



# Effiziente Qualitätssicherung und Testen mit Visual Studio 2010

---

Dieses Whitepaper gibt einen Einblick in die Möglichkeiten der Qualitätssicherung mit Visual Studio 2010 (Beta2) – mit Schwerpunkt auf funktionales Testen.

Zielgruppe sind Tester, Testmanager und Verantwortliche für Qualitätssicherung in Softwareprojekten.

Im Fokus stehen Anforderungsmanagement, Testmanagement, Testautomatisierung sowie Testvirtualisierung und deren Umsetzung mit Visual Studio 2010. Am Ende folgt ein Überblick über weitere Qualitätssicherungsthemen, die weit über das eigentliche Testen hinausgehen, aber zum Methodenbaukasten eines Qualitätsverantwortlichen gehören.

November 2009

Matthias Zieger  
Microsoft Deutschland GmbH

[www.microsoft.de/visualstudio](http://www.microsoft.de/visualstudio)

## Testen Sie mit VS 2010:

- Ihre neuesten Windows 7-Anwendungen
- in aktuellen 64-Bit-Umgebungen
- neueste .NET 4-Applikationen (z.B. WPF)
- in virtuellen und physikalischen Umgebungen

Bringen Sie Ihre Tester und Entwickler enger zusammen.

Vermeiden Sie Bug-Ping-Pong durch nicht reproduzierbare Fehler

**Inhaltsverzeichnis**

Überblick und Zielgruppe ..... 3

Hauptziele von Visual Studio 2010 im Testbereich ..... 3

Anforderungsbasiertes Testen ..... 3

Anforderungs- und Testmanagement mit Visual Studio 2010 ..... 4

Aufzeichnen relevanter Daten in der Testumgebung ..... 5

    „Action log“ und „action recording“ ..... 5

    Videoaufzeichnung und Screenshot während eines Testlaufs ..... 5

    IntelliTrace ..... 5

    Test-Impact-Analyse ..... 5

    ASP.Net Client Proxy für IntelliTrace und Test-Impact-Analyse..... 5

    ASP.NET Profiler ..... 5

    Code Coverage ..... 5

    Event Log..... 6

    Netzwerk-Emulation ..... 6

    Systeminformationen ..... 6

    Eigene Diagnosedaten ..... 6

    Verbindung zur Testauswertung..... 6

Testautomatisierung ..... 7

    Unterstützte Technologien ..... 7

    Unterstützte Testumgebungen..... 7

    Automatisierung mit Visual Studio 2010 in der Praxis..... 8

    Testfall-Recording ..... 9

    Testautomatisierung mit dem Coded UI Test Builder..... 9

    Umwandlung eines manuellen Tests in einen automatisierten Test..... 9

    Datengetriebene automatisierte Tests ..... 9

Last- und Stresstests..... 10

    Loadtest für Webanwendungen ..... 10

    Wiederverwendung von Unittests für Loadtestszenarien ..... 10

    Wiederverwendung von bereits automatisierten Tests im Lastets ..... 10

Testvirtualisierung..... 11

    Testvirtualisierungspraktiken mit Visual Studio 2010..... 11

Fehlerverfolgung, Auswertung und Reporting..... 11

Entwicklung und Test stärker verbinden..... 12

Zusammenfassung: Qualität ist mehr als nur Testen..... 13

## Überblick und Zielgruppe

Dieses Whitepaper soll einen Einblick in die Möglichkeiten der Qualitätssicherung mit Visual Studio 2010 (Beta2) geben. Es richtet sich an Tester, Testmanager und Verantwortliche für Qualitätssicherung in Softwareprojekten. Der Schwerpunkt liegt auf funktionalem Testen.

Im Fokus stehen Anforderungsmanagement, Testmanagement, Testautomatisierung sowie Testvirtualisierung und deren Umsetzung mit Visual Studio 2010 (VS 2010). Am Ende folgt ein Überblick über Qualitätssicherungsthemen, die weit über das eigentliche Testen hinausgehen, aber zum Methodenbaukasten eines Qualitätsverantwortlichen gehören.

## Hauptziele von Visual Studio 2010 im Testbereich

- Bessere Verbindung von Fachseite, Entwicklung und Qualitätssicherung durch Integration in eine ALM-Plattform
- Bereitstellung neuer, integrierter Werkzeuge speziell für Testmanager, Testanalysten und manuelle Tester
- Unterstützung neuester Technologien wie Windows 7, .NET 4, 64-bit-Testumgebungen
- Höhere Testeffizienz durch Testautomatisierung
- Einfachere Fehleranalyse durch automatisches Aufzeichnen wichtiger Testdaten
- Einfache Verwaltung physikalischer und virtueller Testlabore

## Anforderungsbasiertes Testen

Sehr häufig sind Anforderungsmanagement und Testmanagement sowie Testausführung voneinander getrennt. Deshalb kommt es häufig zu Missverständnissen bei der Definition von Anforderungen, falschen oder fehlenden Testfällen und damit zu Kostenüberschreitungen und Qualitätsmängeln. Aus diesen Gründen ist es notwendig, Anforderungsmanagement und Testmanagement enger miteinander zu verbinden. Die Testfälle sind dann direktes Ergebnis der Anforderungen, eine Testausführung und aufgetretene Fehler lassen sich bis zu den Anforderungen zurückverfolgen.

