Nico Orschel

(E-Mail: nico.orschel@aitgmbh.de)
ISTQB zertifizierter Software-Tester und Berater des AIT TeamSystemPro Teams. Hat sich u. a. anderem auf das automatische Testen mit dem Visual Studio Lab Management spezialisiert. Berät Unternehmen bei der Einführung und Anpassung des Visual Studio Team Foundation Server. Seine Erfahrungen vermittelt er zudem als Autor des TFS-Blogs, in Magazinen und in Vorträgen.

Kürzere Testvorbereitungsphasen durch integrierte Testlabore

Die aktuellen Release-Zyklen von Software-Projekten haben sich in den letzten Jahren eher verkürzt, u. a. weil agile Vorgehensmodelle sich in dieser Zeit rapide verbreitet haben und damit einhergehend Entwicklungs- und Testzyklen kürzer wurden. Dies stellt alle Projektbeteiligte vor neue Herausforderungen, wie z. B. viele Zwischen-Softwareversionen in Testumgebungen effizient bereitzustellen sowie die Sicherstellung der Lauffähigkeit von bereits implementierten Funktionalitäten durch automatisierte Tests. Im Vorjahresartikel (siehe [1]) konnte bereits der Austausch von Informationen zwischen Tester und Entwickler durch den Einsatz vom Team Foundation Server (kurz TFS) und dem Microsoft Testmanager (kurz MTM) optimiert werden. Der Fokus lag hierbei primär darauf, die Bereiche Testmanagement und Bugtracking über die gemeinsame Entwicklungsplattform Team Foundation Server zu integrieren und Bug-Reports, mit automatisch im Hintergrund erfassten Diagnoseinformationen, für eine bessere Reproduzierbarkeit zu erstellen. Ein unscheinbarer Bereich mit massivem Einsparpotenzial ist nach unserer Erfahrung der Betrieb und das Management von Testumgebungen. In der Testvorbereitungsphase schlummert zusätzlich noch eine Menge ungenutztes Potenzial bei der automatischen Bereitstellung der Testobjekte in die Testumgebungen. In diesem Artikel zeigen wir Ihnen Wege, wie Sie diese Herausforderungen effizient und effektiv mit dem Visual Studio 2010 Lab Management meistern können.

Einfache Bereitstellung von Testumgebungen

Bei vielen unserer Kunden beginnen die Testvorbereitungen mit der Bereitstellung von Testumgebungen durch eine dedizierte interne IT-Abteilung, die für die Bereitstellung und den Betrieb zuständig ist. Eine solche „einfache“ Bereitstellung von Testumgebungen kann aufgrund von etablierten IT-Betriebs- und Bereitstellungsprozessen gerade bei mittleren bis großen Unternehmen mitunter mehrere Wochen in Anspruch nehmen. Die Bearbeitungsdauer steht damit in direktem Widerspruch zu den Testzyklen bei agilen Entwicklungsprozessen. Für agile Teams ist eine potenzielle Dauer von drei bis vier Sprints schlichtweg untragbar. Unterstellt man dann noch die für einen agilen Prozess typischen sich verändernden Rahmenbedingungen und Anforderungen, dann bekommt ein Team die Testumgebung immer dann zur Verfügung gestellt, wenn es eigentlich schon zu spät ist. An diesem Punkt kann das TFS 2010 Lab Management sowohl den Tester als auch den IT-Administrator entlasten. Mit der TFS-Lab-Management-Funktionalität kann die Testabteilung virtuelle und physische Testumgebungen einfacher verwalten. Bei virtuellen Umgebugen bedeutet dies, dass man in Eigenregie ein-

