Continuous Integration & Automated Testing in a multisite .NET/Cloud Project

Entwicklertag 2013 – Karlsruhe

5-Jun-2013
Vladislav Kublanov
Speaker
Vladislav Kublanov – Tata Consultancy Services (TCS)

- Studied Computer Science at RWTH-Aachen, Masters degree 1999
- 14 years of SW development experience in international projects
- 8 years of Agile experience as ScrumMaster and CI Specialist
- Contributed to introduction of Agile at Nokia, NSN and TCS
- Since 2008 SW Developer and ScrumMaster at TCS Düsseldorf

Continuous Integration & Automated Testing in a multisite .NET/Cloud Project
Content

1. Mission
2. Infrastructure Overview
3. SW development with Sitecore & TDS
4. Continuous Integration process
5. Automated Testing approach
6. „Go Live“ and Branching strategy
7. Technologies & Tools
8. Learnings
9. Discussion
Mission

For a big, multi-site Web Portal Project, executed in Agile mode by TCS, it was required to provide infrastructure and to design and set-up a Continuous Integration & Automated Testing process with the following aims:

- to ensure fast quality feedback during development and
- to have ALL components integrated into a potentially shippable product at the end of every sprint.

It was also expected to achieve cost savings by means of process automation.

Project context

- Web Portal, based on Sitecore Content Management System (Sitecore CMS)
- 1 year to develop and to enable “Go Live” of the release 1.0
- Project size – up to 60 people, distributed over 3 sites:
  - Customer site in Germany
  - TCS Delivery Center in Düsseldorf, Germany
  - TCS in Delhi, India
Challenges

**Accessibility**
- Development and testing environments accessible from all three sites

**Scalability**
- Scalable development and testing infrastructure
- Easy to add and set-up new environments

**Security**
- All machines accessible via secure connection

**Technologies**
- Development and runtime based on Microsoft Technologies

**Incremental development of Sitecore CMS content**
- In course of the project, customer provides the CMS content data incrementally
  - Phase 1: no CMS content data available
  - Phase 2: CMS content data starts to grow

**“Go Live” and maintenance**
- After release 1.0 goes live, development of the next release 2.0 to run in parallel with the maintenance of 1.0
Project Infrastructure Overview

Amazon Cloud

TCS Nearshore
- Up to 10 Development Environments
- 

VPN

TCS Offshore
- Up to 35 Terminals
- 

VPN

Customer
- Up to 15 Development Environments
- 

VPN

CMS content Development Environment

Live System

Up to 15 Development Environments

Development Environment 1
- IIS
- Visual Studio

Database Server

Test Server 1
- IIS

Test Server 2
- IIS

Development Environment 35
- IIS
- Visual Studio

TFS Server

SCM

Build
Live environment setup

CMS content editor

Create CMS content

Content Management Server

End-user

View web page content

Content Delivery Server

Core

Master

Web

Publish for pre-view

Publish to live system

Core

Live

TATA CONSULTANCY SERVICES
Experience certainty.
SW development with Sitecore & TDS

Phase 1: *production CMS content data not yet available*

1. Develop CMS content:
   - Sitecore templates
   - Sitecore items

2. Synchronize CMS content with Visual Studio

3. Check-in code, tests & CMS content

4. Build

5. Deploy TDS.update package

**Local development environment**

- **C# Projects**
  - -> C# source code

- **Unit Test Projects**
  - -> Unit tests

- **TDS Projects**
  - -> Sitecore templates
  - -> Sitecore items

**Amazon Cloud**

- TFS Server -> SCM
- Test Server -> IIS 2.0
- SQL Server

**TATA CONSULTANCY SERVICES**

Experience certainty.
SW development with Sitecore & TDS
Phase 2: production CMS content data starts to grow

1. Develop CMS content: Sitecore templates and Sitecore items
2. Synchronize CMS content with Visual Studio
3. Check-in code, tests & CMS content
4. Build
5. Deploy
6. Develop production CMS content data
7. Export production content as database backup file
8. Deploy TDS.update package on top of production CMS content

CMS content development environment at customer site

Local development environment

Amazon Cloud

C# Projects
- C# Source Code
Unit Test Projects
- Unit Tests
TDS Projects
- Sitecore Templates
- Sitecore Items
CI & AT Process Overview (Phase 1)

1. "Gated" check-in code, tests & CMS content

2. Compilation & unit testing

3. Automated deployment, Full Publishing & Selenium testing

4. Installation & test reports (via e-mail)

5. GREEN BUILD (potentially shippable product)

In phase 1
- Production CMS content data not available, so that Sitecore publishing is fast
  - Short installation time
- Little product functionality available, so that Selenium test suite is small
  - Test automation support is cheap

TFS Server

Test Server

SQL Server

Shared directory with green builds: `\TFSServer\SharedDrive\GreenBuilds`
CI & AT Process Overview (Phase 2)

1. "Gated" check-in code, tests & CMS content

2. Compilation & unit testing

3. "Gated" check-in code, tests & CMS content

4. Installation & test reports (via e-mail)

5. GREEN BUILD (potentially shippable product)

In phase 2

- Production CMS content data started to grow, so that Sitecore publishing gets slower

  Longer installation times

- Product functionality grows and changes faster, new tests are added and existing tests adjusted more frequently