## Anforderungs- und Testmanagement mit Visual Studio 2010

Visual Studio 2010 sorgt für enge Integration von Testausführung, Testmanagement, Anforderungsmanagement, Test-Auswertung, Fehlerverfolgung, und Build-Management. Sind die funktionalen und nichtfunktionalen Anforderungen erstellt und überprüft, kann man Testfälle entwerfen und jeden Testfall den relevanten Anforderungen, Use Cases oder User Stories (je nach Vorgehensmodell) zuordnen. Außerdem ist es möglich, jedem manuellen Testfall eine Testautomatisierung zuzuordnen.

Im Microsoft Test and Labmanager (MTLM) in der Integration mit dem Team Foundation Server haben Sie folgende Funktionen integriert:

- Verschiedene Eingabemöglichkeiten der Anforderungen, u.a. webbasiert, Excel, Visual Studio Team Explorer
- Erstellung der Testsuiten, Testschritte
- Testplanung auf Basis der Anforderungen
- Testparametrisierung (datengetriebene Tests)
- Aufbau einer Testfall- und Testschrittbibliothek
- Verlinkung zum Buildmanagement
- Steuerung für manuelle und automatisierte Testausführung
- Aufzeichnung relevanter Daten in der Testumgebung
- Testauswertung, Fehlerverfolgung, Reporting

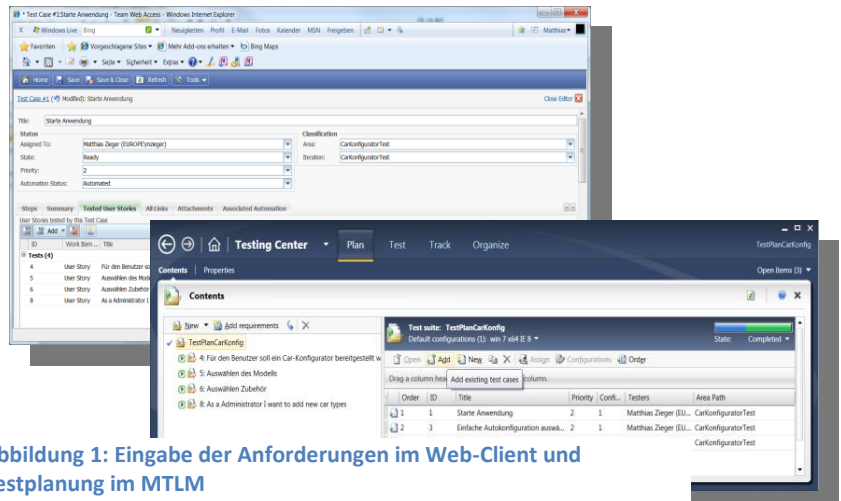


Abbildung 1: Eingabe der Anforderungen im Web-Client und Testplanung im MTLM

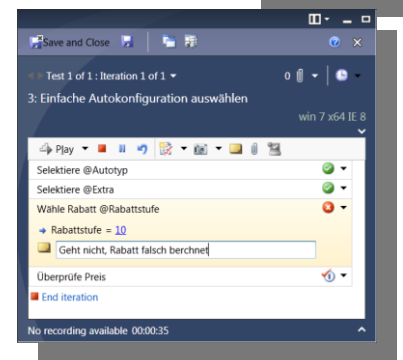


Abbildung 2: Spezieller Client für die manuelle Testausführung

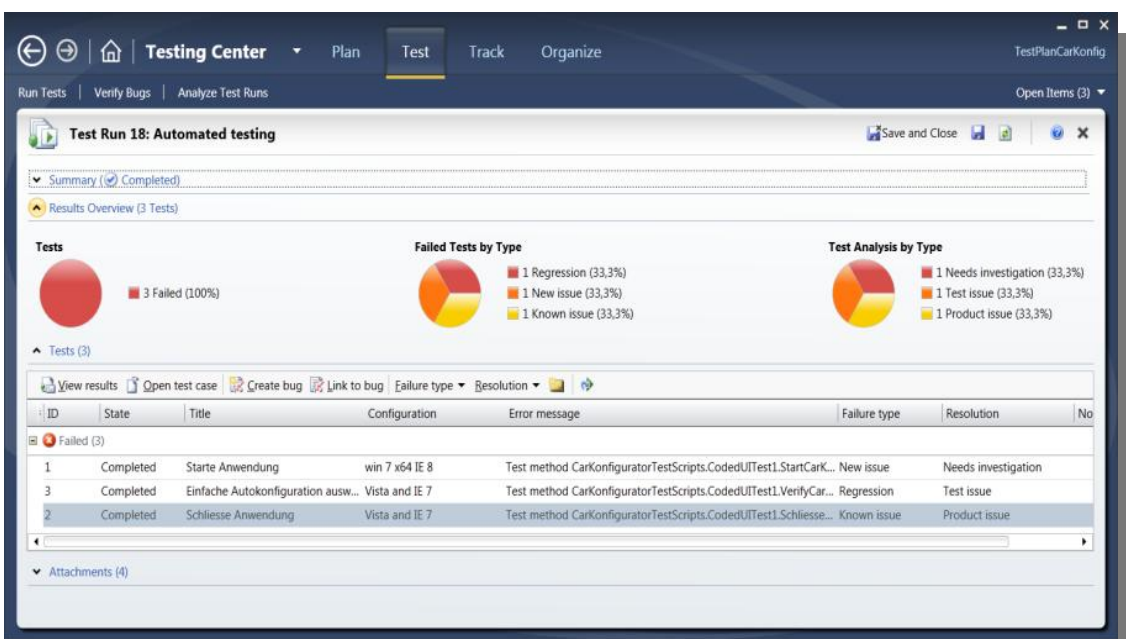


Abbildung 3: Alles im Griff mit MTLM: Testauswertung pro Testlauf (oberer Teil) und pro Testfall (unterer Teil)

## Aufzeichnen relevanter Daten in der Testumgebung

Während einer Testausführung müssen normalerweise sehr viele Daten erfasst werden. Hauptsächlich sind das Daten der Konfiguration der Testumgebung, Prozessdaten der zu testenden Anwendung und Code-Testabdeckungsinformationen.