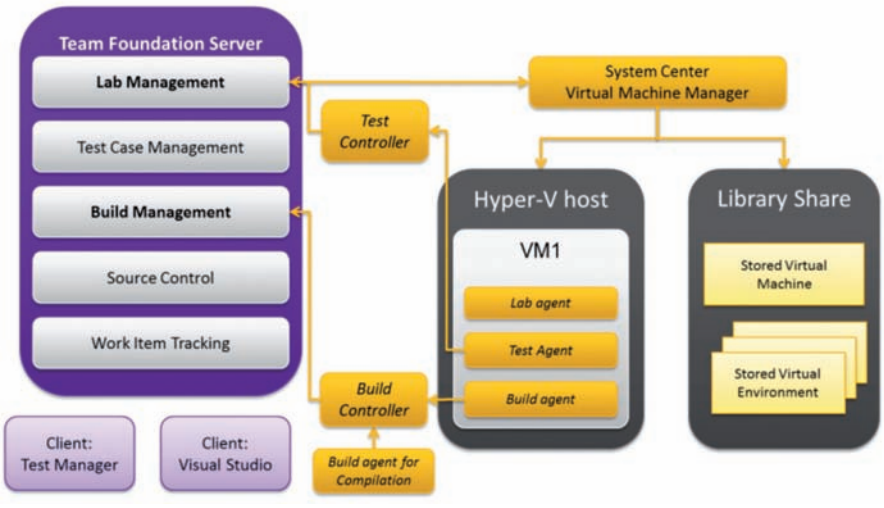

Abb. 1: TFS 2010 Lab Management Infrastruktur

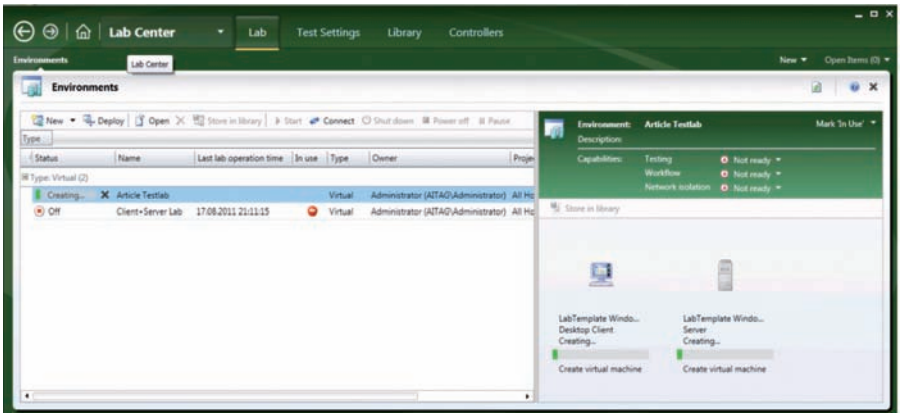

Abb. 2: Microsoft Testmanager 2010 Lab Center

fach und schnell virtuelle Maschinen erstellen, ändern und löschen kann. Im Folgenden werden wir uns auf den Typ der virtuellen Testumgebungen beschränken, weil nur diese den vollen Leistungsumfang zur Verfügung stellen. Über Unterschiede zwischen beiden Testumgebungstypen informiert das Whitepaper unter [2].

Die technische Basis für den Betrieb von virtuellen Umgebungen bilden Server mit der Microsoft Virtualisierungslösung Hyper-V (siehe Hyper-V Host in Abbildung 1).

Ein oder mehrere dieser Server werden über die Managementlösung System Center Virtual Machine Manager 2008 R2 (kurz SCVMM) an den TFS angebunden. Alle Hyper-V-Hosts aus Abbildung 1 werden zentral über SCVMM Library Shares mit Software und Vorlagen von virtuellen Maschinen und vorkonfigurierten Testumgebungen versorgt. Der SCVMM ist bereits ohne zusätzliche Lizenzkosten Teil der Lab-Management-Lizenz (Lizenzierungsdetails siehe [3]). Diese wiederum ist Teil der Programmpakete Test Professional (Microsoft Testmanager) und Visual Studio 2010 Ultimate.

Als Tester sehen Sie von diesen technischen Details im Hintergrund nichts, denn Sie verwalten diese Testumgebungen nur über ihren MTM in der Lab-Center-Perspektive (siehe Abbildung 2). Die zwei-

