ausführung der Systemtestfälle eine entsprechende Infrastruktur notwendig, so ist diese bereitzustellen. Ist beispielsweise ein Datenbanksystem notwendig, dann ist die-

ses für den Systemtest mit den entsprechenden Einträgen zur Verfügung zu stellen. Diese Arbeiten sind zeitintensiv und sollten frühzeitig begonnen werden.

Mit Abschluss der Arbeiten an der Anforderungsdefinition sind auch alle vorbereitenden Aufgaben für den Systemtest beendet, so dass am Ende des Projektes nach erfolgreicher Integration der Teilsysteme der Systemtest ohne Verzögerungen ausgeführt werden kann.

3.2.2 Vorbereitung Integrationstest

Beim Review der Systemarchitektur sind die Auswirkungen auf den Integrationstest zu berücksichtigen. Es ist darauf zu achten, dass die Schnittstellen zwischen den Teilsystemen und zur Systemumgebung präzise definiert und die Abhängigkeiten zwischen den Teilsystemen möglichst gering sind.

Folgende Fragen sind in der Reviewsitzung der Systemarchitektur von den Testern zu stellen und von allen verbindlich zu klären:

- Sind die Schnittstellen präzise genug spezifiziert, um Integrationstestfälle daraus ableiten zu können?
- Sind die Schnittstellen präzise genug spezifiziert, um einen Testrahmen zur Simulation der Umgebung des Teilsystems realisieren zu können?
- Lässt sich der erforderliche Testaufwand durch Änderung der Architektur verringern?

Sind zu viele Abhängigkeiten zwischen den Systemteilen vorhanden, so sind viele unterschiedliche Testfälle zu spezifizieren, um die Abhängigkeiten zu testen. Auch ist dann beim separaten Test der Teile ein hoher Simulationsaufwand für die Umgebung notwendig. Komplexe Schnittstellen sind meist fehleranfällig, so dass sich eine Vereinfachung in dieser Hinsicht lohnt.

Die Integrationstestfälle sind auf Grundlage der Systemarchitektur nach Auswahl von entsprechenden Integrationstestmethoden zu spezifizieren. Auch hier sind die Tester durch das Review in die Materie bereits eingearbeitet.

Eine Priorisierung der Integrationstestfälle wird ebenfalls vorgenommen. Besonders kritische Systemteile, die bei einem Fehlverhalten ein hohes Risiko beinhalten, z.B. die Gefährdung von Menschenleben, sollten eine hohe Priorität erhalten und damit intensiv getestet werden. Insbesondere komplexe Schnittstellen sollten intensiv geprüft werden.

Für den Test der Teilsysteme sind Testrahmen vorzusehen, welche die noch nicht integrierten Systemteile simulieren. Diese Testrahmen können bereits zu diesem Zeitpunkt erstellt werden.

Alle vorbereitenden Aufgaben für den Integrationstest sind abgeschlossen, so dass nach erfolgreichem Test der einzelnen Komponenten der Integrationstest ausgeführt werden kann.

3.2.3 Vorbereitung Komponententest

Die Tester berücksichtigen bei den Reviews der einzelnen Bausteine des Systems die Auswirkungen auf den Komponententest. Die „inneren“ Schnittstellen zwischen den

Komponenten müssen präzise definiert sein und möglichst geringe Abhängigkeiten untereinander haben.

Die Fragen der Tester auf den entsprechenden Reviewsitzung zur Spezifikation der Komponenten sind analog zu den Fragen zur Systemarchitektur zu sehen, nur dass die Aufteilung der Teilsysteme in Komponenten diskutiert wird. Eine Überarbeitung der gewählten Aufteilung sollte auch auf dieser Ebene vorgenommen werden, wenn sich für den Komponententest ein hoher Aufwand abzeichnet.

Da die Tester sich die Spezifikation jeder Komponente für die Reviewsitzung genauestens angesehen haben, können die Komponententestfälle ohne zusätzlichen Einarbeitungsaufwand nach Auswahl der Testverfahren spezifiziert werden. Analog zu den anderen vorbereitenden Aktivitäten der korrespondierenden Testphasen ist eine Priorisierung der Testfälle vorzunehmen und die Testrahmen für die Komponenten sind bereitzustellen. Teile der Testrahmen für den Integrationstest können übernommen werden.

