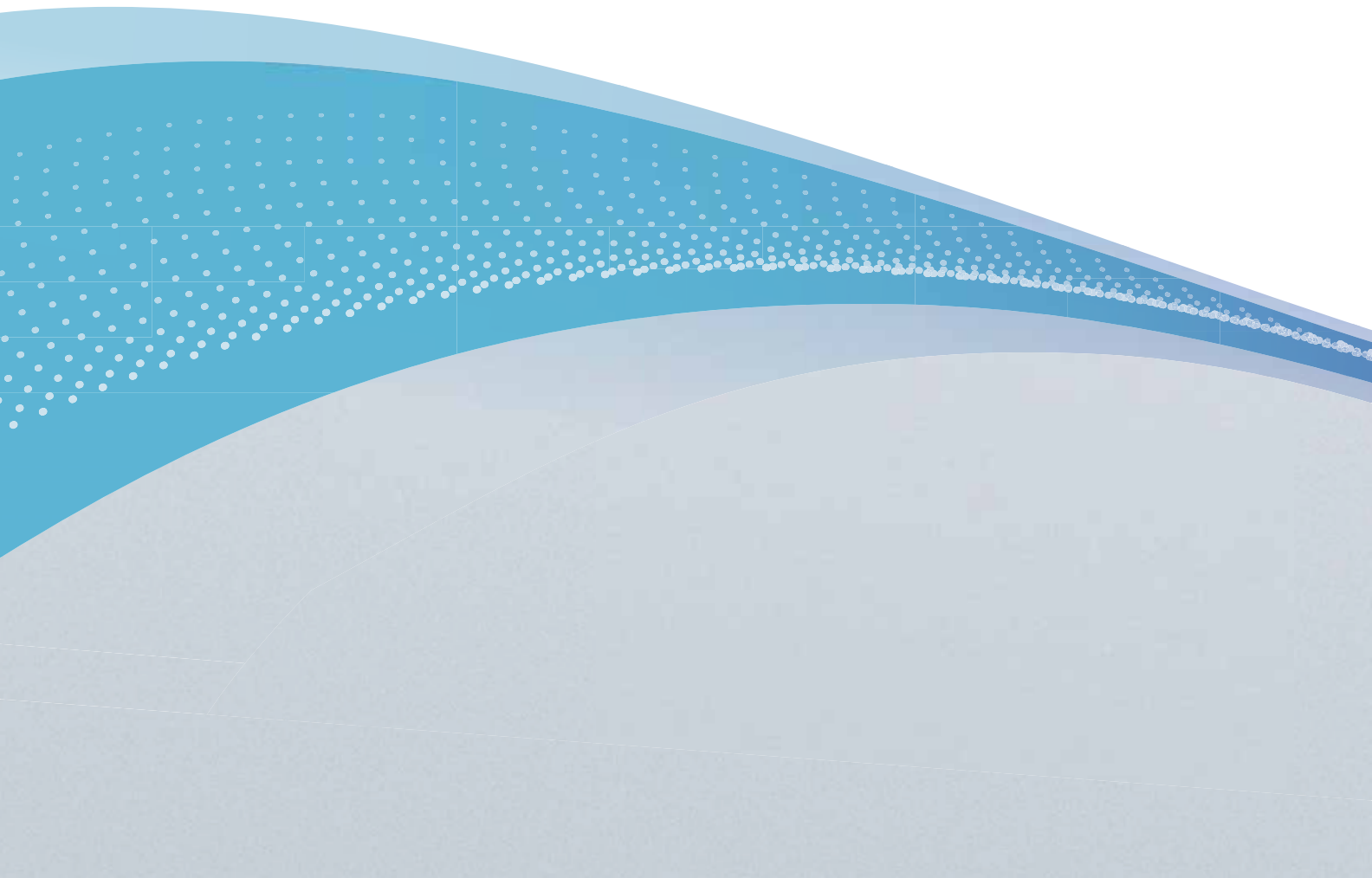
Visual Studio ALM

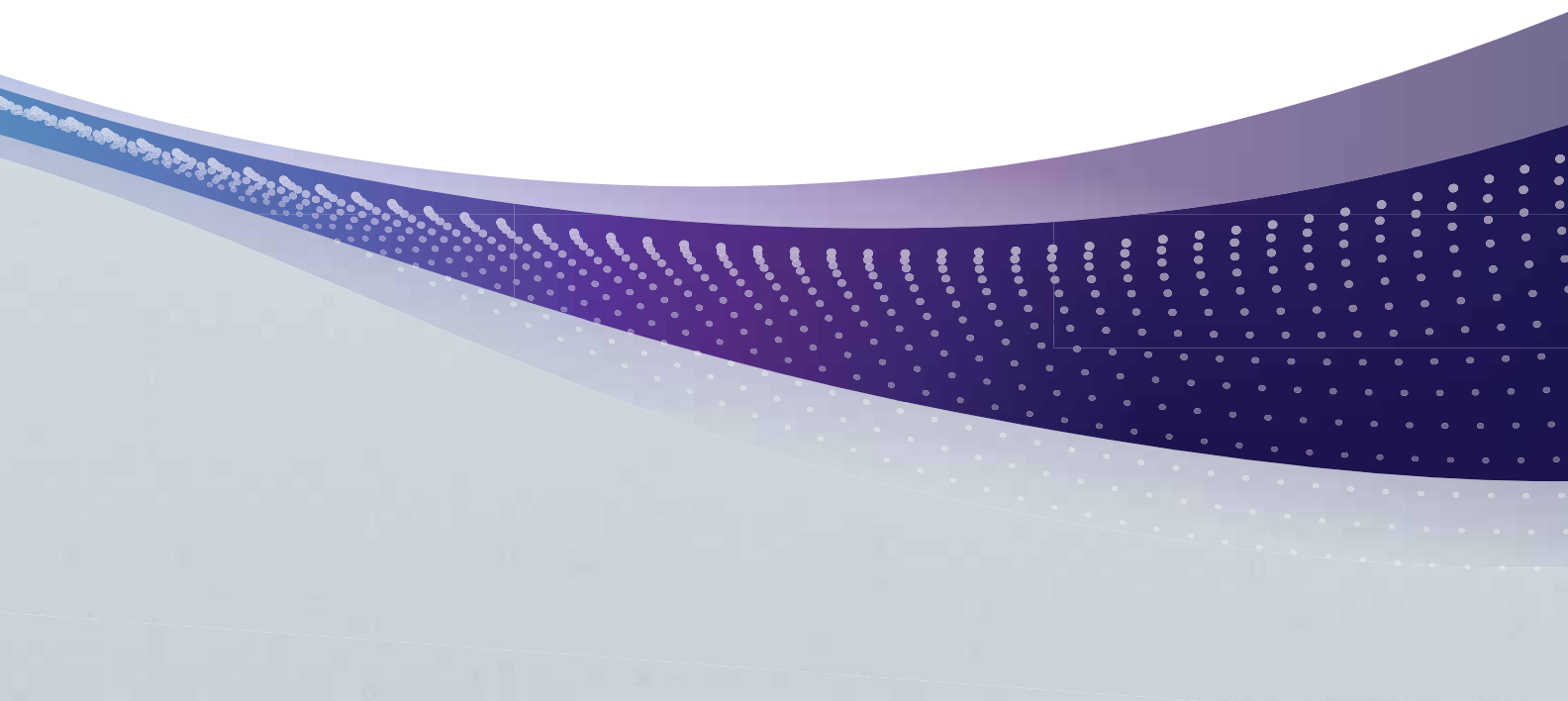
Solving Your Team Development Challenges