Im Detail können in der Testumgebung folgende Daten erfasst werden:

### „Action log“ und „action recording“

- Aufzeichnen der Klickpfade bei manueller Testausführung
- Erlaubt Erstellung automatischer Testprotokolle bei manuellen Tests
- Testschritte können ohne Benutzerinteraktion wiedergegeben werden (Fast Forward Replay)
- Ist eine mögliche Grundlage für die Testautomatisierung

### Videoaufzeichnung und Screenshot während eines Testlaufs

- Zeichnet ein Video der Testsituation auf
- Im Fehlerfall kann zusätzlich automatisch oder manuell ein Screenshot der Fehlersituation erzeugt werden
- Für manuelle und automatisierte Tests
- Hilft, Testabläufe besser zu verstehen
- Gut geeignet auch für Usability-Testszenarien

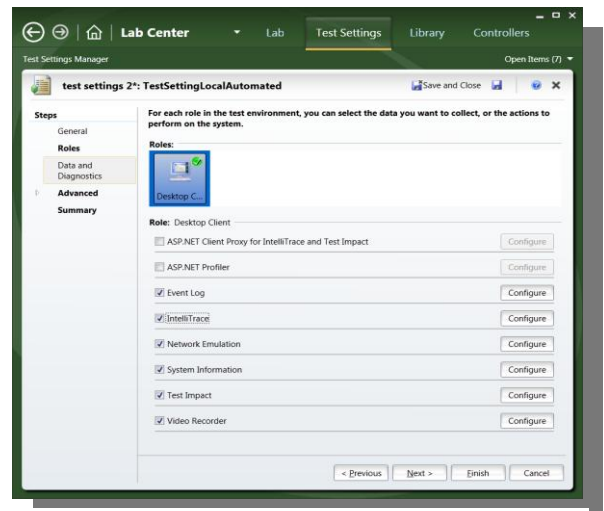


Abbildung 4: Konfiguration der Datenaufzeichnung

### IntelliTrace

- Sammelt Informationen, die helfen, die Fehlersuche und -diagnose zu vereinfachen
- Verbessert die Produktivität beim Debugging
- Aus den IntelliTrace-Daten kann die Test-Session auf einem Entwickler-Rechner wiederhergestellt werden; damit sind Fehler schneller zu reproduzieren

### Test-Impact-Analyse

- Sammelt Informationen, welche Methoden der getesteten Anwendung während eines Testlaufs aufgerufen wurden
- Ermittelt zusammen mit der Code-Änderungsrate aus der Versionierung und dem Build-System, welche Testfälle von Code-Änderungen betroffen sind
- Hilft dabei, Testfälle zu priorisieren und zu selektieren, insbesondere bei Regressionstestszenarien

### ASP.Net Client Proxy für IntelliTrace und Test-Impact-Analyse

- Sammelt Informationen über http-Aufrufe von einem Client zu einem Webserver
- Kann genutzt werden für IntelliTrace und Test-Impact-Analyse von Webanwendungen

### ASP.NET Profiler

- Sammelt Performancedaten (Profiling) bei ASP.NET-Webanwendungen

### Code Coverage

- Ermittelt selbständig, wie hoch die Codeabdeckung während eines Testlaufs ist
- Hilft, die Qualität der Testfälle in Bezug zur Codebasis einzuschätzen und zu verbessern

## Event Log

- Zeichnet automatisch Ereignisse aus dem Event Log auf
- Fügt die Event-Log-Informationen dem Testergebnis hinzu

## Netzwerk-Emulation

- Während des Tests kann Netzwerklast künstlich erzeugt werden
- Hilft dabei, realistischere Testszenerarien aufzubauen

## Systeminformationen

- Sammelt automatisch Systeminformationen über die eingesetzte Testumgebung, u.a. Betriebssystemversion, Patch-Level, eingestelltes Gebietsschema, Bildschirmauflösung und vieles mehr

## Eigene Diagnosedaten

Es ist möglich, einen eigenen Diagnoseadapter zu bauen, der während der Testausführung relevante Daten der eigenen Anwendung sammelt. Das können z.B. eigene Events, Log Files oder Datenbankinhalte sein. Diese können an die Testausführung angehängt werden und stehen dann dem Testanalysten bzw. Entwickler zur Fehleranalyse zur Verfügung.

## Verbindung zur Testauswertung

Alle gesammelten Informationen werden automatisch an die Testergebnisse angehängt. Über die Verbindung der Testfallverfolgung und Testauswertung mit dem Bugtracking bzw. Fehlermanagement kann man mit Hilfe dieser Informationen sehr schnell den Fehler nachstellen und beheben. Damit können die „Develop-Test-Repro-Fix“-Zyklen massiv verkürzt werden.