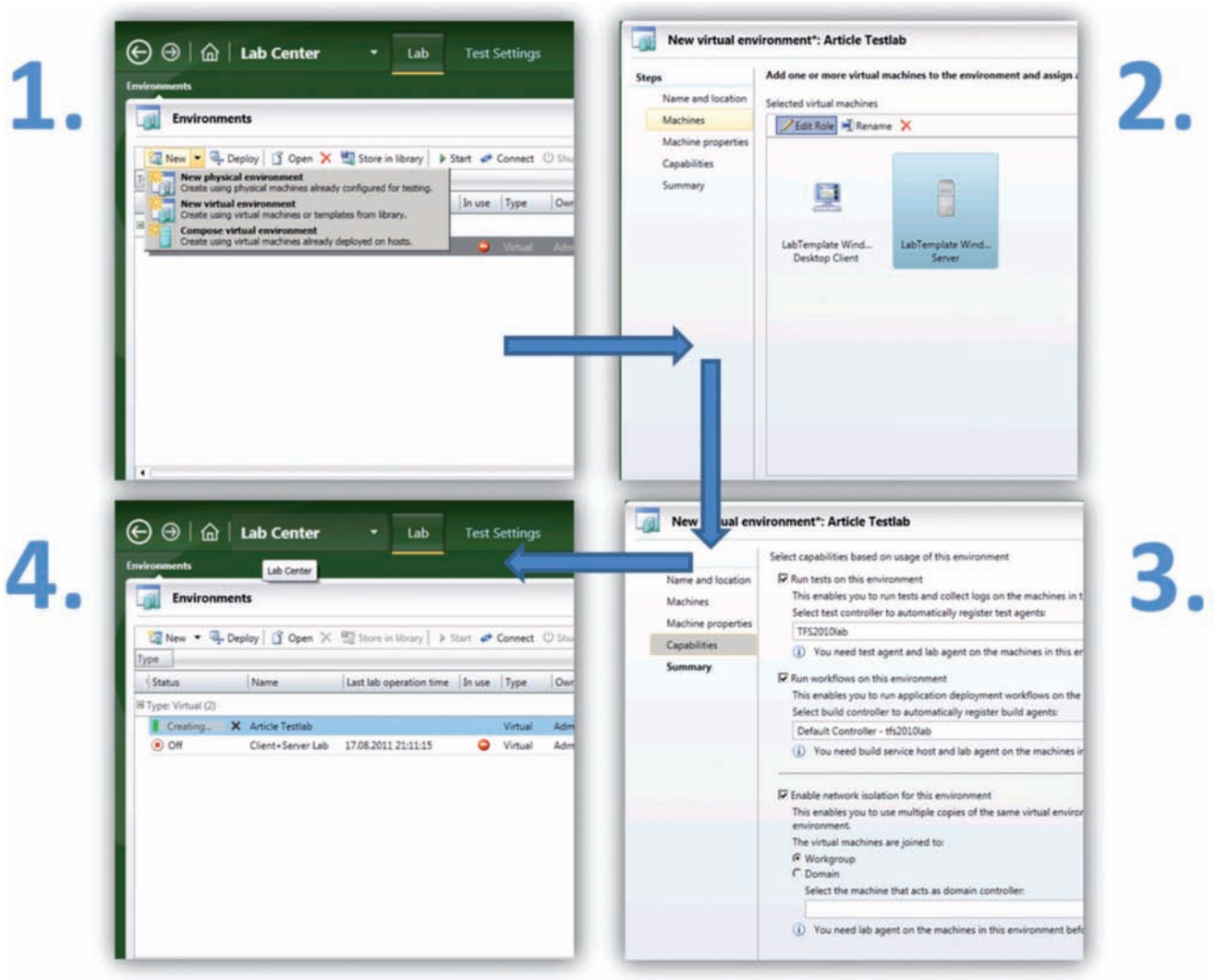

Abb. 3: Anlegen einer Testumgebung im MTM

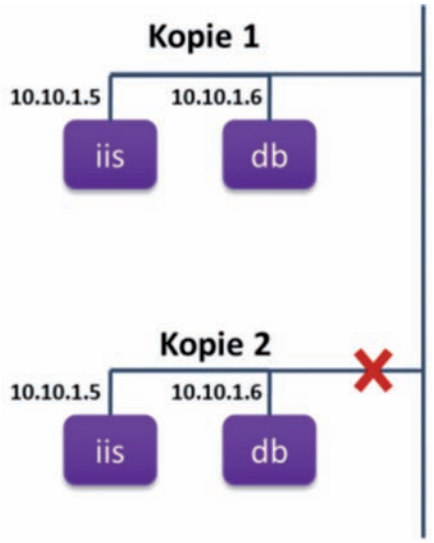

Abb. 4: Duplizieren einer Testumgebung ohne Network Isolation

te Perspektive des MTM ist das bekannte Test-Center für das Testmanagement und die Testausführung (siehe [1]).