Microsoft® Visual Studio® 2010

A Series of Scenarios and Discussion



Welcome to the Microsoft Visual Studio ALM guide to solving your Team Development challenges. Here you will find content showing how Visual Studio 2010 can help you overcome many typical challenges that face software development teams today. You will gain insight from industry experts on the many benefits to be gained from implementing application lifecycle management in your organization and how the ALM tools from Visual Studio 2010, when coupled with good process and good people, can help you achieve success in all your software development endeavors.



Contents

001 Building Quality Applications and other stories

Tolstoy said in Anna Karina, "All happy families resemble one another, each unhappy family is unhappy in its own way". Is this also true for development teams? It often appears from the outside that each team has its own unique set of problems which hinder success, but in truth, there are far smaller subsets of common scenarios, that often manifest in different ways, that are at the heart of most of the challenges experienced by software development teams. This section features 7 common scenarios, that may be familiar to you and your team, and outlines how Visual Studio 2010 can be part of the solution so that you can resemble a happy family as well.

066 Understanding Application Lifecycle Management

Read what David Chappell has to say about why ALM is important to your business. In this section you will find 4 short papers that cover why ALM is at the heart of your success and how todays tools are now ready to take on all of your development challenges.

090 Introductioning Visual Studio 2010 for team development

For most people, the days of the lone developer are long gone. The great majority of software today is created by teams. Given this reality, modern software development tools are used primarily by people working together. Let David Chappell walk you through what Visual Studio 2010 means to team development. If you don't have time to read the whole article, look for the tags at the start of each section which will let you access videos of David presenting each topic. Then let Microsoft's own Sam Guckenheimer tell you about Software Engineering With Microsoft Visual Studio by reading an excerpt from his forthcoming book of the same title.

162 What do the analysts think?

The analyst's reviews are out and they are very positive. Gain some insight into how the Analysts think about the ALM marketplace and where Visual Studio 2010 is placed by read reports from Forrester and voke.

212 Hear what the press have to say

The word is out and the press is all over Visual Studio 2010. See what some of the leading technical publications are saying about Visual Studio 2010 and all of the great new features. Read articles from Martin Heller and Paul Krill from Infoworld and Michael Desmond from Redmond Developer News.

218 Additional Resources

Take a look at some of the additional benefits that Visual Studio 2010 has to offer for delivering innovative, high quality applications, get more details about what is included with Visual Studio Team Foundation Server 2010 and Visual Studio 2010 Ultimate and see what some of our partners have to offer.

Building Quality Applications and other stories

Tolstoy said in Anna Karina, “All happy families resemble one another, each unhappy family is unhappy in its own way”. Is this also true for development teams? It often appears from the outside that each team has its own unique set of problems which hinder success, but in truth, there are far smaller subsets of common scenarios, that often manifest in different ways, that are at the heart of most of the challenges experienced by software development teams. This section features 7 common scenarios, that may be familiar to you and your team, and outlines how Visual Studio 2010 can be part of the solution so that you can resemble a happy family as well.


Get the free mobile app for your phone
http://gettag.mobi

Building Quality Applications with Visual Studio 2010

Quality considerations are of paramount importance in the software business. It has long been understood that the costs of defects rise significantly when they are not caught early. The agile community, for example, refers to “failing fast” as an important rallying cry and, as a result, sees continuous integration and testing as fundamental practices. Visual Studio 2010 provides an aggressive set of innovations around test and quality to help software teams deliver superior results.

Poor quality software costs the United States economy some $59 billion annually according to the National Institute of Standards and Technologies (2002)[1] . This money is lost through poor productivity and wasted resources. The study concludes that one-third of this cost could be saved by improving the testing infrastructure. In the last several years, according to the Standish Group's Chaos Report (2009)[2] shown in Figure 1, projects which are challenged or failed edged up again last year. There is no single silver bullet, but organizations can help themselves by using tools and processes that give themselves the best chance of avoiding project failures. At the end of the day, the only measure of quality that matters is whether or not the customer is happy with the final product. In concise terms this generally means that the product does what the customer wants (provides greater efficiency and effectiveness in their daily work) and it is free of defects which detract from the effectiveness of the software. Visual Studio 2010 is designed to give development teams the best chance possible of producing high quality software

Figure 1- Chaos Reports show no significant improvement in project success.

