The evolution of Microsoft’s solution for Application Lifecycle Management

Team Foundation Server
# Table of Contents

- Introduction ................................................................. 2
- The basics of ALM and Microsoft’s journey ............................................. 3
- Heterogeneous Software Development ..................................................... 6
- A fully integrated testing experience with TFS 2010 ...................................... 9
- About TFS 2012, the Cloud, Git and the new shipping cadence ......................... 13
- TFS 2013 and Visual Studio Online .......................................................... 18
- Conclusion ........................................................................................ 23
- About Me .......................................................................................... 25
Introduction

Software development is so much more than only writing technology specific code (.NET, Java, PHP, Mainframe, …). Applications are also becoming more complex due to numerous distributed services and the boundaries of an application are becoming less clear. Nearly all large enterprises struggle with communication, collaboration and cultural gaps between business users who drive the competitive need for software development, software teams who create the software, and the IT Operations team who manages application deployment and maintenance. IT departments are still often siloed internally with poor hand-offs between development teams and inconsistent approaches to core project life cycle phases and roles (architect, project manager, developer, database administrator, tester, configuration/release manager, …). This negatively affects design, quality, code management and the final deployment of software applications into a production environment.

On top of these issues, it’s quite interesting to see how different technology stacks are coming together in the bigger traditional enterprises. Product features or requirements are not only managed anymore for a specific technology stack but go way beyond specific implementation choices. A complete release/solution can easily consist of a customer facing website written in .NET which connects to Java Web Services for business transactions. The data used for processing could easily come from multiple other applications/sources (don’t forget the existence and importance of many Mainframe applications!). And I didn’t mention yet the different options that pop up if you want to offer a mobile solution on various devices and platforms. At a time of increasing costs, competitive pressure and greater complexity for business requirements, organizations require intuitive, accessible and integrated products. Most organizations have highly constrained resources and add outsourcing to the mix, which further drives the need for better prioritization and metrics for measuring IT success. In addition, regulatory compliance legislation is pushing organizations towards greater rigor with regards to application lifecycle management. On the other side of the spectrum we can identify the need for continuous delivery. Business expects IT to deliver new incremental features as fast as possible. Time-to-market has never been as important as now and can create an important competitive advantage. But let’s face it, many organizations are still far away from the continuous delivery dream. It’s not something which can be easily adopted over the week-end.

In this article I will write about the interesting evolution of Application Lifecycle Management (ALM) and the increased importance of (heterogeneous) software development processes. What I described above about continuous delivery should obviously be the ultimate goal to achieve, but it’s important to set out a realistic roadmap. Don’t run before you can walk and take your time to introduce important changes in your organization. A well-defined ALM strategy is essential to achieve the goals of the IT organization while closely aligning the direction of the business. The key for success is finding a balance between processes, tools and people.
The basics of ALM and Microsoft’s journey

According to Wikipedia:

*Application lifecycle management (ALM) is the product lifecycle management (governance, development, and maintenance) of application software. It encompasses requirements management, software architecture, computer programming, software testing, software maintenance, change management, project management, and release management.*

For each specific technology, there exist one or more dedicated ALM tool(s) in the market. Only a few tools offer true support for cross-technology Application Lifecycle Management. Serena offers a number of related tools for requirements management, version control and release management. Serena Dimensions CM integrates for example with many popular application development tools straight out of the box like Visual Studio (.NET) and Eclipse (Java). Another example is Rational Team Concert (RTC) from IBM which is positioned as a lean collaborative lifecycle management solution. IBM also offers a rich RTC client plugin for Eclipse and Visual Studio. Atlassian is an Australian software company which focuses on issue tracking, collaboration and other related software development products to work faster and smarter, together. Their tools are very popular in the Java community, but they also offer a Visual Studio Connector to interact with JIRA issues and Bamboo builds right from Visual Studio. The Atlassian tools in combination with a powerful version control system like Git or Mercurial can also be seen as a competitor in the cross technology ALM space.

But for now I want to highlight Microsoft’s Application Lifecycle Management solution. Microsoft Visual Studio 2013 was officially released on November 13, 2013 at the Visual Studio 2013 Virtual Launch Event. The RTM version has been made available for a few weeks already and can be downloaded at http://www.microsoft.com/visualstudio/eng/downloads.