Jetzt stellt sich hier natürlich die Frage: Welche Schritte sind für die Bereitstellung einer virtuellen Testumgebung mit dem MTM notwendig? Der Bereitstellungsprozess beginnt damit, dass die interne IT-Abteilung Vorlagen von virtuellen Maschinen nur einmalig erstellt (**siehe Library Share in Abbildung 1**). In diesen Vorlagen werden dabei Software-Agenten der TFS Plattform (Build Agent, Test Agent und Lab Agent) zur Fernsteuerung und weitere Software (in Abhängigkeit vom Testszenario) vorinstalliert. Anschließend werden diese generalisiert (Fachwort: syspreping) und dann über den SCVMM Library Share den MTM-Clients zur Verfügung gestellt, so dass später auf ihrer Basis nach Bedarf neue Maschinen erzeugt werden können. Nachdem man eine Vorlage in die Libary Share kopiert hat, müssen diese einmalig im Lab Center registriert werden. Bei diesem Registrierungsprozess werden Standardwerte für Computereinstellungen, wie z. B. Rechnername, Speicher, Administrator-Konto, Administrator-Passwort und Lizenzschlüssel hinterlegt, damit Anwender diese Daten nicht bei jedem Anlegen von Testumgebungen bei der IT-Abteilung erfragen bzw. doppelt und dreifach eintragen müssen. Ist die Vorlage erfolgreich im Lab-Center registriert, dann können Tester Testumgebungen, bestehend aus einem oder mehreren Rechnern, auf Basis dieser Vorlagen einfach zusammenstellen, so als würden sie mit Legosteinen bauen. **Abbildung 3** zeigt diesen sehr einfachen Prozess, wie die Tester mit nur vier Schritten Testumgebungen bereitstellen können. In der Abbildung wird in Schritt 1. zunächst der Umgebungstyp „Virtual Environment“ ausgewählt, um die zuvor beschriebenen Vorlagen als Basis für eine neue Testumgebung auszuwählen. In Schritt 2 hat die Auswahl der Vorlagen dann stattgefunden. Danach müssen im 3. Schritt noch die notwendigen Eigenschaften der Testumgebung, wie Workflow-, Testing- und Network-Isolation-Unterstützung aktiviert werden. Die Eigenschaften sind notwendig, um Deployment-Skripte auszuführen, sowie für automatische Tests und das Sammeln von Diagnosedaten. In Schritt 4 können Sie die Erstellung mit Finish starten.

Wichtig bei der Testausführung ist die Isolation unterschiedlicher Tester, um gegenseitige Auswirkungen von Tests zu minimieren. Diese Anforderung macht es notwendig, Testumgebungen für jeden beteiligten Tester zur Verfügung zu stellen. Unter normalen Umständen bedeutet dies, dass Sie den Prozess aus Abbildung 3 für jeden Tester x-mal wiederholen. Der Nachteil dieser Vorgehensweise ist gerade bei komplexen Testumgebungen der hohe Zeitaufwand und die Fehleranfälligkeit bei der Konfiguration und zusätzlichen Installationen. Wäre es nicht schön, wenn Sie den Aufwand der Testumgebungsvorbereitung nur einmal tätigen müssten und anschließend einfach für jeden Tester ein exaktes Duplikat anzufertigen? Wenn man

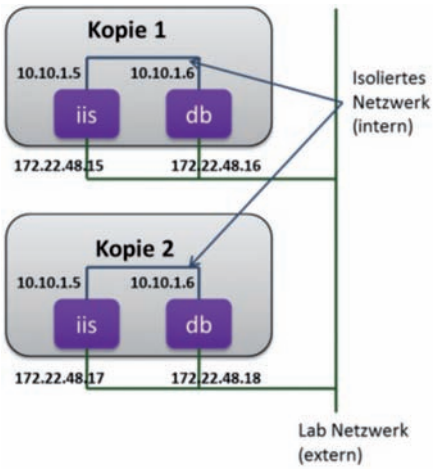

Abb. 5: Duplizieren einer Testumgebung mit Network Isolation

einem IT-Administrator diese Frage stellt, dann wird man sicherlich die Antwort erhalten, dass dies technisch nicht möglich ist. Man hört in dieser Situation technischen Ausführungen wie u. a., dass Rechnername und IP Adresse im Netzwerk stets eindeutig sein müssen (**siehe Abbildung 4**).