MEETING CUSTOMER NEEDS

Gathering complete customer requirements is difficult, no matter what tool you use, because much of this problem is rooted in often unpredictable customer interactions. But stability can be established and the needs of the customer can be monitored and quantitatively reported on as long as you have robust traceability from requirements to test cases to code to bugs and test results. A complete traceability story allows teams to determine which requirements are the highest priority, how much progress has been made on the requirements (work completed and remaining) and the quality of those requirements. Finishing a requirement is unimportant if the requirement has numerous defects associated with it because then it does not meet the needs of the customer using the general definition of quality. TFS provides just that degree of traceability and the data is easily discoverable through the fully customizable out-of-the-box reporting.

1 National Institute of Standards and Technology. (2002). Planning Report 02-3, The Economic Impacts of Inadequate Infrastructure for Software Testing. U.S. Department of Commerce.

2 Standish Group, (2009). Chaos Report.

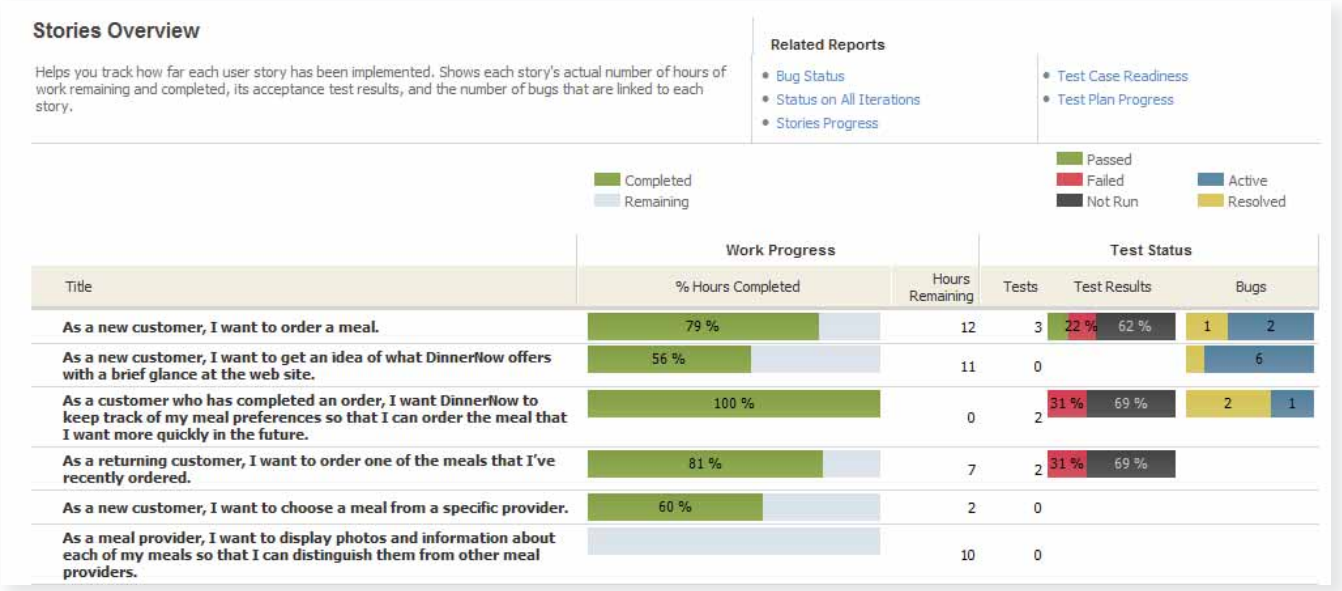


Figure 2 - Stories Overview Report.

At a glance the report shown in Figure 2 is fairly simple but it pulls together a multitude of data such as hours against requirements, test cases against requirements, test results for each requirement, and the number of active and resolved bugs for each requirement. At this point, knowing how well you have met the customer needs is no longer a guessing game – you have a definitive answer.

3 STEPS TO ACHIEVING QUALITY

There are really many steps involved in building quality applications but you can simplify the process down to just three key steps: Design, Develop, and Test.

Quality Starts with Design…

Today, many architecture tools are separate from the development, requirements and testing tools. This makes it extremely difficult to architect systems so that the architecture is not discarded or ignored once coding begins. Developers have to constantly refer to other documents or other tools and eventually they just don't. The side effect of this is that the design becomes out of sync with the actual code and traceability is lost at the technical level. This has an impact on the maintenance of existing systems as well. If you can't understand the system, then maintaining it is difficult. Visual Studio 2010 Ultimate incorporates a new set of architecture tools built around UML which allow teams to not only architect solutions in their development environment, but link those designs to requirements in a way that they can be easily referred to and kept up-to-date. Figure 3 shows examples of the Use Case and Activity UML Diagrams, alongside another new architectural tool directly related to quality, the Layer Diagram.

