## Anforderungsmanagement und Testen

# Die imbus TestBench – ein iterativer Prozess

Die Spezifikation von Anforderungen ist nur eine Seite der Medaille – das Testen gegen zuvor spezifizierte Anforderungen hingegen wird meist noch stiefmütterlich betrieben. Dabei liegen die Vorteile einer solchen Vorgehensweise klar auf der Hand: Nur so kann glaubhaft versichert werden, dass der Kunde das in der entsprechenden Qualität erhalten hat, was er auch wollte.

### Der generische Testprozess

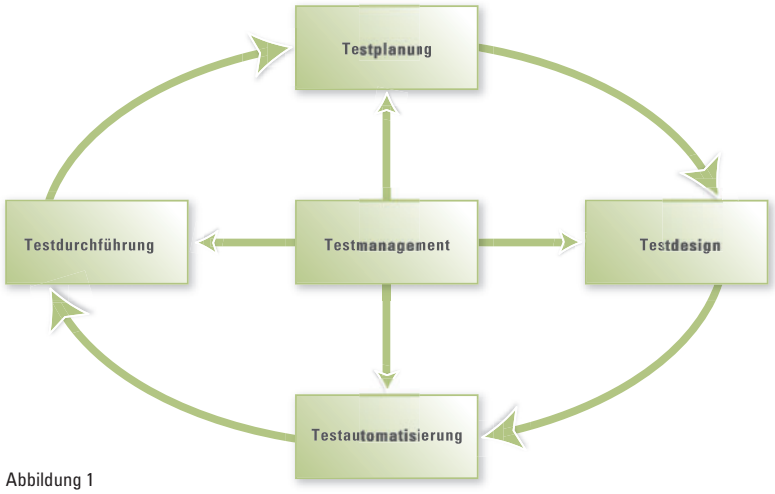


Abbildung 1

Viele Softwareprojekte leiden heutzutage unter einem (oder auch mehreren) der folgenden Mängel:

- Die Testdokumentation ist lückenhaft und nicht revisionssicher.
- Die Testfälle sind unsauber beschrieben und schwer reproduzierbar.
- Die automatisierten Testprozeduren und Testdaten sind inkonsistent, weil sie auf unzählige Skripte und Tabellen verstreut sind.
- Jedes neue Testobjekt-Release erzwingt aufwändige Anpassungen.
- Es existiert keine Anbindung zum Anforderungsmanagement bzw. zu Anforderungsmanagementwerkzeugen – es kann also nicht gegen existierende Anforderungen getestet werden
- Die Freigabe eines Produktes erfolgt oft "aus dem Bauch heraus".

**Die Auswirkungen** dieser Mängel sind offensichtlich – Qualitätsprobleme, Wartungsschwierigkeiten, Kundenunzufriedenheit usw. Zur Lösung dieses Problems hat die imbus AG mit der imbus TestBench (siehe auch Kurzbeschreibung Seite 26) ein Werkzeug auf den Markt gebracht, das alle Phasen im Testprozess unterstützt und über eine Anbindung zu den relevanten Anforderungsmanagementwerkzeugen, wie zum Beispiel IRqA von QA Systems oder CaliberRM von Borland verfügt.

## Generischer, iterativer Prozess

Die imbus TestBench basiert auf der Sicht des Software-Tests als iterativen Prozess, der aus den folgenden Phasen besteht:

- Testplanung,
- Testdesign,
- Testautomatisierung und
- Testdurchführung.

Diesem Standard-Testprozess liegen Rollen- und Rechtedefinitionen, Use Cases und erzeugbare Dokumente zu Grunde, wie in Abbildung 1 dargestellt ist. Im folgenden sollen die einzelnen Phasen und deren Inhalte näher dargestellt werden.

**Die Testplanung** beinhaltet alle Aktivitäten, bei denen identifiziert wird, was im weiteren Verlauf des Testprozesses zu validieren und zu verifizieren ist. Hier werden Releasepläne und Testobjekte verwaltet und die Testthemen aufgelistet, priorisiert und den entsprechenden Testobjekten zugeordnet. Ferner findet eine Zuweisung der Testaufgaben auf die einzelnen Teammitglieder statt. Damit wird eine saubere Planung der Testarchitektur und die erforderliche Aufgabenverteilung gewährleistet.