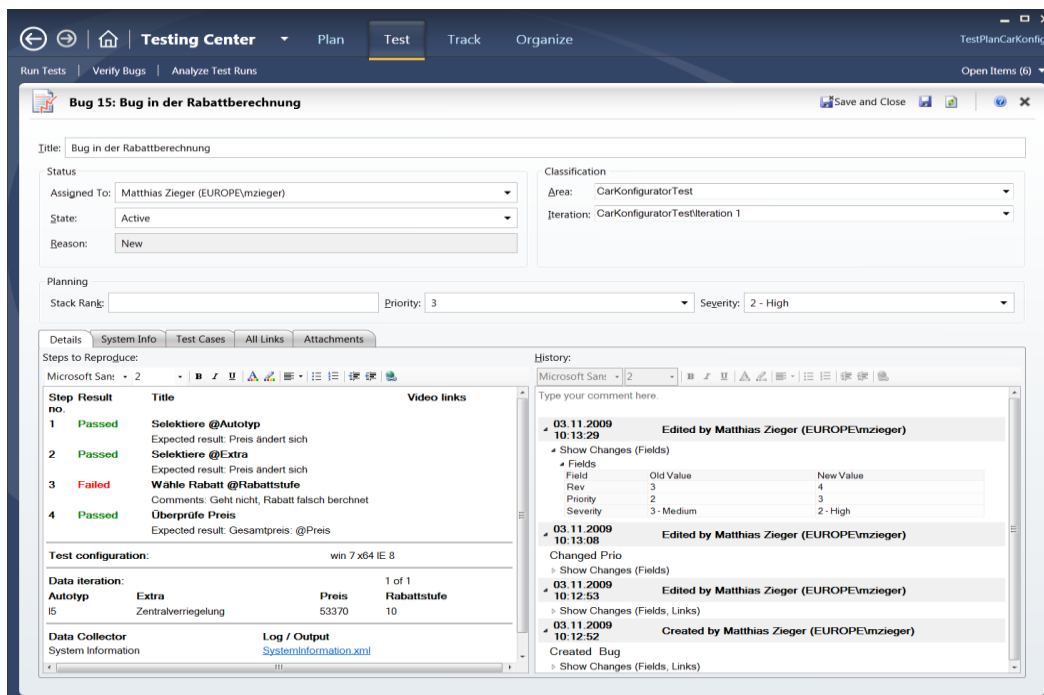


Abbildung 5: Fehlermeldung/Bug inkl. aller wichtigen Informationen

## Testautomatisierung

Die Testautomatisierung hat die Aufgabe, die Testeffizienz zu verbessern sowie für verlässlichere Testergebnisse zu sorgen. Eine große Rolle spielt sie insbesondere bei Regressionstests und agilen Praktiken wie z.B. „Continues Integration and Test“.

### Unterstützte Technologien

In Visual Studio 2010 können verschiedene Typen von automatisierten Tests erstellt werden. Dabei können unterschiedliche GUI/Oberflächentechnologien getestet werden, wie z.B. Windows Forms, WPF und Webanwendungen. Neben Webanwendungen werden Oberflächentechnologien unterstützt, die entweder Microsoft Active Accessibility (MS AA) oder Microsoft User Interface Automation (MS UIA) unterstützen.

### Unterstützte Testumgebungen

Folgende Testumgebungen werden im Moment unterstützt:

**Tabelle 1: Unterstützte Testumgebungen. Details unter [http://msdn.microsoft.com/en-us/library/dd380742\(VS.100\).aspx](http://msdn.microsoft.com/en-us/library/dd380742(VS.100).aspx)**

Konfiguration	Unterstützt
Operating Systems	Windows 2000 mit Service Pack 4 Windows XP mit Service Pack 3 Windows Server 2003 mit Service Pack 1 Windows Server 2003 mit Service Pack 2 Windows Server 2008 und Windows Server 2008 R2 Windows 7
Architecture	x86 und x64
.NET	.NET 2.0, 3.0, 3.5, und 4

Damit ist VS 2010 im Moment die erste Wahl für manuelle und automatisierte Tests von 64-bit-Applikationen, .NET 4-Anwendungen und auf Windows 7 bzw. Windows Server 2008 R2 basierenden Testumgebungen.

## Automatisierung mit Visual Studio 2010 in der Praxis

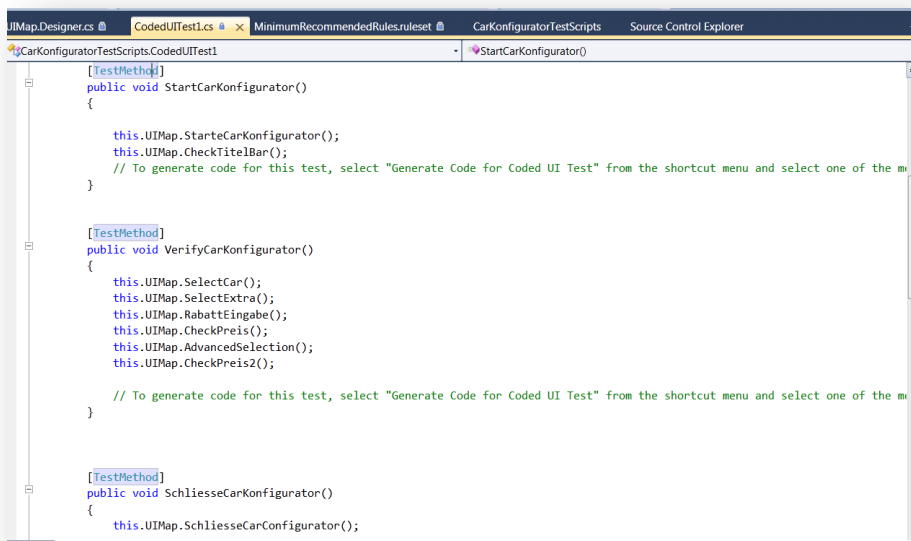
VS 2010 bietet die Möglichkeit, sogenannte Coded UI Tests zu erstellen. Das sind Testautomatisierungsprogramme bzw. Testtreiber. Als Technologie für diese Testtreiber wird das .NET 4 Framework genutzt, als Testprogrammiersprachen stehen C# und VB.NET zur Verfügung. Diese Coded UI Tests führen Aktionen auf der Benutzerschnittstelle aus und überprüfen, ob die vorhandenen Oberflächenelemente und deren Inhalte korrekt dargestellt werden.