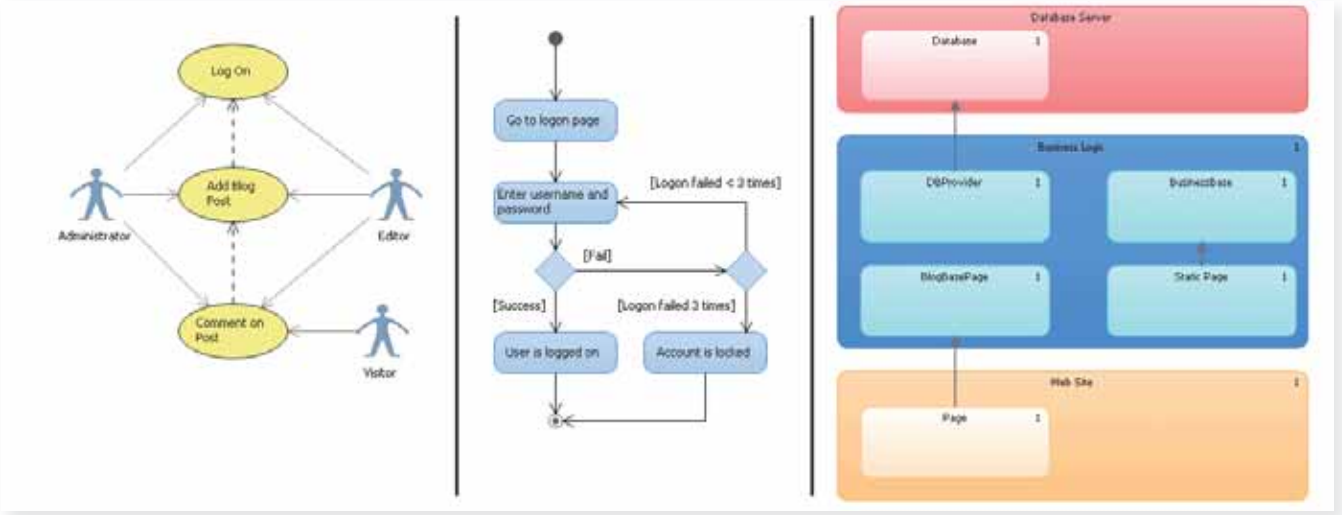


Figure 3 - A Use Case, Activity and Layer Diagram.

The Layer Diagram not only defines the layer design (a layer can be anything – method, class, namespace, assembly or an arbitrary grouping) and communication pathways but it can be used to constrain them. Once a design is created to achieve a particular goal, when that design is not followed the appropriate steps can be taken to either change the design or the code.

continues with Developers...

Visual Studio 2010 includes many of the tools developers need to ensure they are meeting the needs of the customer. Features such as static code analysis (for .NET code and database code) provide insight into where there may be potential performance, stability, maintainability, scalability and security issues before they become a problem. Code metrics allow developers to determine where their code is most complex (a.k.a. less maintainable) and change the code before it becomes spaghetti. Developers can also perform comprehensive performance tests on their code, whether it is a WPF application or an ASP.NET application, to ensure the application meets the scalability needs of the customer. Figure 4 shows a Load test during the test run. Detailed information is provided on any aspect of any system involved in the testing. This ensures that enough information is gathered to determine where performance problems exist. You can also compare Load test runs to determine the amount of change in the performance characteristics over time.

After a Load test, if performance problems have been identified, detailed reports can be examined to determine exactly which parts of the application contain the problem code so that it can be fixed. Figure 5 shows an analysis of the user load against the application during the performance run. The figure shows not only the user load but the pages they were executing and any exceptions or other failures that occurred.

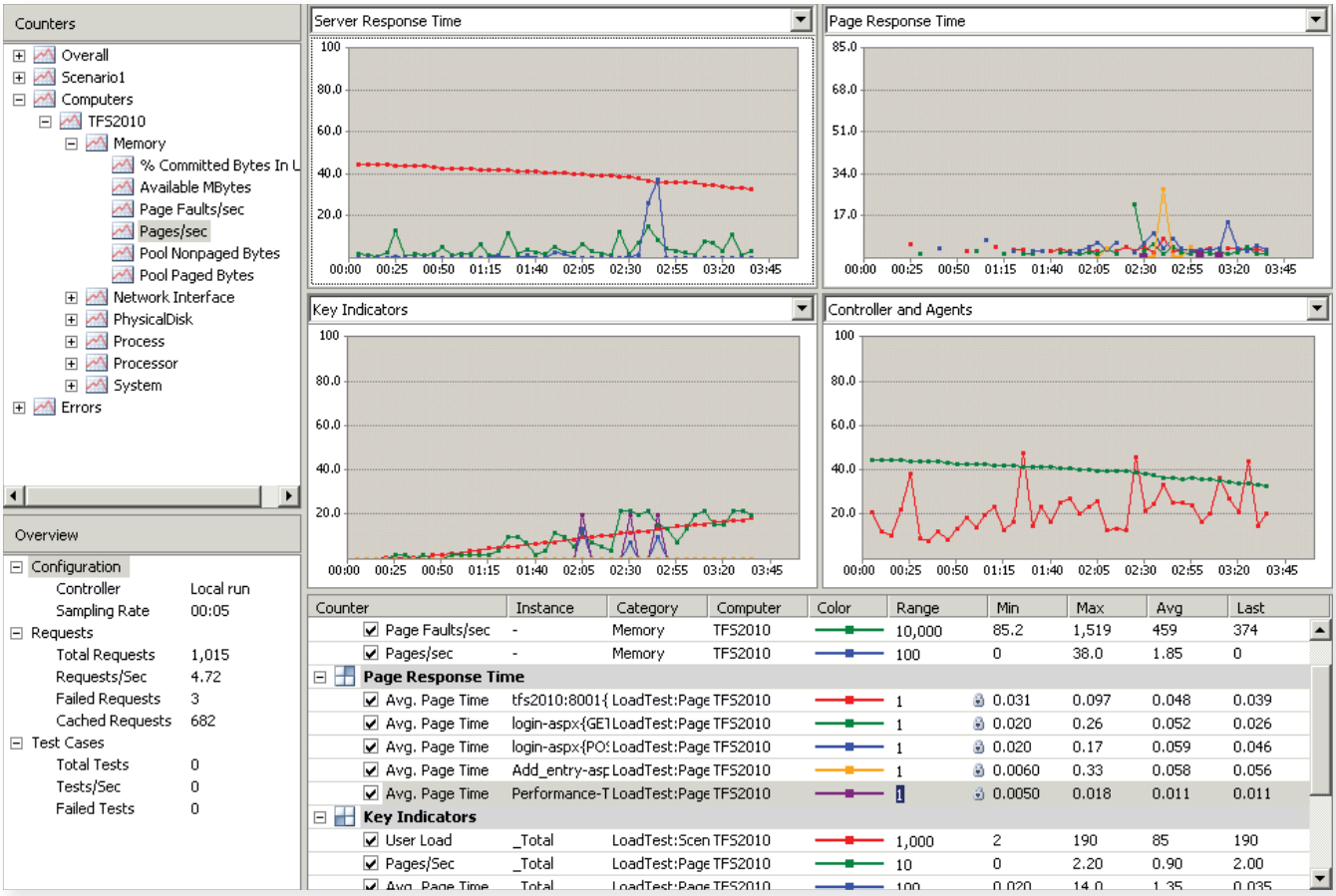


Figure 4 - A running load test.