**Bei der Testspezifikation** (auch als Testdesign bezeichnet) wird festgehalten, wie zu verifizieren und zu validieren ist. Ein formales, modulares Design von Testfällen ist der wesentliche Erfolgsfaktor für die Wirksamkeit von systematischen Softwaretests. Die imbus TestBench nutzt dabei die sogenannte Interaktionsmethode. Die Interaktionsmethode basiert auf wiederverwendbaren Ablaufsequenzen – eben den Interaktionen – und abstrakten Datentypen. Letztere werden in Äquivalenzklassen zerlegt, wobei jeder Äquivalenzklasse ein oder mehrere Repräsentanten zugeordnet sein können. Die den Testfällen zugeordneten Testdaten werden dadurch formal definiert abstrahiert. Die Testfälle werden in beliebig vielen Hierarchiestufen in Interaktionsschritte gegliedert. Je nach Abstraktionsebene ist eine Interaktion ein fachlicher oder auch ein technischer Testschritt. Hierunter versteht man das Anreizen des zu testenden Systems, zum Beispiel durch Eintragen von Daten, Senden einer Nachricht oder ähnlichem, oder die Prüfung einer erwarteten Sollreaktion des Systems, zum Beispiel ein erzeugtes Datum oder eine empfangene Nachricht. Interaktionen sind parametrierbar. Sie können bei jeder Verwendung unterschiedliche Daten senden, empfangen und verarbeiten.

Durch Abstraktion und Kapselung von Interaktionen und Datentypen wird eine robuste Test-Architektur gewährleistet, die stabil gegenüber Veränderungen des Testobjektes, von Testfällen oder Testdaten ist.

**In der Phase der Testautomatisierung** werden diese Testspezifikationen zu automatisch ablauffähigen Testprogrammen umgesetzt. Primäres Ziel ist es, durch die Automatisierung Zeit und Ressourcen bei der Wiederholung von Tests einzusparen. Die Praxis zeigt leider vielfach, dass solche Automatisierungen, wenn sie nicht durch eine hinreichend reife Methodik bei der Entwicklung und Verwaltung der Automatisierungs-Skripte unterstützt werden, diesen Anspruch nicht erfüllen, sondern im Gegenteil hohe Aufwände bei der Wartung und Erweiterung der Skripte erzeugen.

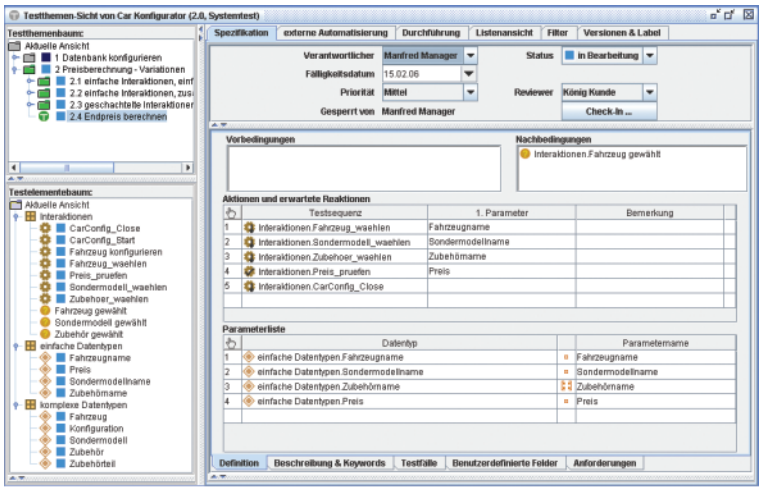


Abb.2: Mittels der Interaktionsmethode werden Testabläufe aus einzelnen, wieder verwendbaren Testelementen zusammengesetzt.