Some history about Team System

Let’s get back in time when it all started. Microsoft released in November 2004 (9 years ago) their first version of an integrated application lifecycle management solution (codename Burton): Microsoft Visual Studio Team System 2005. Before, Visual Studio was only a dedicated tool for developers/coders. This shifted completely with the release of Team System where multiple stakeholders were involved in the software development process.
Microsoft studied the way how software teams developed software and decided to build a new product that would substantially increase the likelihood of project success, predictability and software quality. Team System actually transformed the Visual Studio IDE into a powerful collaboration tool for the entire software development team. Visual Studio 2005 Team System was designed to achieve four primary goals:

- Be productive by reducing the complexity of delivering modern service-oriented solutions
- Be integrated with all tools, and facilitate better team collaboration
- Be capable by being robust, secure, and scalable, and by having the ability to work remotely
- Be extensible by allowing processes to be customized

The complete solution consisted of a series of different editions, targeted to the various stakeholders:

- Visual Studio 2005 Team Edition for Software Architects
- Visual Studio 2005 Team Edition for Tester
- Visual Studio 2005 Team Foundation Server

A common misconception was that TFS replaced Visual SourceSafe as just a new version control system. This is of course incorrect because version control is only a small part of the TFS feature set. Team Foundation Version Control (TFVC) was written from scratch (no link with SourceSafe) and uses a SQL Server database as a means of scaling to large environments and infrastructures.
Team Foundation Server is the central and key component in the ALM solution which offered extensible team collaboration. Team System was clearly designed as a lifecycle platform that enabled third-parties, customers, and solution providers to extend the base functionality with new features and customize the tool for the unique aspects of certain businesses. The Microsoft Solution Framework (MSF) – a set of software development processes, principles, and proven practices – was merged into the product via two Team Project process templates: MSF for Agile Software Development and MSF for CMMI Process Improvement.

At this time, the target audience for Team System were (large) enterprise development teams with a focus to improve the joint outcome of software development on the Microsoft platform (.NET). No sign yet of cross-platform development tools or support for multiple (non-Microsoft) development languages. Visual Studio 2005 Team System can be seen as v1 of the ALM offering from Microsoft. Later in 2006 another client Team System edition was added to the offering: Visual Studio 2005 Team System for Database Professionals. The DB Pro or Data Dude edition worked in conjunction with version control to manage database changes next to application code changes.
Heterogeneous Software Development

What about cross-platform development in the early versions of Microsoft Visual Studio Team System? Microsoft did not have a solution for this. Teamprise – founded in 2006 as a division of SourceGear – developed a suite of client applications for accessing Microsoft Visual Studio Team Foundation Server from outside the Visual Studio client environment. Teamprise enabled developers to use the source control and work item tracking features of Team Foundation Server from other platforms (operating systems that support a Java Runtime Environment), including Linux, Solaris and Mac OS, and from within the Eclipse IDE through a plug-in.

This plug-in was also compatible with IBM’s Websphere Studio and Rational Application Developer IDE. There was also a stand-alone Teamprise client application (written in Java) which featured an Explorer-style user interface for developers not working within an IDE. The product was written by accessing the Team Foundation Server SOAP web services. A bit later in 2007 the company also started working on a Java SDK for use with Team Foundation Server. Big enterprises and third-party software vendors were interested in using libraries developed by Teamprise in order to link existing Java products with Team Foundation Server data. This addition was opening up a large set of core libraries to expose functionality to Java developers in the form of an API they can use to build their own TFS-enabled applications.
**TFS 2008**

Back in November 2007, Microsoft released a new version of the Team System product: Visual Studio Team System 2008. This release can be seen as the first update to the Team System brand. Visual Studio 2008 Team Foundation Server delivered improvements in several areas, such as enhanced features in Team Build and version control, better performance, scalability and more flexible configuration. But it was only in November 2009 that Microsoft acquired Teamprise to provide cross-platform support for Visual Studio. It was announced that functionality from the Teamprise Client suite would be integrated into the Visual Studio product line, beginning with the next version of Visual Studio (Visual Studio 2010). TFS in combination with the Teamprise Client Suite technologies would enable development teams to use a single tool to overcome core development challenges, regardless of the core platform in use. This decision by Microsoft was a commitment to provide ALM tooling support across the entire enterprise, a desire to avoid the artificial technological boundaries inside big companies. Heterogeneous development was at that time officially a new focus for the TFS/ALM product team inside Microsoft and the old TeamPrise (Java) developers joined the team. A few months later in April of 2012, Microsoft came up with a (temporary) name for the product: Microsoft Team Explorer 2010 codename “Eaglestone”, or TEE in short. The Program Manager for the new addition to the Visual Studio product suite was Martin Woodward, a senior software engineer from the former TeamPrise employees.