Alle vorbereitenden Aufgaben für den Komponententest sind vor der Programmierung der Komponenten abgeschlossen.

3.2.4 Testdurchführung und Debugging in allen Testphasen

Nach der Programmierung sind die entwickelten Programmteile zu testen und zu integrieren. In jeder Testphase sind die vorbereiteten Tests entsprechend der Priorisierung auszuführen. Ein Vergleich der tatsächlichen Ergebnisse mit den erwarteten Ergebnissen der Testfälle ist vorzunehmen sowie eine Bewertung, ob ein Fehler vorliegt. Die Ergebnisse werden im Testprotokoll festgehalten. Damit ist die Arbeit des Testers vorerst abgeschlossen.

Der Entwickler bekommt die Ergebnisse der Testdurchführung und lokalisiert die Fehlerursache (Debugging). Nach der Beseitigung der Fehler wird vom Tester ein erneuter Test durchgeführt, der prüfen soll, ob der Fehler beseitigt ist und keine neuen Fehler hinzu gekommen sind.

3.3 Nachteile des W-Modells

Wie alle Modelle so ist auch das W-Modell eine vereinfachte Darstellung. Es existieren mehrere Teilsysteme und jedes dieser Teilsysteme besteht meist aus mehreren Komponenten. Somit sind viele unterschiedliche Komponenten- und Integrationstests durchzuführen. In der Darstellung spiegelt sich dieser Sachverhalt nicht wider. Genauso wenig ist die Notwenigkeit dargestellt, dass bei einem Fehler ein „Rückfall“ mindestens auf die Ebene der Programmierung stattfindet. Fehler können aber auch Auswirkungen auf die linke Seite des Modells haben und zu einer Überarbeitung der dort erstellten Dokumente führen. Ein mühevoller „Wiederaufstieg“ wird notwendig.

Softwareentwicklung findet meist in Iterationen statt. Auch dieser Aspekt wird im Modell nicht deutlich. Für jede Iteration ist bildlich gesehen ein weiteres „W“ hinzuzufügen, die auch zeitlich überlappend angeordnet sein können.

Während der Entwicklung eines Softwaresystems werden sich Anforderungen ändern, neue hinzu kommen und andere möglicherweise wegfallen. Nach dem W-Modell werden zu Beginn des Projektes zu allen Anforderungen die entsprechenden testvorbereitenden

Aktivitäten durchgeführt. Ändern sich Anforderungen, so sind auch die entsprechenden Testdokumente anzupassen, kommen neue hinzu, so müssen auch die Testdokumente ergänzt werden. Fallen Anforderungen weg, sind die entsprechenden Arbeiten zum Test ebenfalls obsolet. Dieser Nachteil wird dadurch etwas ausgeglichen, dass durch die intensivere Auseindersetzung mit den Anforderungen, aus Sicht der Entwicklung und aus Sicht des Tests, einige der unklaren Kundenanforderungen präzisiert werden und somit später keinen Änderungen mehr unterliegen.

4 Zusammenfassung

Im W-Modell werden die Testaktivitäten parallel zu den Entwicklungsaktivitäten gesehen und beginnen mit dem Projektstart. Bei den entwicklungsbegleitenden Reviews werden auch die jeweiligen Auswirkungen auf den Test diskutiert und führen ggf. zu einer Überarbeitung der Ergebnisse. Alle testvorbereitenden Aufgaben werden parallel zur jeweiligen Entwicklungsphase durchgeführt, die zur korrespondierenden Testphase gehört. Alle Testvorbereitungen sind vor der Programmierung abgeschlossen. Nach der Programmierung sind die Tests auszuführen und aufgedeckte Fehler sind zu beseitigen und anschließend erneut zu testen. Test und Debugging sind unterschiedliche Aufgaben, die von unterschiedlichen Personen durchzuführen sind. Im W-Modell wird diese Trennung der Aufgaben deutlich.