  Costs of test automation increase

---

TFS Server

- SCM

Developer's PC

Visual Studio

Test Server 1

- Automated Deployment
- Partial publishing & Smoke Selenium Testing

Every 2h

Test Server 2

- Automated deployment
- Full publishing & Smoke Selenium testing

Daily

Testing Team

Manual testing

New

Weekly

Automated integration of production CMS content Data

SQL Server

New

Shared directory with green Builds: \TFSServer\SharedDrive\GreenBuilds

New

---

In phase 2

- Longer installation times
- Costs of test automation increase
Automated Deployment process

Developer's check-in triggers a new build

A scheduled job fetches and starts installation of the new build

Stop Web Page instance from last installation running in the IIS

Prepare filesystem components of Sitecore: Website and Data

Initialize database component of Sitecore using remote access

Deploy new installation package

Configure the newly installed package

Launch automated Selenium testing

Backup the build and send build status e-mail

Detach old and restore original databases

Start Web Page in the IIS

TFS Server

Test Server

DB Server
Development of Selenium tests

- Recording initial set of Selenium tests with Selenium IDE
- Extending and enhancing recorded tests manually (e.g. provide user-extensions.js to store UI element ids)
- Check-in Selenium tests into TFS
- Copy Selenium tests to the test environment as part of the installation package
Automated Testing – Test Execution

- As the very last step, installation script starts an RDP session (mstsc.exe)
- RDP session, when launched, is pre-configured to execute Selenium server with the test suite as parameter
- After execution of Selenium, installation script parses browser log files for errors
- Installation script sends an e-mail to the project team informing of success or failure
“Go Live” of release 1.0 & development of release 2.0

Establishing of a “maintenance” branch for the released 1.0 software

Trunk

Release 1.0 development

Development fixes for 1.0 software

Release 2.0 development

Merging to trunk

Merging to trunk

Merging to trunk

Maintenance branch
Continuous Integration after “Go Live”

**TFS Server**
- After each check-in on the Trunk
  - Compilation & unit testing

**Test Server**
- Automated deployment & Selenium testing
- Automated deployment & Selenium testing

**Live System**
- IIS
- SQL Server

**Testing Team**
- Functional testing of both releases
- Manual deployment of 1.0 update

**Trunk**
- v2.0 development
- v1.0 maintenance

**Maintenance branch**
- After each check-in on the Maintenance branch
- Compilation & unit testing
Technologies & Tools

**Development & Testing**
- Windows Server 2008 R2
- Team Foundation Server 2010 (TFS) as SCM and Build Server
- „Gated check-in“ feature of TFS to reject compilation breaking check-ins
- Visual Studio 2010 as Integrated Development Environment (IDE)
- Moles framework for unit testing
- ASP.NET v4
- Internet Information Services 7 (IIS 7)
- SQL Server Express 2008 R2
- Sitecore 6.5 - .NET based Content Management System (CMS)
- Hedgehog TDS (Team Development for Sitecore) to manage Sitecore objects in Visual Studio and SCM
- Package Installer tool of Sitecore to deploy built packages to Sitecore
- Powershell scripts to automate deployment and testing
- Remote Desktop Protocol (RDP)
- Selenium

**Runtime**
- Windows Server 2008 R2
- ASP.NET v4
- IIS 7
- SQL Server 2008 R2
- Sitecore 6.5
Financial benefits of process automation

Automated deployment and configuration
- One-off effort spent on automation was much less compared to effort which would have been spent for manual deployments/configuration.

Earlier detection of compilation failures
- „Gated Check-in“ feature ensured rejection of compilation breaking check-ins.

Testing effort reduced
- Automated Tests saved effort which the testing team would otherwise have spent on testing of the regression functionality.
Learnings

Microsoft Technologies
- Mature and reliable
- “Gated check-in” feature of TFS
- When going for Microsoft – all tools must come from Microsoft

Amazon Cloud
- General impression - positive
- Amazon HW located in Singapore
- Good experiences with OpenVPN
- Fast and stable connection to and within the cloud
- Easy set-up new, and move existing environments to e.g. stronger HW
- More expensive on long term than buying own HW, but provides more flexibility

Sitecore tools
- Sitecore 6.5 is a solid commercial product, stable and well documented
- Sitecore Package Installer enables deployment of TDS packages
- TDS - although tricky to use – enables CMS content development with VS in a big project
Summary

- The project infrastructure incorporating Cloud, OpenVPN and TDS, enabled the big, distributed project team to develop a Sitecore based Web Portal.

- Fast quality feedback due to Selenium-based Test Automation, usage of the „Gated Check-in“ feature provided by the TFS and due to the scalability of the testing infrastructure.

- Availability of the potentially shippable product at any point of time during the project due to Continuous Integration of the software with the CMS Content Data provided by the customer.

- Cost savings due to automation of the CI process.
Thank You

Vladislav Kublanov
SW Developer & ScrumMaster
TATA CONSULTANCY SERVICES
Tata Consultancy Services Deutschland GmbH
Heltorfer Straße 1, D-40472 Düsseldorf
Mobile +49 173-678 0583, Fax +49 211 91319-99
Phone +49 211 91319-161
vladislav.kublanov@tcs.com website: www.tcs.com