Verantwortungsbereiche des Testteams in der Anwendungsentwicklung

Phase	Verantwortungsbereich
Grobanalyse	■ Entwicklung von Teststrategien. ■ Festlegung von Akzeptanzkriterien. ■ Entwicklung eines Fehlerüberwachungssystems.
Feinanalyse	■ Bewertung des Entwurfs, um zu ermitteln, ob die Funktionen korrekt getestet werden können. ■ Erarbeitung eines Test- und Zeitplans für Funktionstests. ■ Erarbeitung von Methoden und Maßstäben für die Rückverfolgung von Fehlern. ■ Entwicklung von Teststrategien.
Realisierung	■ Erstellung detaillierter Spezifikationen, Pläne, Einsatzszenarien, Daten und Skripts für Durchführung erster Funktionstests. ■ Prüfung von internen Zwischenlieferungen, Alpha- und Betaversionen. ■ Fehler finden und verfolgen. ■ Dokumentation des Testprozesses.
Test und Freigabe	■ Freigabetests.

Tabelle 1 Verantwortungsbereiche des Testteams in der Anwendungsentwicklung

Funktionstest werden von den Entwicklern durchgeführt, Eignungstests vom Testteam. Mitglieder des Testteams sollten sich mit der Anwendungsentwicklung auskennen, an der Entwicklung der zu testenden Anwendung jedoch nicht beteiligt sein.

Funktionstests sind:

- *Komponententests* konzentrieren sich darauf, ob Eigenschaften vorhanden und funktionsfähig sind.
- *Check-in-Tests* sind schnelle, automatisierte Tests, die von den Entwicklern ausgeführt werden, bevor sie Code in ein Quellcodeverwaltungssystem einchecken.
- *Tests zur Verifizierung der Erstellung*, die nach der täglichen Erstellung ausgeführt werden, um zu überprüfen, ob das Produkt erfolgreich erstellt wurde.
- *Regressionstests* sind automatisierte Tests, die nach der täglichen Erstellung und deren Verifizierung ausgeführt werden, um zu überprüfen, dass die Codequalität nicht nachgelassen hat, d.h. bereits realisierte Funktionalität immer noch fehlerfrei ist.

LITTLE softwarehouse GmbH
Zentnerstr. 19, D-80789 München
www.littlesoft.com

Haben Entwickler sich das Ziel gesetzt, Fehler vor den Testern zu finden, führen sie Komponententests aus.

Eignungstests sind:

- *Konfigurationstests* bestätigen, dass ein Produkt auf der Zielhardware und dem Zielsystem ausgeführt werden kann.
- *Kompatibilitätstests* bestätigen die Verträglichkeit mit anderen Programmen und unter Umständen mit älteren Versionen des Programms.
- *Belastungstests* belasten ein Produkt bis zu seinen Grenzen. Dazu gehören die Speichermangel im Arbeitsspeicher und auf der Festplatte, umfangreicher Netzwerkverkehr und eine hohe Benutzeranzahl.
- *Leistungstests* bewerten den gesamten Systemdurchsatz (Transaktionen pro Sekunde) und die Reaktionszeit (z.B. 95% aller Anfragen werden innerhalb einer Sekunde beantwortet).
- *Tests der Dokumentation und der Hilfedateien* konzentrieren sich auf falsche Inhalte und Abweichungen vom Produkt.

Fehlerüberwachung

Ein *Fehler* ist alles, was vor der Freigabe der Anwendung adressiert und behoben werden muss. Die weit verbreitete Meinung, dass Fehler immer Defekte sind, stimmt nicht. Defekte bilden eine Fehlerkategorie, doch Fehler umfassen mehr. Ein Fehler ist jeder Problempunkt, der sich aus der Verwendung der Anwendung ergibt, z.B.:

- Forderungen von Verbesserungen,
- Vorschläge, die bis zur Freigabe nicht mehr berücksichtigt werden können,
- Probleme durch Benutzervorlieben,
- Unvermeidbare Entwurfskonsequenzen,
- Defekte

Fehlerüberwachungsprozesse umfassen Aspekte wie Fehlerberichte, Prioritäten, Anweisungen, Behebung und Abschluss. Fehler müssen verfolgt und verwaltet werden, damit das Projektteam während der Stabilisierungsphase (Phase „Test und Freigabe“) die notwendigen Entscheidungen treffen kann.

Vor Beginn des Fehlerüberwachungsprozesses muss eine Fehlersammlung, üblicherweise eine Datenbank, eingerichtet werden. Neue Fehler werden durch Eingabe in die Datenbank veröffentlicht. Jeder Tester gibt die Fehler ein, die er selbst gefunden hat. Neue Fehler werden als aktive Fehler gekennzeichnet.