**Die imbus TestBench** gliedert Testfälle innerhalb sogenannter Testfallsätze und diese wiederum in Testthemen. Alle Testfälle in einem Testfallsatz haben die gleiche Sequenz von Interaktionen zwischen Testobjekt und Tester (bzw. Testumgebung), aber ggf. verschiedene Parametrierungen, also unterschiedliche Angaben, welche konkreten Daten während der Durchführung dieses Testfalls an das Testobjekt übergeben und welche Sollreaktionen (Ausgabedaten) von ihm erwartet werden. Hierdurch ist eine Erhöhung der Testtiefe oftmals durch einfaches Hinzufügen von Parametrierungen möglich. Testfallsätze können baumartig in Testthemen strukturiert werden, wobei ein Testthema sowohl Testfallsätze als auch andere Testthemen enthalten darf.

Durch die Anwendung der Interaktionsmethode bei der Spezifikation wird eine kostenoptimal nutzbare, robuste und hochgradig wiederverwendbare Testautomatisierung auf einer abstrakten Ebene bereits automatisch erzeugt. Zur Erzeugung einer ablauffähigen Automatisierung bedarf es lediglich noch der

Ausprogrammierung sogenannter elementarer Interaktionen (oder auch Treiber) auf der Ebene eines konkreten, an die TestBench ankoppelbaren, Scripting-Werkzeuges. Die imbus TestBench arbeitet mit praktisch allen solchen Werkzeugen im Bereich von GUI-, Embedded- und API-Testautomatisierung zusammen.

**Bei der Testdurchführung** werden die Testläufe – einheitlich für manuelle oder automatisierte Tests – gestartet, überwacht und ihre Testergebnisse erfasst und revisionssicher archiviert. Dabei können Testteilmengen gefiltert und zu zielgerichteten Testläufen bzw. Auswertungen zusammengestellt werden. Ein Highlight dieser Phase ist der sogenannte manuelle Durchführungsassistent, der sowohl im Online- als auch im Offline-Betrieb (d.h. beispielsweise im Offshore-Testdurchführungslabor, bei Tests in Fahrzeugen etc.) dem manuellen Tester eine komfortable Umgebung zur Darstellung der Testschritte, zur Protokollierung der Ergebnisse und zur Erfassung von Soll/Ist-Abweichungen zur Verfügung stellt. Solche Soll/Ist-Abweichungen können aus der TestBench heraus direkt in marktübliche Defect Management-Systeme wie z.B. Bugzilla übernommen werden und bei Retests auch zur Planung z.B. von Fehlernachtests herangezogen werden.

Das Testmanagement überprüft dann die Abdeckung und die Produktqualität. Gerade hier haben eine Vielzahl von Unternehmen mit den folgenden Schwierigkeiten zu kämpfen:

- Unklarer Testfortschritt (niemand hat den Überblick, was schon getestet wurde, und was noch nicht. Es ist nicht nachvollziehbar, welche Version des Testobjekts mit welcher Version der Testspezifikation mit welchem Ergebnis getestet wurde)
- Unvollständige Testdokumentation
- Unbekannter Zustand der Anforderungen
- Unvollständige Berichte
- Als Folge: unbekannte Produktqualität und Freigabe "aus dem Bauch heraus"

**Das Testmanagement** verfolgt den Fortschritt der Testentwicklung und -durchführung. Die TestBench sorgt für die homogene Ablage und Berichterstattung sowohl manuell als auch automatisiert erzeugter Testergebnisse und stellt zur Auswertung eine sehr flexible Reporting-Schnittstelle zur Verfügung.

## Automatische Versionierung inbegriffen

Alle Produkte des Testprozesses wie Testspezifikationen und Durchführungsergebnisse können in der imbus TestBench unabhängig voneinander versioniert werden, das heißt, der Bearbeiter kann den Zustand der Bearbeitung eines Elements sichern und später wieder herstellen. Abhängigkeiten zwischen einzelnen Elementen, wie zum Beispiel zwischen Testfällen und den Designelementen, aus denen sie bestehen, oder zwischen Testdurchführungsergebnissen und der Testspezifikation, auf der sie basieren, werden dabei automatisch gepflegt und die Konsistenz zwischen ihnen wird sichergestellt.