Ensuring that defects do not make it to the final product requires that developers can find and fix bugs quickly during a debugging session. Microsoft's latest innovation is IntelliTrace™ which allows for this and so much more. IntelliTrace™ allows developers to run an application, pause it at any point, and then move forward and backward in the debugging session to examine application state at a given point in time. It records a navigable history of the stack trace so that developers do not have to set breakpoints and hope they hit the right one! Not only that but when an exception occurs, developers can move immediately to the point where the exception occurred, determine why it occurred, and fix it. On top of this, testers running tests on the compiled application can record IntelliTrace™ logs so that when an error does occur the developer has access to everything that happened during the test run. This allows developers to find and fix bugs more efficiently than ever before.

Developers also have a responsibility to execute some tests before handing the code off to testers or even checking it into version control. Unit tests allow developers to discover if the code does not behave the way that it should at a granular level. For those teams that use unit testing heavily (such as Test Driven Development teams), the new

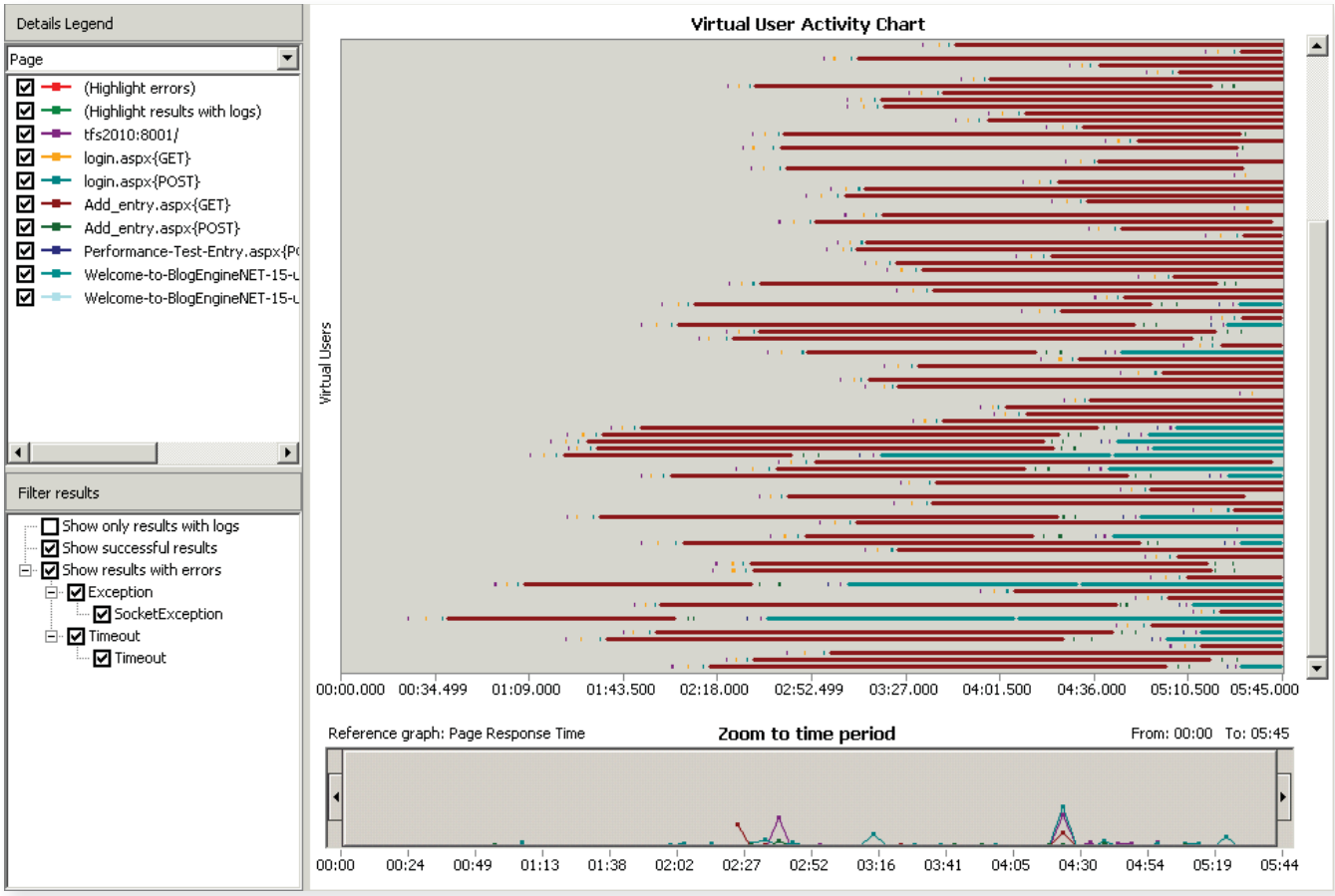


Figure 5 - Load Test Reporting.

Test Impact Analysis feature can reduce the number of tests which have to be executed and still give a high degree of confidence that the necessary tests have been executed. Test Impact Analysis analyzes code changes and correlates them with unit test runs to determine when a code change may impact the results of a previously executed test. This allows developers to spot check their code before sending it to the testers or the QA team.

Visual Studio 2010 also gives developers the ability to automate user interface testing. Developers can use Coded User Interface Tests to automate the testing of UI elements in their applications. Visual Studio 2010 automatically generates test code, either from a test run performed by a tester or from a run recorded by the developer themselves. This CodedUI test can be executed manually or incorporated as part of your build process to automate UI regression testing. Visual Studio Lab Management provides a virtualized testing environment which developers can use to run their code on clean systems which more accurately match supported release environments. This can help developers achieve their testing and experimentation goals in safe, easily replicated environments.