Das bietet folgende Vorteile:

- Kein zusätzliches Know-how für eine neue Testsprache notwendig
- Nutzung sämtlicher Komfort-Features von Visual Studio 2010 für die Testautomatisierung
- Nutzung aller Möglichkeiten des .NET 4-Frameworks für die Testautomatisierung
- Integration mit der Team-Foundation-Server-Infrastruktur (u.a. für Validation Builds, Versionierung der Testtreiber, Fehlermanagement und Test-Reporting)

In VS 2010 gibt es drei Möglichkeiten, Testautomatisierungsskripte zu erstellen:

- Testfall-Recording
- Nutzung des Coded UI Test Builders
- Umwandlung der Action Recordings eines manuellen Tests



```

[TestMethod]
public void StartCarKonfigurator()
{
    this.UIMap.StarteCarKonfigurator();
    this.UIMap.CheckTitelBar();
    // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the m
}

[TestMethod]
public void VerifyCarKonfigurator()
{
    this.UIMap.SelectCar();
    this.UIMap.SelectExtra();
    this.UIMap.RabattEingabe();
    this.UIMap.CheckPreis();
    this.UIMap.AdvancedSelection();
    this.UIMap.CheckPreis2();

    // To generate code for this test, select "Generate Code for Coded UI Test" from the shortcut menu and select one of the m
}

[TestMethod]
public void SchliesseCarKonfigurator()
{
    this.UIMap.SchliesseCarKonfigurator();
}
    
```

Abbildung 6: Typischer "Coded UI"-Testfall

## Testfall-Recording

- Zeichnet alle Benutzerinteraktionen auf der Programmoberfläche (GUI) auf
- Erzeugt aus diesen Benutzerinteraktionen Testscripte in C# oder VB
- Validierungen für unterschiedliche Merkmale der Oberflächenelemente lassen sich hinzufügen

## Testautomatisierung mit dem Coded UI Test Builder

- Erlaubt sehr schnell, neue GUI-Elemente und Verifizierungen für einen neuen oder zu einem bestehenden automatischen Testfall hinzuzufügen
- Einfache Nutzung
- Erzeugt Code sowohl für die Ansteuerung der GUI-Elemente als auch für die Validierung

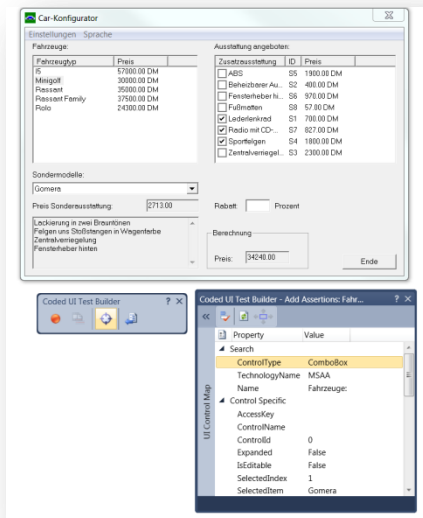


Abbildung 7: Der Coded UI Test Recorder und Validierung zusammen mit dem Testobjekt

## Umwandlung eines manuellen Tests in einen automatisierten Test

- Erzeugt automatisch Testscripte in C# oder VB aus den Action Recordings der manuellen Testausführung
- Tests können dann wahlweise manuell oder automatisiert ausgeführt werden
- Verkürzt die Zeit, um automatisierte Testscripte zu erzeugen

## Datengetriebene automatisierte Tests

Nachdem die logischen Testfälle erstellt sind, sollen diese natürlich mit verschiedenen Testdaten ausgeführt werden. Dazu bietet Visual Studio 2010 die Möglichkeit, eine Datenquelle zum Coded UI Test hinzuzufügen und diesen dadurch zu einem datengetriebenen Test umzuwandeln. Jede Zeile in der Datenquelle ist dann eine Iteration oder Ausprägung des logischen Testfalls. Das Testergebnis setzt sich dann aus den Einzelergebnissen jeder Testiteration zusammen.

## Last- und Stresstests

Das Ziel von Load Tests ist es, zahlreiche simultane Benutzerzugriffe auf eine Server-Applikation zum gleichen Zeitpunkt zu simulieren. Damit soll sichergestellt werden, dass die Anwendung auch bei starker Last korrekt wiedergegeben wird.

Visual Studio 2010 unterstützt dabei drei Modelle, Last zu generieren:

### Loadtest für Webanwendungen

- Ist eine etablierte Funktion der Visual Studio Team System 2008 Test Edition
- Einige neue Funktionen für VS 2010, z.B. neue Reports, Vergleich von zwei Testläufen
- Simuliert viele gleichzeitige Benutzer, die http-Requests gegen einen beliebigen Webserver ausführen
- Zahlreiche Parameter wie z.B. Browser, Bandbreite, Think-Time usw. lassen sich simulieren
- Testscripte lassen sich in C# oder VB konvertieren und damit die Mächtigkeit des .NET Frameworks für Lasttests nutzen

### Wiederverwendung von Unittests für Loadtestszenarien

- Ist ebenfalls eine etablierte Funktion der Visual Studio Team System 2008 Test Edition
- Ermöglicht Loadtestszenarien für nicht-webbasierte Komponenten, z.B. Datenzugriffs-, Workflow- und sonstige Kommunikationskomponenten
- Wiederverwendungseffekte führen zu Kosteneinsparungen