**Wichtige Ordnungselemente** sind hierbei die sogenannte Testobjekt-Version und der Testzyklus: Die Testobjekt-Version kennzeichnet die Zugehörigkeit von Testspezifikations- und -automatisierungsdokumenten zu einem bestimmten Funktionsumfang oder Lieferstand des Systems unter Test. Der Testzyklus kennzeichnet die Zugehörigkeit von Testdurchführungs-Ergebnissen zu einer bestimmten Testobjekt-Version.

Innerhalb eines Testprojekts sind die Versionen der verschiedenen Objekte baumartig geordnet.

Innerhalb einer Testobjekt-Version können zudem beliebig viele Versionen eines Testspezifikations- oder -automatisierungsdokuments erzeugt werden und innerhalb eines Testzyklus beliebig viele Versionen einer Durchführungsinformation.

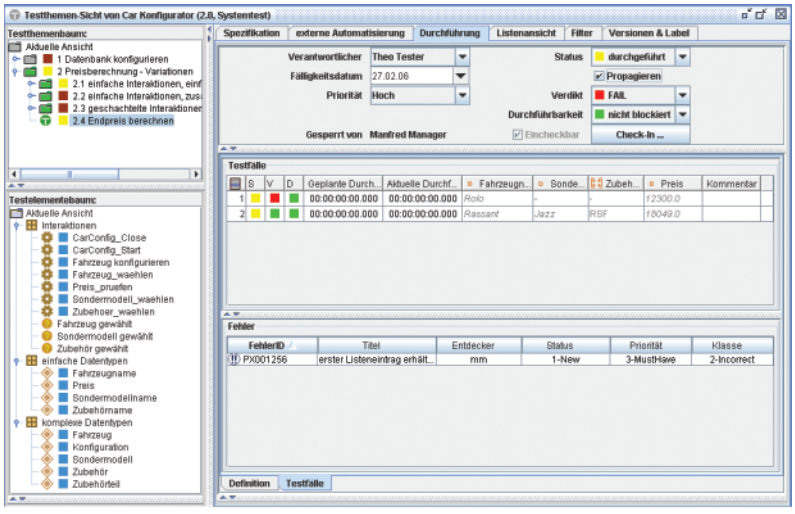


Abb. 3: Die Testdurchführungssicht ermöglicht einen schnellen Überblick über alle Testergebnisse.

## Stabile Architektur

Die imbus TestBench ist ein plattformunabhängiges skalierbares System auf der Basis von Sun Java 2 Enterprise Edition. Es handelt sich also um ein 3-Schicht-System mit einer SQL-basierten Datenbank im Hintergrund, einem Java-Application-Server und beliebig vielen "Rich"-Clients mit grafischer Oberfläche ebenfalls auf der Basis von Java. Als Datenbank kann Microsoft SQL-Server oder ORACLE RDMBS zum Einsatz kommen, als Application Server dient JBoss.

Die imbus TestBench ist als Single-Server-System ausgelegt, d.h. alle Benutzer arbeiten auf dem gleichen Application Server mit der gleichen Datenbank. Durch die Clustering-Eigenschaften von hinterlegtem Datenbank- und EJB-Server ist eine hohe Skalierbarkeit gegeben, so

dass ein TestBench-System – je nach Leistungsfähigkeit der eingesetzten Hardware – durchaus Dutzende parallele Benutzer bedienen kann.

### Highlight: Schnittstelle zum Anforderungsmanagement

Die imbus TestBench bietet eine generische Schnittstelle zur Ankopplung von Werkzeugen für Anforderungsdefinition und -management. Innerhalb der TestBench werden die Anforderungsdaten verschiedener Werkzeuge wie Borland CaliberRM oder QA Systems IRqA gleichartig dargestellt. Die Abbildung zwischen dem konkreten Anforderungsmanagementwerkzeug und dem allgemeinen Datenmodell der imbus TestBench leistet ein toolspezifischer Wrapper, der Bestandteil der TestBench-Serverarchitektur ist. Damit können die vier wichtigsten Inhalte umgesetzt werden:

[ Anforderungsbasierte Testplanung: Durch die generische Schnittstelle erhalten Testmitarbeiter innerhalb der TestBench Zugriff auf Struktur und Inhalte von Anforderungen; Informationen wie z.B. Anforderungspriorität und -realisierungsstatus, Kategorisierung nach verschiedenen Qualitätsmerkmalen und die textuelle Beschreibung der Anforderungen sind im direkten Zugriff und können zur Planung und zum Entwurf von Testfällen herangezogen werden.