Genau für diesen Anwendungsfall wurde beim TFS 2010 Lab Management die Funktion „Network Isolation“ eingeführt. Mit Hilfe eines Dienstes – dem sogenannten Lab Agent – in den virtuellen Maschinen (**siehe Abbildung 1 und Abbildung 5**) können diese voneinander isoliert betrieben werden, obwohl es sich um Klone handelt. Das Duplizieren ist dank dieser Funktion so einfach wie das Kopieren einer CD.

Ein weiterer positiver Nebeneffekt dieser Funktion besteht darin, dass Sie somit der Entwicklungsabteilung Duplikate ihrer Umgebungen zur Verfügung stellen können. Im vorherigen Artikel wurde bereits das Konzept der Rich-Bugs, d. h. Bugs mit zusätzlichen Diagnoseinformationen vorgestellt. Trotz dieser vielen Informationen lassen sich Fehler in Entwicklungsumgebungen nicht immer nachvollziehen und man benötigt als Entwickler besser die ursprüngliche Testumgebung des Testers. Mit der Duplizierungsfunktion kann man als Tester einfach die Umgebung duplizieren und dem Entwickler zur Verfügung stellen. Ein Beispiel für dieses Szenario im Falle eines Fehlers zeigt **Abbildung 6**. Mehr noch, es lässt sich ein Link auf den Snapshot der virtuellen Maschine zum Zeitpunkt des Fehlereintritts anhängen und dem Entwickler somit die gesamte „eingefrorene“ Testumgebung übermitteln.

Kompilieren, Ausliefern und Testen

Nachdem die Testumgebungen durch den MTM und Lab Management bereitgestellt wurden, bleibt noch der Punkt der Bereitstellung von Testobjekten durch die Entwicklung offen. Bei der Bereitstellung von Testobjekten folgt das aktuelle Kapitel der Methodik „Continous Delivery“.

Continous-Delivery-Prozesse (im Folgenden kurz CD genannt) bestehen aus drei Phasen: Übersetzen der Lösung, Ausliefern oder Installieren der Lösung in einer Zielumgebung und abschließend die Ausführung von automatisierten Tests. An dieser Stelle fällt sofort die Ähnlichkeit zu dem weitverbreiteten Begriff Continous Integration (im Folgenden kurz CI) auf. CI