### Wiederverwendung von bereits automatisierten Tests im Lasttest

- Ist eine neue Funktion von VS 2010
- Automatisierte GUI-Tests lassen sich in Loadtestszenarien wiederverwenden
- Dadurch sind eigene Anwendungen als Testtreiber für Loadtestszenarien nutzbar

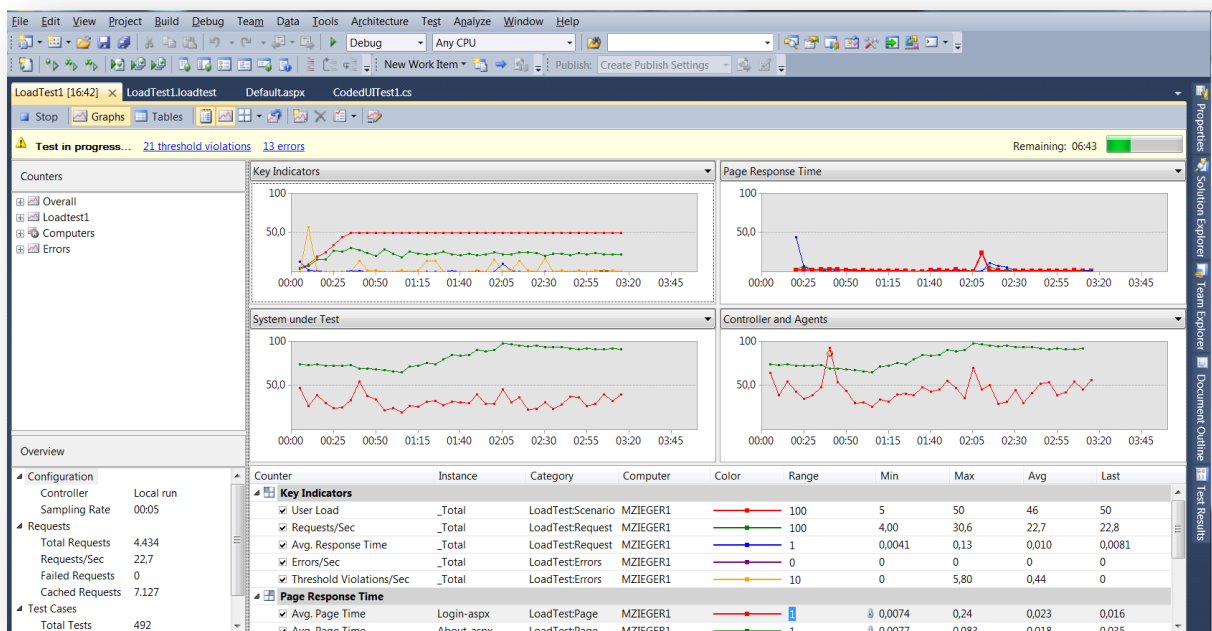


Abbildung 8: Monitoring eines Lasttests mit VS 2010

## Testvirtualisierung

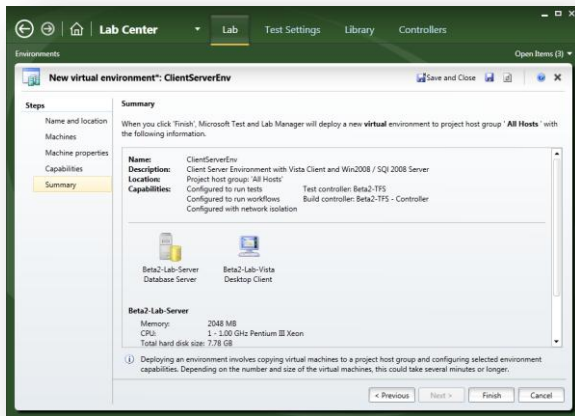


Abbildung 9: Definition neuer Testumgebungen im "Test and Lab Manager"

Visual Studio 2010 ermöglicht ein schnelles Deployment virtueller Testumgebungen zur Ausführung von Tests und zur Automatisierung von Builds. Damit kann man sehr schnell komplexe Konfigurationen erstellen, die weitestgehend der Produktivumgebung entsprechen. Die Testvirtualisierung ermöglicht durch die Nutzung der Snapshot-Technologien eine schnellere Reproduzierbarkeit von Fehlersituationen. Außerdem sind Testumgebungen schneller in definierte Zustände zurücksetzbar. Zusammen mit dem Testmanagement verringert sich massiv der Aufwand für professionelles Testen innerhalb des Application Lifecycle Managements.

Aufwand für professionelles Testen innerhalb des Application Lifecycle Managements.