**TFS 2010**

At the global Visual Studio 2010 launch event in Las Vegas (April 2010) *Visual Studio Team Explorer Everywhere* (TEE) was announced. The Teamprise features were now completely integrated and part of the Microsoft solution set for development teams.

*Previously sold as a separate product, Visual Studio Team Explorer Everywhere 2010 works with your favorite Eclipse-based IDE, in the operating system of your choice, and helps you collaborate across your .NET and Java development teams using Team Foundation Server 2010. It’s an easy-to-install standalone plug-in that’s now a free download.*
Java developers of course also need to run builds and are adopting continuous building/integration practices. So, why not provide this via TFS Builds?

The Team Foundation Build Extensions (to be installed on the Team Foundation Build Agent) offered the ability to execute Ant or Maven 2 builds from Team Foundation Server and publish the results of the build along with any associated JUnit test results back to Team Foundation Server. This release was compatible with Team Foundation Server 2005, Team Foundation Server 2008 and Team Foundation Server 2010.

Team Explorer Everywhere and the Build Extensions got also updated in the newer versions of TFS to improve the Team Explorer experience in the non Visual Studio world.

- What’s new in Team Explorer Everywhere 2012
- Team Foundation Server 2013 Build Extensions
- Announcement Team Explorer Everywhere 2013
A fully integrated testing experience with TFS 2010

The third release of Microsoft’s ALM Solution (Visual Studio 2010) no longer used the Team System moniker and brought all the available SKUs back under the good old Visual Studio brand. The 2010 release was definitely the most significant release with many new functionality and improvements since the launch of the product with Team System 2005. Microsoft put a lot of effort into building a full testing experience into Team Foundation Server with a test management suite for managing test planning and for running manual/automated UI tests, including the ability to analyze test impact based on code changes. The Lab Management solution on top of TFS was quite revolutionary and can now be seen as the first indications of the growing importance of automated deployment and testing.

As a result, Gartner included Microsoft as Visionary for the first time in their Magic Quadrant for Integrated Software Quality Suites (published report of January 2011). Microsoft Test Manager (MTM) was the new kid on the block and could be purchased separately with the Test Professional Suite (with MSDN Subscription). The Ultimate edition of Visual Studio also contained Microsoft Test Manager. Many existing testing tools on the market focused particularly on helping specialists testers (strong programming/scripting skills) with white box testing and API testing, but the majority of testing activities in the industry was still done manually (black box testing) while not a lot of testing tools provided extra support for this manual work. Consider MTM as an integrated test environment (ITE) for testers, compared to an integrated development environment (IDE) for developers. The new SKU wanted to provide a bridge between the testing team and the development team. It’s the interface that a tester will use to create test cases, organize test plans, track
test results, and file bugs when defects are found. Test Manager is integrated with Team Foundation Server, and is designed to improve the productivity of testers.

An interesting feature was for example the ability to play back action recordings to fast forward to a particular test step in a manual test to verify if a bug was fixed. It was also possible to create automated UI tests in Visual Studio (known as Coded UI Tests) using new automation libraries. You could generate C#/VB.NET code in Visual Studio from an existing action recording and add validation code to verify that the application under test was performing correctly.

The most compelling story was to file actionable bugs as a tester to avoid the typical ping-pong game between developers and testers. Familiar with the famous “it works on my machine” reply from the developer who marked the bug as non-reproducible? During a manual/automated test run with the built-in Test Runner, a lot of Diagnostic Data Adapters (DDA) could be enabled (test settings) to automatically collect and attach all data to a new Bug report: video recording, system information, IntelliTrace, Test Impact Analysis, screenshots, ... It became plain easy for a tester to gather all important information and send it to the development team for further validation. Next to this, test managers could follow up which builds had bug fixes and/or new features added. They were also presented with recommendations about which tests needed to be rerun, based on the effect of code changes.
that were made to the codebase (test impact analysis on managed code). Instead of rerunning all test cases over and over again, testers could stay up-to-date with a new iteration by designing new test cases.