Automated builds have been a staple of Team Foundation Server and in TFS 2010 new features have made it easier than ever to discover coding defects before they ever make their way into the main code base. TFS 2010 sits on top of Windows Workflow 4.0 to provide powerful workflow capabilities with uses limited only by your imagination. One of the most innovative features that it is used for is called Gated Check-In and is a further refinement of Continuous Integration. With Continuous Integration, when the build breaks it's already too late – the code stored in version control is already broken and the team has to take time out to fix it. Gated Check-In executes the build and any required tests before the code is checked in. This way, if the build breaks or the tests fail the code is never checked in. This allows the rest of the development team to keep working and the developer who wrote the code can fix it. Gated Check-In uses a hallmark TFS feature called the Shelf Set to accomplish this. When a Gated Check-In fails, the developer can ask other developers for help in fixing the code – all without the broken code ever being checked in.

is verified by Testers…

Testers have traditionally not been part of the development team. Testing was an activity that happened after development was finished. The realization that testers are an integral part of the development team is just now dawning on organizations because of the very numbers mentioned at the beginning of this paper. Visual Studio 2010 fully incorporates testers into the heart of any project by including Test Case Management and a professional test management and execution tool – Microsoft Test Manager (MTM), shown in Figure 6.

At the outset of a project, the QA team can associate test cases directly with requirements. This allows developers to see the test cases that will be executed. Later, testers can execute tests against the code to determine if it meets the customer needs. There are many types of testing that can help ensure the quality of your application. Manual or general testing makes up about 70 percent of all the testing carried out by organizations today but other test types are also available. These test types range from standard unit tests and performance tests to load tests and automated tests. As part of this, testers can collect detailed diagnostic information and videos of their testing session so that when a bug is filed it is actionable – that is, developers can actually fix the discovered defects. Figure 7 shows the test repro section of a bug filed through MTM. All of the steps are detailed including which steps passed, which failed, and any attachments or comments included with each step. The video time indexing allows developers to jump right to the spot in the video where the bug occurred. Also included is the IntelliTrace™ log file which provides detailed debugging information for the test session leading up to the bug.

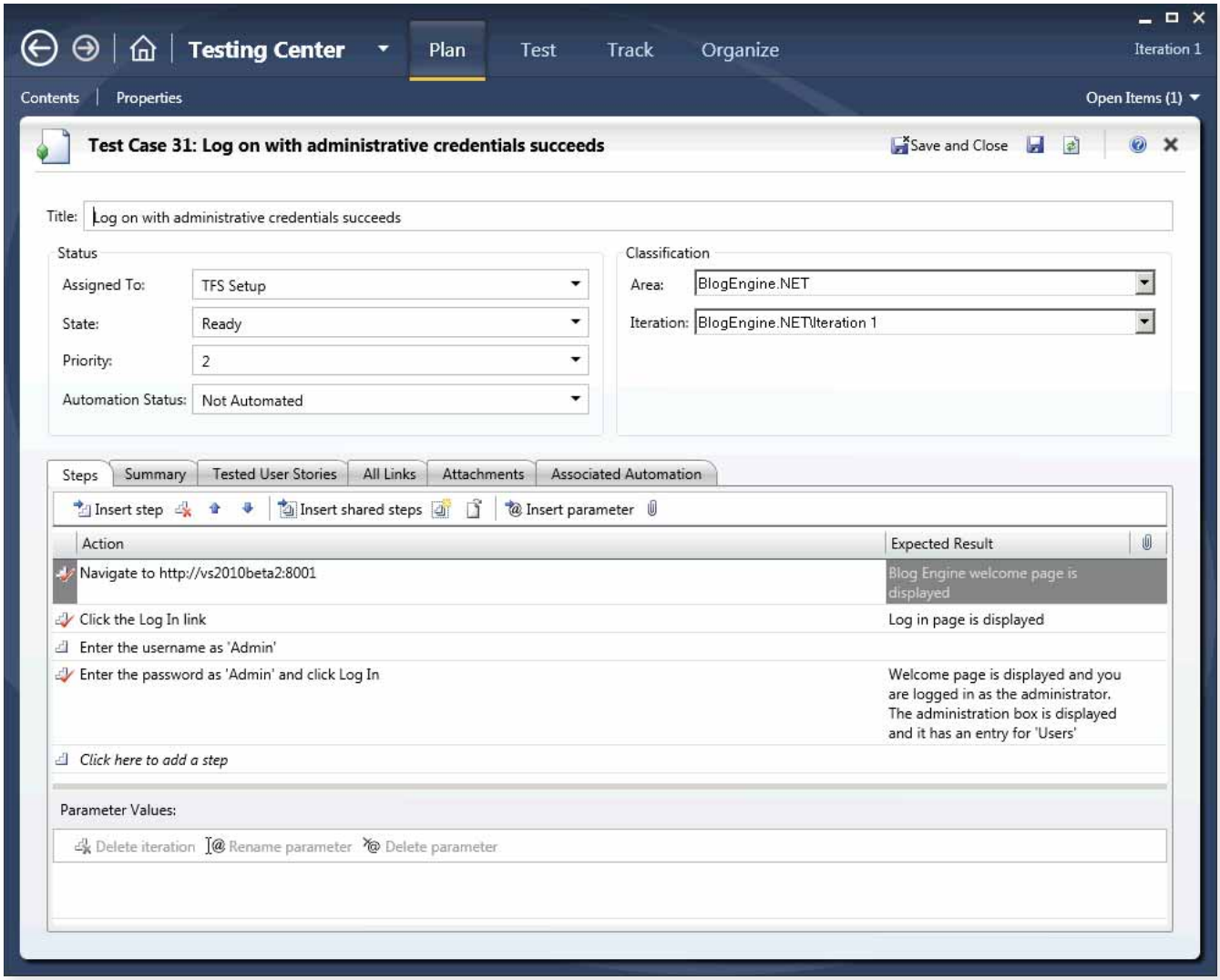


Figure 6- Microsoft Test Manager

Test Impact Analysis is a new feature that provides testers with the information they need to select which tests are high priority tests based on the code changes that a developer makes. When code changes that has already passed testing, the testers are alerted and can re-execute those tests to reduce or eliminate regression bugs. Lab Management refines this further by virtualizing the testing infrastructure so that when a defect occurs a snapshot can be taken of the virtual environment and provided to the developer so he can easily re-create the bug in the environment in which it was found.