### Testvirtualisierungspraktiken mit Visual Studio 2010

- Zurücksetzen von Umgebungen – stellt sicher, dass die Testumgebungen immer in einem definierten Zustand sind
- Für manuelle und automatisierte Tests nutzbar
- Snapshots von Testumgebungen – ermöglicht jederzeit, in verschiedene Testumgebungszustände zu wechseln
- In Fehlersituationen können Snapshots automatisch erzeugt und an Testergebnisse angehängt werden
- Klonen von Testumgebungen – durch die Netzwerkisolierung ist es möglich, Kopien von komplexen Testumgebungen zu erstellen, die aus mehreren virtuellen Maschinen bestehen können
- Integration mit Team Foundation Server und Buildmanagement – ermöglicht einen durchgehenden Workflow vom zentralen Build über automatisches Deployment bis hin zur Testautomatisierung
- Verwalten von Umgebungsbibliotheken – bietet die Möglichkeit für Deployments und Tests auf korrekten, definierten Umgebungen aus einer VM-Bibliothek

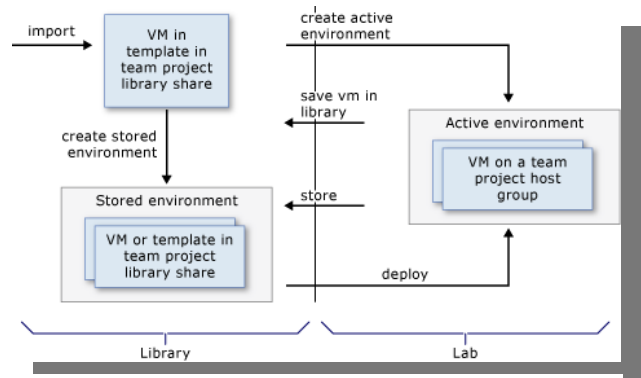


Abbildung 10: Zusammenspiel VM Bibliothek und Lab Manager

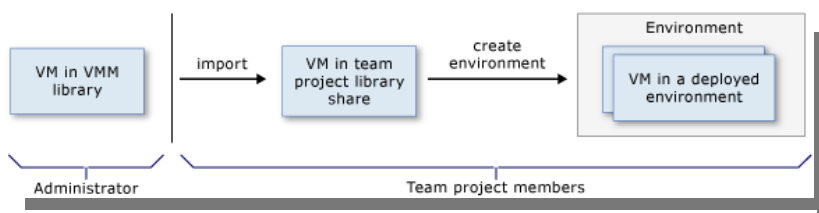


Abbildung 11: Workflow in der VM-Bibliothek

## Fehlerverfolgung, Auswertung und Reporting

Visual Studio 2010 und Team Foundation Server sind ein nützliches Paar, um die Qualität eines Softwareentwicklungsprojekts zu beurteilen. Zahlreiche Auswertungsmöglichkeiten stehen zur Verfügung.

- Welche SW-Builds haben Codeänderungen im Zusammenhang mit Bug Fixes, Change Requests, Anforderungen usw.
- Welche Testfälle müssen im Regressionstest basierend auf Codeänderungen erneut in einem Testplan zusammengestellt werden?
- Sind die Testfälle fertig definiert, und wenn ja, wie ist der Testfortschritt?

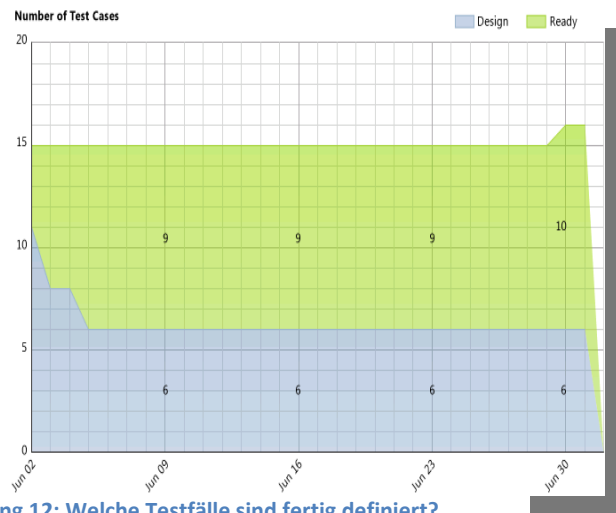


Abbildung 12: Welche Testfälle sind fertig definiert?

Es ist möglich, beliebige eigene Reports zu definieren, die verschiedene Aspekte des Application Lifecycle Managements umfassen können

- Anforderungsmanagement
- Changelogmanagement
- Versionierung
- Build- und Releasemanagement
- Testmanagement- und Testausführung
- Fehlerverfolgung

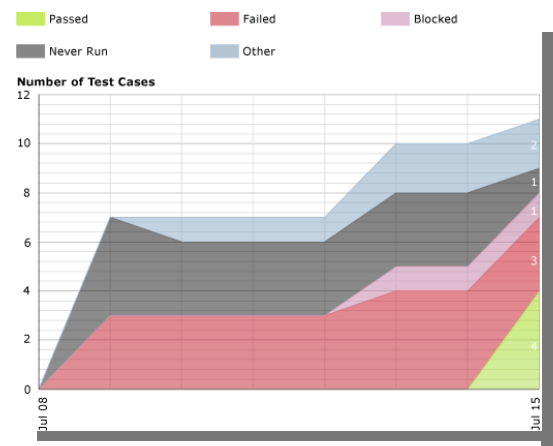


Abbildung 13: Testfortschritt pro Testfall

## Entwicklung und Test stärker verbinden

Entwickler können die im Test gefundenen Fehler viel besser als bisher nachvollziehen, indem die Testumgebung als Snapshot zum Zeitpunkt des Auftretens des Fehlers vorliegt.

Die neuen Features im Testmanagement, in der Datenaufzeichnung während der Testausführung und die Testvirtualisierung sorgen dafür, dass Fehler schneller analysiert und behoben werden können. Fehlermeldungen mit dem Status „nicht reproduzierbar“ werden damit seltener.

Dadurch sind Entwickler und Tester besser als jemals zuvor in der Lage, miteinander statt gegeneinander zu arbeiten. Das verbessert die Qualität der Software.