[ Anforderungsbasierter Testfallentwurf: Testdesigner können direkt Einsicht in die Spezifikation der Anforderungen nehmen und sie mit den Testfallsätzen verknüpfen und somit zum Ausdruck bringen, welche der Testfallsätze welche Anforderungen verifizieren.

[ Anforderungsbasiertes Reporting: Testmanager können sich mit den mächtigen Filter- und Reportingfunktionen einen schnellen Überblick über die jeweils erreichten Testdesign- und Testdurchführungsabdeckungen schaffen.

[ Management von Änderungen: Die Schnittstelle liefert auch jederzeit Informationen über Änderungen der Anforderungen zwischen zwei "Baselines" (Mengen zusammengehöriger Versionen

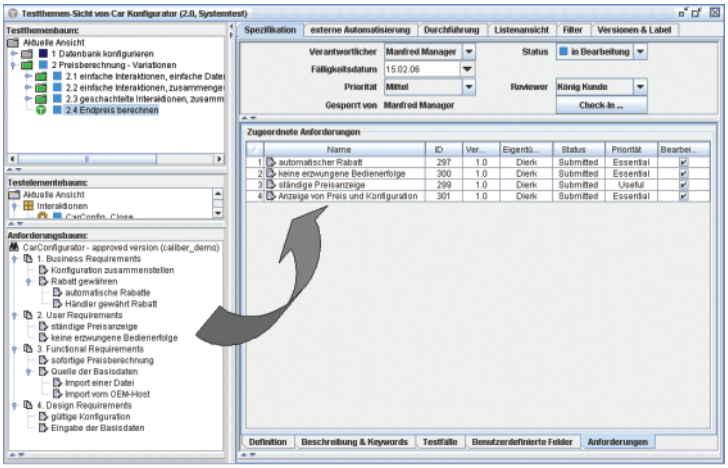


Abb. 4: Anforderungen können per Drag&Drop Tests oder ganzen Testthemen zugeordnet werden.

der Anforderungs-Objekte). Auf diese Weise kann die Planung der Testaktivitäten eng mit dem Fortschritt der Anforderungsdefinition und -umsetzung synchronisiert werden.

Technisch ist die Anbindung so gelöst, dass ein spezieller Client, die Requirements-Bridge, der imbus TestBench mit der IRqA-API verbunden wird und darüber das Repository des Testmanagementsystems mit Daten versorgt wird. Die Anforderungen werden innerhalb des Testmanagements genau in der gleichen Struktur, wie durch das Anforderungsmanagementsystems vorgegeben, abgelegt und dem Testdesigner innerhalb seiner Testspezifikation zur Verknüpfung angeboten. Der Testmanager kann dabei entscheiden, welches Projekt mit welchem View aus dem Anforderungsmanagementsystems ins Testprojekt übernommen wird. Die eventuell notwendige Aktualisierung der Anforderungsdaten kann ebenfalls jederzeit vom Testmanager ausgelöst werden. Innerhalb des Testspezifikation werden die Anforderungen durch einfaches Drag&Drop vom Testdesigner mit den Tests verknüpft. Sind die Tests durchgeführt, wird aus den Testergebnissen für jede Anforderung ein Teststatus gebildet, der auch die mögliche n:m-Beziehung von Anforderungen zu Tests berücksichtigt. Der Teststatus kann wiederum über einen getriggerten

» **Die Schnittstelle liefert auch jederzeit Informationen über Änderungen der Anforderungen** zwischen zwei "Baselines". [...] Auf diese Weise kann die Planung der Testaktivitäten eng mit dem Fortschritt der Anforderungsdefinition und -umsetzung synchronisiert werden. «

Abgleich in das Anforderungsmanagementsystem übernommen werden.

Abbildung 5 zeigt die Anbindung des Anforderungsmanagementwerkzeuges IRqA in der imbus TestBench.