Im W-Modell wird eine enge kooperative Zusammenarbeit zwischen Entwicklern und Testern von Projektbeginn an gefordert. Dabei sollen aber nicht die Rollen der beiden Personengruppen in der Softwareentwicklung verwischt oder vermischt werden. Die enge Zusammenarbeit bringt den meisten Gewinn, wenn die unterschiedlichen Sichtweisen beibehalten werden. Hierin unterscheidet sich das W-Modell von der paarweisen Programmierung im Extreme Programming, wo Entwickler mit Entwicklern eng zusammenarbeiten.

Das Allgemeine V-Modell sollte durch das W-Modell abgelöst oder zumindestens ergänzt werden. Durch das Hinzufügen eines parallelen Testprozesses im W-Modell wird der in der Praxis oft anzutreffende Testaufwand von 30-50% „sichtbar“. Damit wird deutlich herausgearbeitet, dass der Aufwand sich nicht allein auf die Ausführung der Testfälle bezieht, wie es beim Allgemeinem V-Modell leicht fehlinterpretiert werden kann. Dem Testen wird im W-Modell der Stellenwert gegeben, den es in der Praxis bereits hat, allerdings mit den klaren Aussagen, dass die Aktivitäten zum Testen nicht erst nach der Programmierung beginnen dürfen und dass Testen und Debugging unterschiedliche Aktivitäten sind.

Das W-Modell ist sicherlich nicht für alle Softwareprojekte gleichermaßen einsetzbar. Ein universelles Modell „für alles“ kann und wird es nicht geben. Das W-Modell hat seine Stärke in der einfachen und plakativen Darstellung, die als Diskussionsgrundlage dienen kann, um Änderungen an der bisherigen Projektabwicklung einzuleiten, oder auch als „Prozess-Rahmen“, der durch Aktivitäten des eigene Projektprozesses ergänzt und angepasst wird. Ein Prozessmodell hat nur dann eine Chance im Entwicklungsalltag zu bestehen, wenn es klar und einsichtig ist, von allen als verbindliche Richtlinie gesehen und mit Leben gefüllt wird. Das W-Modell hat bereits Einzug in die Praxis gefunden und wird in der Ausbildung, im Consulting und in der Entwicklung eingesetzt.

Literatur

[Be00] Beck, K.: Extreme Programming. Addison Wesley, München, 2000

[Bo73] Boehm, B.W.: Software and it´s impact: a quantitative assessment. Datmation, 19, No. 5, 1973, 48-59

[Bo79] Boehm, B.W.: Guidelines for Verifying and Validating Software Requirements and Design Specification. Euro IFIP 1979, 711-719

[Ja99] Jacobson, I.; Booch, G.; Rumbaugh, J.: The Unified Software Developement Process. Addison-Wesley Longman, Amsterdam, 1999

[Po02] Pol, M.; Koomen, T.; Spillner, A.: Management und Optimierung des Testprozesses. dpunkt Verlag, Heidelberg, 2. Auflage, 2002

[SL02] Spillner, A.; Linz, T.: Basiswissen Softwaretest. Aus- und Weiterbildung zum Certified Tester, dpunkt Verlag, Heidelberg, 2002

[Sp00] Spillner, A.: From V-Model to W-Model - Etablishing the Whole Test Process. Proceedings Conquest 2000 – 4th Conference on Quality Engineering in Software Technology, Nürnberg, 2000, 222-231

[Sp02] Spillner, A.: The W-Model – Strengthening the Bond Between Development and Test. Proceedings STAR*EAST*, 10th International Conference on Software Testing, Analysis & Review, Orlando, Florida, USA, May 2002

[Ve99] Versteegen, G. (Hrsg.): Das V-Modell in der Praxis, dpunkt Verlag, Heidelberg, 1999

[Vo01] Vohwinkel, D.: Test process assessments and improvement. In [WM01], 61-72

[WM01] Wieczorek, M.; Meyerhoff, D.: Software Quality. State of the Art in Management, Testing, and Tools. Springer Verlag, Heidelberg, 2001