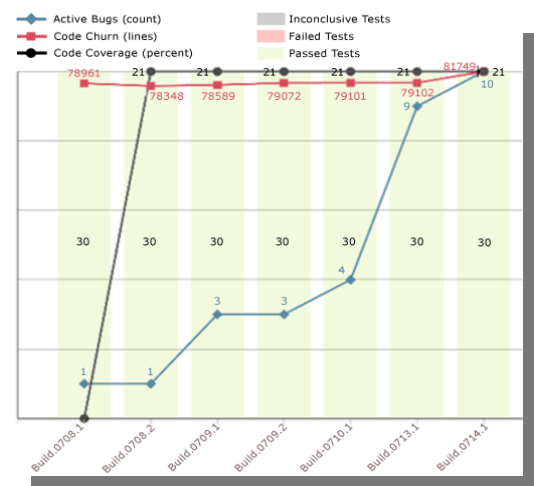


Abbildung 14: Aspekte Fehler, Codeänderungsrate, Codeabdeckung und Teststatus im Überblick

## Zusammenfassung: Qualität ist mehr als nur Testen

Typischerweise manifestiert sich die Qualität eines komplexen Softwaresystems in den während der Erstellung gelieferten und produzierten Artefakte. Dazu zählen insbesondere:

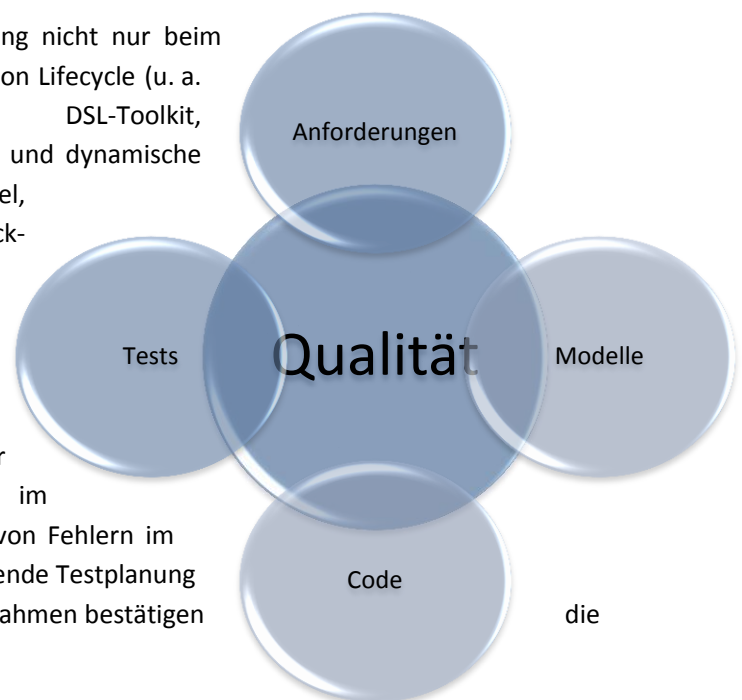
- Die Anforderungen als Basis für alle anderen Artefakte
- Abstrakte Prozessmodelle oder konkrete Implementierungsmodelle (UML, DSL, Datenbankmodelle) und deren Diagramme
- Quellcode, Skripte, Konfigurationselemente, Handbücher
- Buildskripte
- Testpläne, Testfälle, Testumgebungen und Testdaten

Alle diese Artefakte bzw. Arbeitsprodukte müssen qualitätsgesichert werden. In welchem Umfang dies geschieht, ist abhängig von den Rahmenbedingungen des Projekts, beispielsweise Compliance-Anforderungen (SOX, Basel II) oder Regularien (z.B. Medizintechnik ISO 13485, IEC 62304 und diverse FDA-Regeln).

Visual Studio 2010 unterstützt die Qualitätssicherung nicht nur beim reinen Testen, sondern über den gesamten Application Lifecycle (u. a. Anforderungsmanagement, UML-Modellierung, DSL-Toolkit, Architekturvalidierung- und Visualisierung, statische und dynamische Code-Analyse, Code Coverage Analyse, Check-In-Regel, Gated Check-

Ins).

Damit ist klar, dass Qualität nicht in ein System hineingetestet werden kann. Vielmehr ist der Sinn des Testens, letztendlich Qualität zu bestätigen oder zu widerlegen – und zwar möglichst früh im Anwendungsentwicklungszyklus. Eine Abwesenheit von Fehlern im Bugtracking-Werkzeug lässt meist auf eine unzureichende Testplanung oder falsche Testfälle bzw. Testdaten schließen; Ausnahmen bestätigen die Regel.



VS 2010 mit den Komponenten Microsoft Test and Lab Manager, Test Essentials sowie Team Foundation Server unterstützen dabei, hochwertige Testpläne mit Testfällen auf Basis von validen Anforderungen zu erstellen und diese Testfälle sehr effizient zu automatisieren.

Durch die hohe Integration von Testmanagement, Testautomatisierung sowie Testvirtualisierung und die damit verbundenen Möglichkeiten zur Fehleranalyse bzw. Reproduktion ist Visual Studio 2010 nicht nur das Tool der Wahl für Softwareentwickler, sondern ab Version 2010 auch für Tester, Testmanager und Testanalysten.

© 2009 Microsoft Corporation. Alle Rechte vorbehalten. Namen und Produkte anderer Firmen können eingetragene Warenzeichen der jeweiligen Rechteinhaber sein. Änderungen und Irrtum vorbehalten. Dieses Dokument dient ausschließlich Informationszwecken. MICROSOFT ÜBERNIMMT MIT DIESER ZUSAMMENFASSUNG KEINERLEI AUSDRÜCKLICHE ODER IMPLIZIERTE GEWÄHRLEISTUNG. Stand: November 2009.