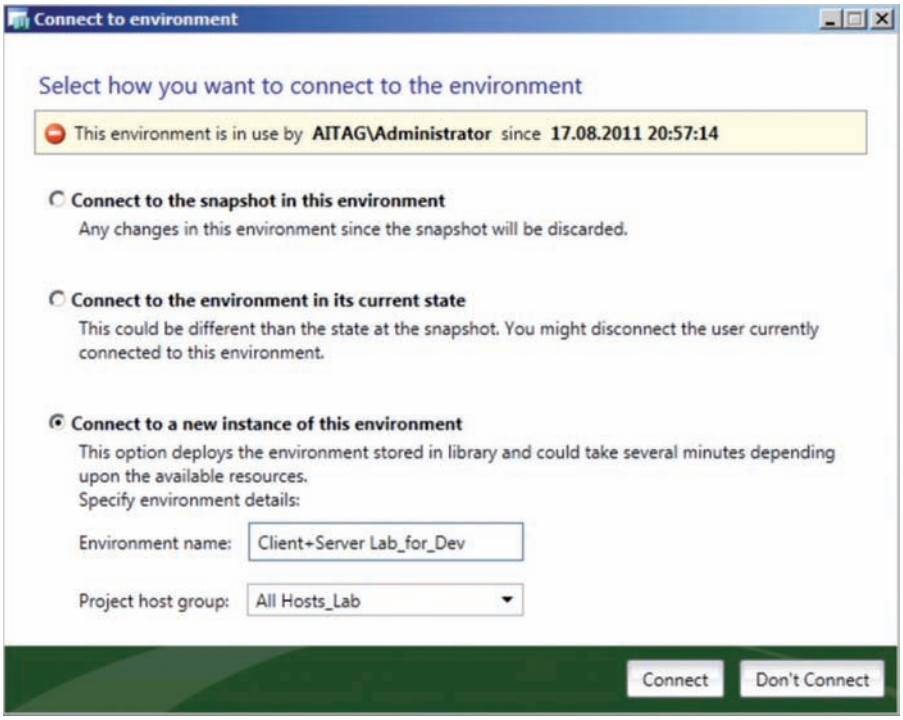

Abb. 6: Wiederherstellung einer Testumgebung im Fehlerfall

besteht aus der Prozesssicht aus zwei Hauptphasen: Übersetzen der Lösung und anschließendes Ausführen von Unit Tests. Die Unterschiede liegen dabei zum einen darin, dass CI-Build-Prozesse keine Software in Zielumgebungen ausliefern und zum anderen, dass nur Modul- bzw. Komponententests (Unit Tests) in Isolation ausgeführt werden. CI verfolgt im Gegensatz zum CD das Ziel, schnell Feedback an die Entwickler zu liefern. CD hingegen führt die Tests an Testobjekten in den jeweiligen Zielumgebungen aus. Die Zielumgebungen sind dabei an den Produktivumgebungen angelehnt. Desweiteren setzen die Tests eine lauffähige Software für die Ausführung voraus. Die auszuführenden Tests haben als Fokus die Testphasen Systemtest oder Akzeptanztest und sind dadurch szenario-orientiert – prüfen also potentiell das Software-Produkt als Ganzes. Zusammengefasst stellt CD erstmals die Verbindung zwischen Testumgebung und Release-Prozess her. Es ist damit ein „CI plus".

Der zuvor beschriebene CD-Prozess ist beim TFS 2010 über eine spezielle Build-Workflow-Vorlage mit dem Namen LabDefaultTemplate.xaml realisiert, welche bereits alle notwendigen Basisschritte zur Verfügung stellt. Microsoft hat mit dem Workflow einen Release-Prozess entsprechend der Darstellung in Abbildung 7 realisiert, welcher neben der Infrastrukturthematik aus dem vorherigen Abschnitt zum Begriff Lab Management zusammengefasst wird.

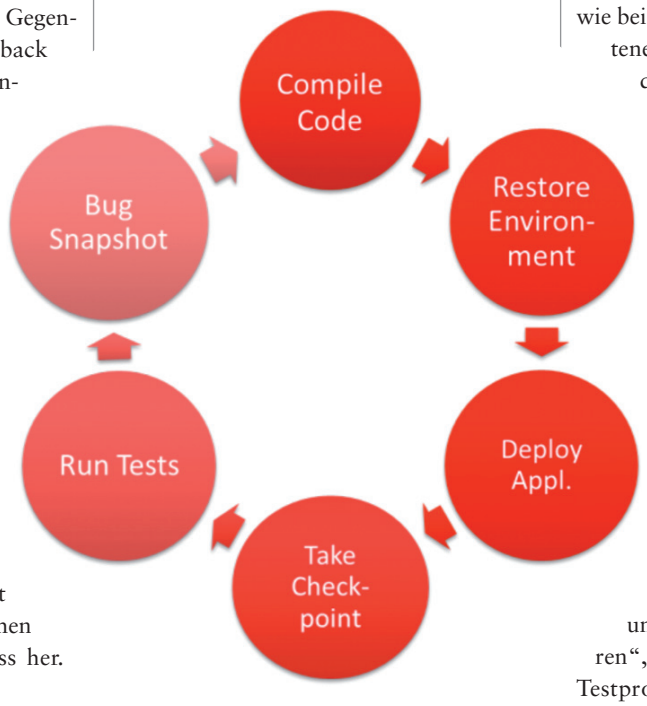

Abb. 7: Lab Management Workflow

Ein Lab Management Workflow sorgt zu Beginn für die Übersetzung des jeweiligen Codes. Technisch gesehen führt der Lab Management Workflow einfach einen bereits existierenden Release-Build-Prozess der Entwicklung aus, so dass man für Testumgebungen und spätere Releases nur einen Build-Prozess pflegen muss. Nach der Kompilierungsphase stellt der Workflow eine Verbindung zu den Testumgebungen her und setzt diese im Fall von virtuellen Testumgebungen auf einen definierten Stand (Fachwort: Abbild oder Snapshot) zurück. Nachdem die Verbindung hergestellt wurde, wird die Anwendung über Skripte oder Installationsprogramme in die Testumgebung ausgerollt. Die Art und Weise des Rollout-Verfahrens ist abhängig vom jeweiligen Artefakttyp (Programm, Website, Datenbankskript, Testdaten). Nach dem Rollout wird im Fall von virtuellen Testumgebungen ein weiteres Abbild der Testumgebung erzeugt. Dieses spezielle Abbild hat den Vorteil, dass Tester und Entwickler ihre Umgebungen zu jedem Zeitpunkt in einen nicht durch Tests beeinflussten Zustand zurücksetzen können. Werden Testobjekte in der Testumgebung bereitgestellt, starten anschließend die automatischen Tests in der Testumgebung. Die hier ausgeführten Tests sind nicht mehr wie bei CI auf die Modul- oder Komponentenebene begrenzt, sondern die Anwendungen können auch direkt ausgeführt werden. Zum Testen von Oberflächen (wie z. B. WinForms, WPF, Silverlight, Webanwendungen im IE und Firefox) kann man hier das neue CodedUI Framework einsetzen. Die auszuführenden Tests werden über Testfälle, welche über Testpläne und Testsuiten im TFS 2010 strukturiert sind, ausgewählt (Testfallmanagement mit MTM siehe [1]). Wenn ein Fehler auftritt bzw. ein Testfall nicht erfolgreich war, dann kann der Tester die Testumgebung im Fehlerzustand „einfrieren", so dass der Entwickler neben den Testprotokollen auch eine Testumgebung zur Fehlerdiagnose zur Verfügung hat.