The icing on the cake for me was Visual Studio Lab Management 2010, an add-on for TFS 2010 which extended the Build Automation feature. It allowed builds to easily deploy their output to virtual environments which could exist of multiple virtual machines. Relying on the snapshotting features of the virtualization technology (Hyper-V), complete environments could be rolled back to a known clean snapshot/checkpoint to always enforce a clean deployment for each new manual/scheduled build. Automated Tests could be integrated in the Build-Deploy cycle of the Lab Build to validate each new deployment. The ultimate goal was to detect bugs as soon as possible in every incremental update of your application in Development and once a defect was found, all collected data through the collectors could be easily stored in an actionable bug report. No more no repro!

Team Foundation Server 2010 was designed to be easily customizable and extensible. TFS ensures that developers using any development platform can participate. This was also identified by an independent ALM study of Gartner, published in a report in June 2012. Microsoft has been identified as a leader in this area with the TFS 2010 offering. This is a clear indication that Microsoft has great tools for Application Lifecycle Management. Many larger companies also needed to think about compliance (Sarbanes-Oxley Act). This is often a very important and crucial requirement when choosing an enterprise ALM tool. Team Foundation Server 2010 provided three main pieces of functionality as it relates to SOx – Work Item
Tracking, Version Control and Automated Build. These three features working together provided the capability to eliminate many of the risks associated with SOx 404 compliance. At the heart of SOx is the ability to know who made what change when, what requirement was the change made for and who authorized the change to be made. Was the requirement tested to make sure that it works correctly and who authorized the release of that requirement to production? Or, to state it another way, is the requirement traceable? Many technical solutions require multiple tools to provide this kind of traceability and most of the time they break the traceability chain at some point because they are not integrated at all required points. With Team Foundation Server 2010, full traceability was baked in. Team Foundation Server provided a comprehensive and flexible set of capabilities through traceability, test case management and automated builds. The rich extensibility of both the built-in features and a third-party ecosystem produce tools your company need in order to meet the requirements of SOx. A full article on Sox and TFS 2010 is available at http://msdn.microsoft.com/en-us/library/gg983694.aspx.
About TFS 2012, the Cloud, Git and the new shipping cadence

The next major release of Team Foundation Server (TFS 2012) was released on September 12, 2012. Some of the key new features in this release were full integration of Agile Project Management through a new improved Web Access UI, involvement of feedback from other stakeholders in the software development process by using PowerPoint StoryBoarding and the Feedback Client, increasing the productivity of developers by introducing a new Team Explorer UI and by providing important version control enhancements like the availability of Local Workspaces.

Continuous Value Delivery

The main theme of this release was Continuous Value Delivery. TFS 2012 delivered workflows and tools that are able to shorten delivery cycles, include customers and operations (non-development stakeholders) in software construction, and it looks for opportunities to reduce waste in the software development process. As a result, your organization should be able to reduce risks, solve problems faster, and continuously deliver value that exceeds customers’ expectations.
Primary goals of ALM with Visual Studio 2012:

- Prioritize collaboration among everyone involved in developing an application, incorporating customers as equal members of the development team.
- Deliver timely and actionable feedback to reduce wasted effort and its associated costs.
- Provide natural and appropriate tools for the tasks involved in designing, developing, testing, delivering, and maintaining quality software.
- Support best practices for application development, while remaining independent of any specific methodology.

Reflecting on the impact this release had for many of my customers, it must have been the best release of Team Foundation Server. The majority of the new big features were easy to demonstrate and it didn’t take long to convince decision makers of the business value to
upgrade old TFS environments to TFS 2012 or to introduce it to new customers who were struggling with implementing ALM in general. There were so many known pain points in software development tracks which could benefit heavily from built-in solutions in the TFS 2012 release. Think about for instance the possibility to shorten the feedback loop between business and the development team, but it also included important development productivity features like explicit code reviews in the daily workflow of developers and the inclusion of a new improved diff/merge experience. After feeling a bit lost in the redesigned Team Explorer UI in the beginning, it became clear that it focused more on a task-oriented approach, freeing the development team from many of the distractions that can occur when working on a complex project and enabling the team to work more quickly and more efficient (for example: Suspend-And-Resume functionality to stay in the zone). Testing became a true first-class citizen in the development process with a unified Test Explorer in Visual Studio and a unique way to perform automated and exploratory testing from Microsoft Test Manager.