3/16/2010 4:16:14 PM Bug filed on "Logged on users information is pre-filled in the comment fields"

Step no.	Result	Title	Video links
1	Passed	**Navigate to http://tfs2010:8001**	Video:00:01:09
		Expected result: Blog engine welcome page is displayed	
2	Passed	**Click the Log In link**	Video:00:01:12
		Expected result: Log in page is displayed	
3	Passed	**Enter 'Jeff' for the username**	Video:00:01:14
4	Passed	**Enter 'P@ssw0rd' for the password and click Log In**	Video:00:01:16
		Expected result: Welcome page is displayed and you are logged on	
5	Failed	**Click the first blog post**	Video:00:01:05
		Expected result: Post details page is displayed and the name = 'Jeff' and the e-mail address = 'Jeff.Jones@nowhere.com'	
		Attachments: Screenshot1(TC143Iteration1Step5).png	

Test configuration: Windows 7 and IE 8

Diagnostic Data Adapter	Log / Output
Actions	TC143_ActionLog.1.txt
Actions	TC143_ActionLog.1.html
IntelliTrace	w3wp_100316_161610918_7180.1.iTrace
System Information	SystemInformation.1.xml
Video Recorder	TC143Video.1.wmv

Figure 7- Bug Repro Steps.

… And is visible to Management

Visual Studio 2010 and TFS 2010 promote quality from the ground up – but management wants to know about the project status. Management has to report to customers on the progress and quality of the software and discovering at the last minute that there are problems can make it difficult to alter course. TFS includes detailed reporting based on SQL Server Analysis Services. When combined with Microsoft Office SharePoint Server and Excel Services, information – not just data – is always available, up-to-date and accurate. Figure 8 shows just one of the quality information dashboards available to you.

The Test Plan Progress shows how many tests there are, how many have been executed, and how many have passed. Build Status indicates the number of automated builds and their success over time – it combines build pass/fail with the results of automated testing that runs as part of the build. This provides a daily snapshot of the quality of your application. Bug Progress and Bug Reactivations provide detailed information on the status of all bugs filed against the application and how much re-work is being done because bugs are not being fixed correctly.

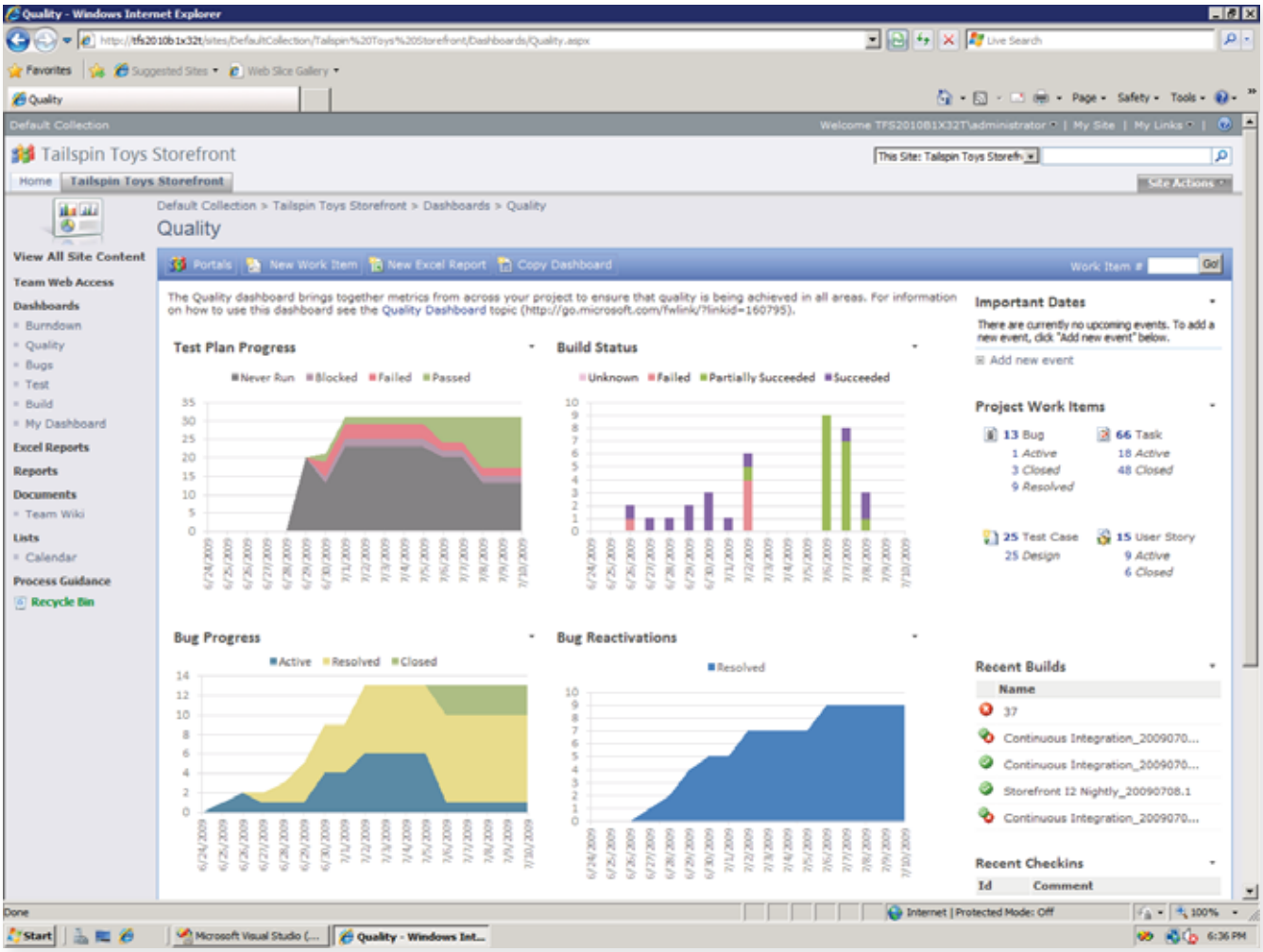


Figure 8- Quality Metrics

This information allows management to make course corrections early on. If a high number of bugs are being filed, actions can be taken to reduce the bug count. If tests are not being executed then more testers can be brought on. This provides the ability to better ensure a quality product is delivered on time and to the customers' satisfaction.