Auf den ersten Blick scheint der beschriebene Workflow recht kompliziert zu konfi-

gurieren, er lässt sich aber dank der Assistenten im Visual Studio 2010 schrittweise einrichten (technische Details siehe [2]).

Fazit

Der Artikel zeigt Ihnen Wege, wie Sie mithilfe des TFS 2010 Lab Management die Testvorbereitungsphase optimieren können. Die Bereitstellung von Testumgebungen im Baukastenprinzip kann mithilfe des MTM durch Tester stark vereinfacht werden. Für die Tester wird dabei kein neues Werkzeug benötigt, sondern es wird nur eine andere Perspektive des MTM (Lab Center) genutzt. Die interne IT-Abteilung kann durch diese neue Arbeitsweise stark entlastet werden, weil sie nicht mehr jede Umgebung individuell aufsetzen, sondern jetzt nur noch Vorlagen für virtuelle Maschinen sowie die entsprechenden Host-Systeme zur Verfügung stellen müssen. Gerade bei agilen Projekten gehören Wartezeiten von mehreren Sprints dank dieser neuen Möglichkeiten der Vergangenheit an. Zusätzlich haben Tester nun die Möglichkeit, wenn einmal eine komplexe Umgebung aufgesetzt ist, diese einfach durch die Duplizierungsfunktion (Network Isolation) ähnlich wie eine CD in kürzester Zeit zu kopieren. Umfangreiche Konfigurations- und Installationsarbeiten können so eingespart werden und Tester können sich auf die Testspezifikation und -ausführung konzentrieren. Die Lab-Management-Infrastruktur hat zusätzlich noch den positiven Nebeneffekt, dass Tester ihre Testumgebungen mit den Entwicklern „teilen“ können. Die Entwickler können dadurch beim Testen und beim Nachvollziehen von Fehlern auf die gleichen Testumgebungen zugreifen. Durch die enge Verzahnung von Build-Prozessen der Entwicklung mit den Testumgebungen können neue Releases schnell, regelmäßig und automatisiert in den Testumgebungen zur Verfügung stehen, so dass manueller Aufwand sowohl auf Seiten der Qualitätssicherung- und Entwicklungsabteilung eingespart werden kann. Das TFS Lab Management umfasst bereits einen Standard-Workflow, welcher der Methodik „Continuous Delivery“ folgt. Konkret bedeutet dies, dass er bereits die „Continuous Delivery“ – Phasen Kompilieren, Ausliefern und Testen unterstützt. Im Rahmen des Schrittes Testen wird bereits eine erste Eingangsprüfung über automatisierte Tests durchgeführt, so dass nur Programme mit bestimmten Qualitätsstandards ihren Weg in die manuellen Testphasen finden.

Referenzen

[1]: Ausweg aus der Kommunikationskrise oder das Ende von „Bei mir funktioniert's“?: http://www.sigs.de/publications/os/2010/Testing/orschel_OS_TESTING_2010.pdf

[2]: Whitepaper Lab Management: http://www.aitgmbh.de/labmngwhitepaper

[3]: Visual Studio 2010 and MSDN Licensing White Paper: http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=13350

[4]: Continuous Delivery: Reliable Software Releases Through Build, Test, and Deployment Automation, Addison-Wesley Longman, ISBN: 0321601912