**DevOps**

Team Foundation Server 2012 provided also a server monitoring solution for teams that use or want to adopt System Center Operations Manager (SCOM). A monitoring agent can be installed/configured for ASP.NET applications running on a web server. This agent collects rich data about exceptions, performance issues, and other errors. Using the TFS Connector for SCOM, Operations staff can file these exceptions as operational issues (work items) in TFS and assign them to the development team to investigate in order to improve and fix production web applications. Visual Studio and the TFS connector, in conjunction with SCOM, provide a real-time improvement feedback loop for server-based applications deployed in production environments, leading to continuous improvements, an increased mean time to repair (MTTR) cycle and better quality in the end of the application in production.
The Cloud Story

More importantly, Microsoft has also been working very hard on an ALM “cloud” offering in this timeframe: Team Foundation Service or Team Foundation Server in the cloud. A preview of Team Foundation Service was announced for the first time at the BUILD Conference in 2011 and the public launch was communicated on June 11 of 2012. What started as a technical experiment to get TFS running on Windows Azure eventually evolved into a new product which also completely shifted the release cadence for new Team Foundation Server releases. More info on this in one of the next paragraphs.

The cloud version of Team Foundation Server provides (small) development teams easy access to ALM features (Version Control, Agile Collaboration and Automated Builds) without having to install and manage the server application on-premises. Not all existing on-premises TFS features like Reporting and Lab Management are available in Team Foundation Service, but it’s clear that this offering is quite compelling for small companies that want to be instantly up-and-running with the foundations of Application Lifecycle Management.

The Git Story

Also in 2012, Microsoft announced (basic) Git integration with Team Foundation Server via the open source Git-TF solution (http://gittf.codeplex.com/) which offered a set of cross-platform command line tools (running on Windows, Mac and Linux) for sharing version control changes between TFS and Git. Instead of building their own Distributed Version Control System (DVCS), Microsoft decided to embrace Git (the most popular DVCS solution on the market) even more when it was publicly announced in January 2013 that Team Foundation Server would host genuine Git repositories and Visual Studio would get native Git support for managing local and hosted repositories. I remember Brian Harry introducing that
idea in a private NDA event at the MVP Summit in February 2012 (almost a year before the public announcement). There were mixed opinions internally in the TFS product team, but for the Visual Studio ALM MVP group it sounded immediately as the way to go: adopt the best DVCS solution in the market! I’m still very glad Microsoft eventually took this decision because it’s nowadays actually adding extra value for DVCS workflows in Visual Studio on top of Git instead of playing catch up with Git by building their own “Microsoft” DVCS solution. Git didn’t have the best client development experience on the Windows platform in the past, but with the Visual Studio Tools for Git, developers are getting native support for managing Git repositories inside Visual Studio and they could also target hosted Git repositories in Team Foundation Service. Including Git in the broad ALM vision of Microsoft resulted today in an integrated enterprise-ready ALM offering with Git: version control, work item tracking and build automation. It’s good to see that Microsoft is now actively working together in the open-source space to provide the best possible experience for software developers and it strengthens Microsoft’s presence on non-Windows platforms.

**TFS Shipping Cadence**

As from the beginning, Team Foundation Server has been a boxed product which had a 2 to 3 year release cycle. Bringing TFS to the cloud started a new unknown era. Rolling out fixes/updates for an on-premises product or for a cloud service is totally different. Imagine a cloud service which doesn’t update for a long period of time. People would assume it’s dead and forgotten. Customers of TFS in the cloud expect the product to evolve much faster than a boxed product and a service in the cloud must always be “on”. Delivering small and big updates to Team Foundation Server (different on-premises versions + cloud) was a big challenge for the TFS Product Team, but they managed to get it aligned pretty well now. Team Foundation Service will always be the front-runner and is updated every three weeks with features enabled/disabled by feature flags. Many features you will detect in the cloud may not be present for quite some time in the version of TFS you are running on-premises. On average, updates to VS/TFS (on-premises software) are pushed in a quarterly update and new major versions of TFS will be released yearly.
TFS 2013 and Visual Studio Online