Der Leiter des Entwicklungsteams weist den Fehlern Prioritäten zu und verteilt sie an bestimmte Entwickler, die sich um die Korrektur kümmern. Die Entwickler korrigieren alle Fehler, die eine entwicklungsbasierte Behebung erfordern und testet diese. Ist der aktive Fehler korrigiert, ändert sich dessen Status. Danach prüft ein Tester die Qualität der Korrektur und stellt sicher, dass sie keine neuen Fehler nach sich zieht. Tritt ein neuer Fehler auf, dann trägt ihn der Tester in die Fehlerdatenbank ein, und damit wird der gesamte Zyklus erneut gestartet. Wurde der ursprüngliche Fehler nicht erfolgreich korrigiert, dann reaktiviert der Leiter des Entwicklungsteams den Fehler.

Das Team (aber nicht ein einzelner Entwickler) kann immer entscheiden, den Fehler nicht zu korrigieren, sondern ihn stattdessen zu dokumentieren. Auf diese Weise entscheidet das Team, die Fehlerkorrektur zu verschieben und dies Arbeit nach der Produktfreigabe zu erledigen.

Die Fehlerklassifizierung bietet eine Möglichkeit, Prioritäten und Risiken zu erkennen. Die Klassifizierung zeigt zwei wichtige Aspekte auf: die Dringlichkeit, die sich auf die Auswirkungen des Fehlers auf die gesamte Anwendung bezieht und die Priorität, welche die Auswirkungen des Fehlers auf die Stabilität der Anwendung bewertet. Fehler müssen nicht nur gemeldet, sondern auch klassifiziert werden.

Eine typische Klassifizierung der Dringlichkeit sieht folgendermaßen aus:

- *Dringlichkeit 1: Systemausfall* Systemabstürze, Systemblockierungen, das System arbeitet sonst wie nicht mehr oder Datenverluste treten auf.
- *Dringlichkeit 2: Großes Problem* Schwer wiegender Defekt der Softwarefunktion, der nicht unbedingt zum Systemabsturz oder zu Datenverlusten führt.
- *Dringlichkeit 3: Kleines Problem* Defekt in der Funktion der Software, der gewöhnlich nicht Daten- oder Arbeitsverluste nach sich zieht.

- *Dringlichkeit 4: Trivial* Kleinere kosmetische Probleme, die von den meisten Benutzern unbemerkt bleiben.

Eine typische Klassifizierung der Prioritäten sieht folgendermaßen aus:

- *Priorität 1*: Die Anwendung kann nicht ausgeliefert werden und das Team erreicht den nächsten Meilenstein wahrscheinlich nicht.
- *Priorität 2*: Die Anwendung kann nicht ausgeliefert werden, doch das Team erreicht den nächsten Meilenstein wahrscheinlich dennoch.
- *Priorität 3*: Die Anwendung kann ausgeliefert werden und das Team erreicht den nächsten Meilenstein. Die Behebung dieser Fehler werden tendenziell nur dann korrigiert, wenn am Ende des Projektes genügend Zeit und die Behebung nicht riskant ist.
- *Priorität 4*: Forderungen von Verbesserungen und Fehler, die es nicht wert sind, korrigiert zu werden.

Eine Fehler wird abgeschlossen, wenn ein Tester geprüft hat, dass die Behebung des Fehlers nicht ein anderes Problem erzeugt und der Fehler höchstwahrscheinlich nicht mehr auftritt.

Fehler werden normalerweise wie folgt aufgelöst:

- *Korrigiert*: Der Entwickler hat den Fehler korrigiert, die Korrektur getestet, den Code geprüft, der Korrektur eine Produktfreigabenummer zugewiesen und den Fehler wieder an den Tester zurückgegeben, der ihn gemeldet hat.
- *Duplikat*: Der Fehler ist ein Duplikat eines anderen Fehlers, der schon in der Fehlerdatenbank eingetragen ist. Der doppelt eingetragene Fehler wird aufgelöst, geschlossen und mit dem Originalfehler verknüpft.
- *Verschoben*: Der Fehler wird nicht in der aktuellen Version behoben, aber wahrscheinlich in einer Folgeversion. Diese Auszeichnung wird eingesetzt, wenn das Team Vorteile darin sieht, den Fehler zu beheben, allerdings keine Zeit oder Ressourcen besitzt.
- *Gemäß des Entwurfs*: Das gemeldete Verhalten ist beabsichtigt und in der Spezifikation beschrieben.
- *Nicht reproduzierbar*: Der Entwickler kann den Fehler nicht verifizieren.
- *Nicht korrigieren*: Der Fehler wird in der aktuellen Version nicht behoben, weil das Team sich entschieden hat, dass sich die Mühe nicht lohnt oder weil das Management oder der Kunde dem Fehler keine Bedeutung zumisst.

LITTLE softwarehouse GmbH
Zentnerstr. 19, D-80789 München
www.littlesoft.com