### Beteiligte Rollen

An der Durchführung der in der imbus TestBench integrierten Methode sind verschiedene Rollen beteiligt:

- Die Fachabteilung (Kunde) hat das Wissen über Anforderungen und Use Cases des zu testenden Systems und stellt es in Form von Dokumentationen und/oder Interviews oder durch Erstellung einer High-Level-Spezifikation mit der Interaktionsmethode zur Verfügung.
- Der Testdesigner transformiert dieses fachliche Wissen basierend auf den zu prüfenden Qualitätsmerkmalen und zur Prüfung geeigneter Teststrategien in entsprechende formale Testfälle.
- Der Testprogrammierer kann darauf aufbauend Testprogramme erzeugen, die die im formalen Testfallentwurf festgelegten Eigenschaften besitzen. Er benötigt hierzu keinen tiefgreifenden fachlichen Hintergrund, sondern hauptsächlich Wissen über Testautomatisierungswerkzeuge und ihre Programmiersprachen.
- Die Aufgabe des Testers ist es dann im Fall eines automatisierten Tests, die durch die imbus TestBench teilweise automatisch generierten Testprogramme zu starten, ihre Ergebnisse auszuwerten und ggf. aussagekräftige Fehlermeldungen zu erstellen. Im Fall manueller Tests führt er – optional gestützt auf den Durchführungsassistenten – die spezifizierten Tests exakt und wiederholbar durch und protokolliert Ergebnisse und begleitende Artefakte (zum Beispiel Screenshots oder Ist-Datensätze).

*Thomas Roßner*

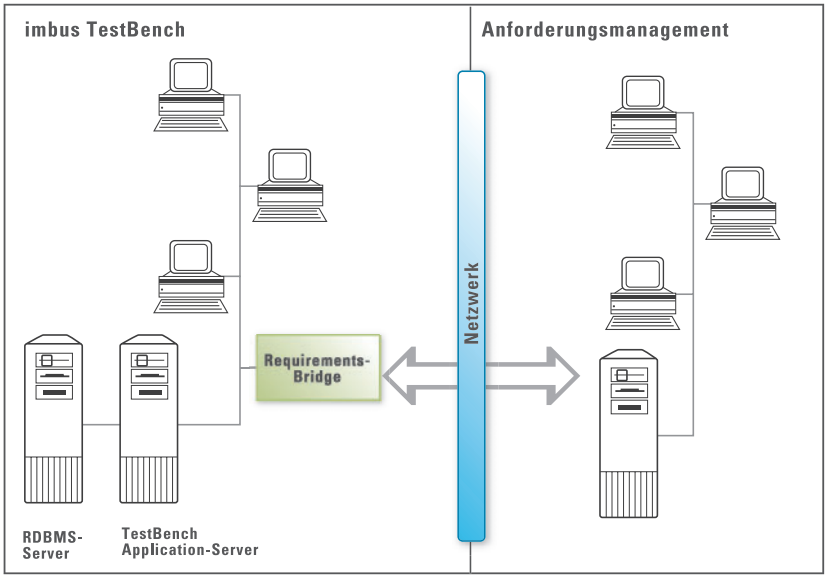


Abb. 5: Die Anbindung des Anforderungsmanagements in der imbus TestBench, wie sie mit dem Anforderungsmanagementwerkzeug IRqA vorgenommen wurde

### Die TestBench zeichnet sich durch die folgenden Eigenschaften aus:

- Die imbus TestBench ist eine integriete Entwicklungsumgebung für Softwaretests. Sie ist plattformunabhängig und bietet eine konsequente, durchgängige Toolunterstützung über alle Phasen im Testprozess.
- In der imbus TestBench erfassen und bearbeiten Testteams alle testrelevanten Informationen. Insbesondere die Toolunterstützung der Testdesign-Aufgaben erzeugt robuste und wiederverwendbare Test-Architekturen.
- Unterschiedliche Testroboter, zum Beispiel für Komponenten- und Integrationstest, automatisierten GUI-Test oder embedded Test, können aus der TestBench heraus angesteuert werden.
- Mit der imbus TestBench lassen sich bereits entwickelte Testabläufe einfach wiederverwenden und damit Testzykluszeiten verkürzen.