The latest release of Visual Studio and Team Foundation Server (Visual Studio 2013) will empower development teams to live up to the demand of a new breed of applications, to develop and manage modern software that provides the best experience across multiple screens and devices, always-connected services. The wave of extra ALM features will help to become more productive and eases the collaboration between team members with improved support for agile development practices: Agile Portfolio Management, Team Rooms, Test Management from the browser, Work Item Charting, on-premises support for Git repositories, Code Health Indicators (CodeLens), Cloud Load Testing via Team Foundation Service, the connected IDE experience across all your Visual Studio instances …

An important trend which continued to be reflected in the 2012 release was the improved integration towards other Microsoft products (for example System Center and Windows Azure). In that respect we will probably also see some upcoming modifications in the integration with InRelease, a release management solution from InCycle. Microsoft reached an agreement to acquire InRelease on June 3, 2013.
The solution allowed software development teams to automatically deploy and track applications (from Team Foundation Server) as a whole to multiple environments – up to production – based on a business-approval workflow.

While the official release of Visual Studio 2013 was planned on November 13 with the Launch Event, the RTM bits were already made available on October 17, 2013. Before the official release, the public could experience already the more tight integration with Windows Azure. For example:

- The Cloud Loud Testing feature to provide virtual machines that generate the load of many users accessing your website at the same time.
- The Connected IDE: Sign in to Visual Studio!
- The interesting Azure benefits with MSDN Subscriptions to develop and test in the cloud for free using monthly credits.

**Visual Studio Online**

But, important new announcements were made at the official Launch Event. Team Foundation Service became Visual Studio Online in the perspective of broadening the complete offering to indicate that Visual Studio Online will become the services component of the developer tools for the future.
Visual Studio Online, formerly Team Foundation Service, is the home for your project data in the cloud. Get up and running in minutes on our cloud infrastructure without having to install or configure a single server. Set up an environment that includes everything from hosted Git repos and project tracking tools, to continuous integration and an IDE, all packaged up in a monthly per-user plan. Connect to your project in the cloud using your favorite development tool, such as Visual Studio, Eclipse or Xcode.

Visual Studio Online is now running in Commercial Preview mode (paid usage is enabled and price plans have been announced) and will shift to General Availability somewhere in 2014, after all billing services have been put into place. Until that time, everyone may still freely use the system as an early adopter. Cloud build and Cloud Load Testing as described above are already part of the released features of the service, while 2 new preview features have also been announced at the Visual Studio 2013 Launch event: Application Insights and Monaco.
**Application Insights, Build-Measure-Learn**

This Visual Studio Online feature now provides a 360 degree view of running applications by collecting, processing and displaying a wide variety of telemetry in three areas: performance, usage and availability.

Without any effort during the development lifecycle (seamless integration experience), development teams will benefit from the extensive set of collected diagnostics to analyze what’s going on in their deployed applications. With these valuable insights, future (to-be-developed) improvements can be more focused and impactful. Application Insights will be able to deliver the required metrics to adopt validated learning and to move into a real Build-Measure-Learn cycle.
**Monaco, Visual Studio in the browser**

In an attempt to bring the client development experience in Visual Studio to the cloud, Microsoft released a preview of **Monaco** which immediately enables the development of Windows Azure Websites directly from any modern browser on any device. It complements the Visual Studio desktop IDE as a low-friction entry point that will help any developer to get started or make quick changes to an existing cloud service. It has support for C#, VB.NET, HTML5, JavaScript applications and PHP.
Conclusion

I have been a “Team System” user/advocate from the early beta versions in 2004 and it’s amazing to experience at short range what has been delivered with all releases of Team Foundation Server to improve the quality and predictability of software development. While the first release wasn’t all that glamorous to get up-and-running, it has started a complete new way of collaborating as a team in software development projects.

Nevertheless IT software development projects still have a very bad reputation to deliver quality software on time and on budget. Often this is due to a lack of experience and a lack of software craftsmanship, but on many projects it’s also due to lack of a decent ALM vision and strategy. In a world where software is eating the world, all companies are becoming a software company and should invest in adopting Application Lifecycle Management. It’s not about writing some code that runs on your box, but it’s about bringing value to the market for the consumers of your product in a reliable and sustainable way. That’s why I like so much the cloud story for TFS and the increased delivery speed of new features. Have a closer look at the details in the Features Timeline page on Visual Studio Online.

The last few years have been extremely interesting for those people living in the ALM space. The evolution of ALM tools in the market has been phenomenal. From very specific niche tools of small companies to the more all-in-one solutions of the traditional big players like HP, IBM and Microsoft. The specific niche tools are more widespread in the startup landscape and might often be the best fit for a specific purpose. The big vendors on the other hand will always have their place in large organizations, but the challenge will remain to integrate all active ALM related tools to improve the collaboration and efficiency between different stakeholders. Many people are still keen to have quick access to the data they need within their tool of choice. Data will need to synchronize flawlessly across different ALM tools/solutions without losing focus on the value of a corporate ALM strategy. Companies like TaskTop and OpsHub are already playing an important role in this area.
Another important evolution of Microsoft’s ALM solution has been to also support non-
Windows environments. Team Explorer Everywhere (TEE) was the result of specific attention
for cross-platform collaboration from a dedicated group inside Microsoft. Nowadays, all
different TFS teams inside Microsoft share a common vision to deliver ALM features for hybrid
environments with extensibility in mind.

With the recent announcements of Application Insights and Monaco, Microsoft demonstrates
to drive the industry further in an improved and fast Build-Measure-Learn cycle (Lean ALM)
with an increased focus to embrace the opportunities of cloud computing.

With the recent announcements of Application Insights and Monaco, Microsoft
demonstrates to drive the industry further in an improved and fast Build-Measure-Learn
cycle (Lean ALM) with an increased focus to embrace the opportunities of cloud computing.

I would like to finish off with a link to the recent publication of the new Gartner report
(November 19, 2013) on Application Development Lifecycle Management where Microsoft
has been identified again as leader in the Magic Quadrant.

Microsoft is a Leader in the ADLM market with a strong customer base and
partner base, together with a solid stream of innovation. Microsoft offers
one of the broadest sets of ADLM functionality available in the market;
second only to IBM. Since delivering the Azure-based version of Microsoft
TFS, the vendor has moved to a consistent release train, moving new
features first to the cloud-based versions and then into on-premises
releases.
About Me

I’m the founder of Sparkles (www.sparkles.be) and as a Visual Studio ALM MVP I focus on Application Lifecycle Management solutions for the Microsoft .NET Development Platform based on Team Foundation Server. During the last 10 years, I have been involved in many large enterprise development projects as a developer, team lead and software architect. I’m very passionate about setting up the optimal team development environment to deliver high quality software and I strive to get the best out of people, tools and processes. You can read about my experiences at intovsts.net. Since a few years now, I’m also the User Group Lead for the Belgian Visual Studio User Group (www.visug.be) which organizes many free technical community events with local and international speakers.

Subscribe for my specialized 3-day hands-on ALM/TFS training courses at http://www.sparkles.be/training/almwithvsnext.aspx

Or contact me for a custom on-site workshop!

Day 1
- Introduction to the new Visual Studio Family
- Application Lifecycle Management
- The Benefits of Visual Studio and Team Foundation Server
- Gathering User Requirements through Storyboarding
- Agile Project/Portfolio Management
- Process Template Customization
- Continuous Feedback
- Visual Studio Online (previously known as Team Foundation Service)

Day 2
- TFVC vs Git
- Version Control improvements
- Built-In Code Review activities
- The new Team Explorer experience in TFS
- Increased Developer Productivity
- Cross-technology Application Lifecycle Management
- Build & Symbol Management in TFS
- Versioning & Release Management ["InRelease"]
- Digging into the TFS API

Day 3
- Continuous Unit Testing
- Exploratory Testing with MTM
- Test Impact Analysis
- IntelliTrace for production environments
- Microsoft Monitoring Agent & IntelliTrace
- DevOps collaboration (integration with SCOM)
- Lab Management improvements
- Application Insights
- Reporting/Analytics: transform data to information!