QUALITY MATTERS

Every development team approaches each project with one goal in mind – to produce a quality application. But that goal can be difficult to reach without the right supporting tools. From application architecture through development, test and maintenance, Visual Studio 2010 provides the right supporting tools to help ensure that applications are of the highest quality. No matter what role you play in the development process, these tools are built with you in mind to help you meet the needs of your customers.

SEE WHAT OTHER ARE SAYING: ICONICS

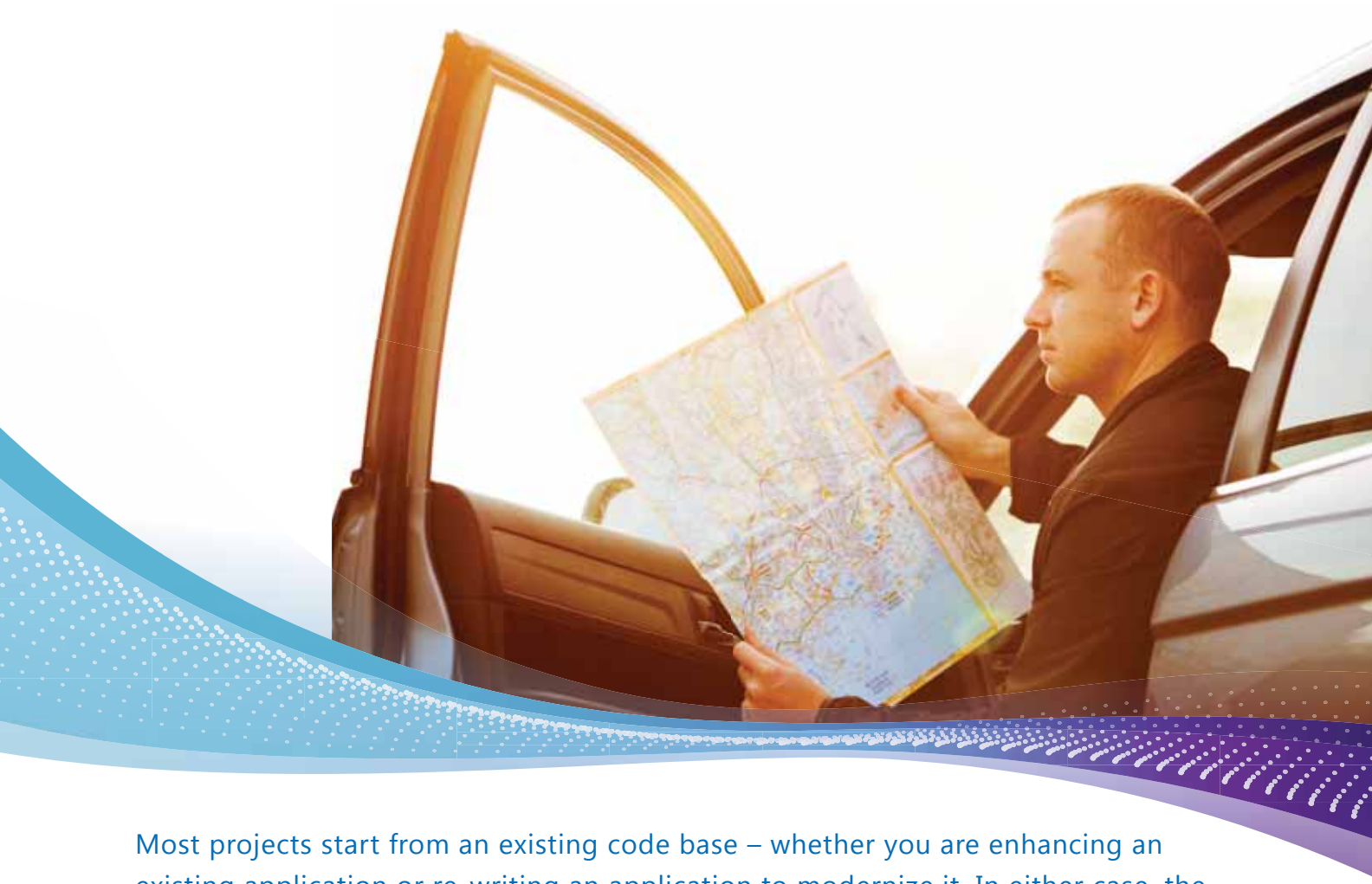
Global Software Developer Expects to Double Productivity with Development Solution

ICONICS, a Microsoft Gold Certified Partner, provides industrial automation and visualization software to organizations in more than 60 countries. The company's development teams include employees from locations in three countries—and the geographic distance created collaboration challenges. Developers lacked an integrated tool set that could facilitate project management, promote data access, accelerate development, and automate testing processes. In 2009, ICONICS enhanced its development environment by deploying Microsoft Visual Studio 2010 Ultimate, Visual Studio Team Foundation Server 2010, and Visual Studio Lab Management 2010. The fully integrated solution gives global teams unified tools to simplify project management and streamline product lifecycle processes. As a result, ICONICS can cut costs and improve product quality, and it expects to increase productivity by 100 percent.

"The new testing capabilities in Visual Studio 2010 are revolutionary. Once we get more teams using the tools, I can see our productivity doubling. We can run automated tests overnight without anyone being present. In the future, we also expect that features like IntelliTrace will significantly reduce the time required for us to debug applications." **- Chris Elsbree,** Chief Software Architect, ICONICS

"The advantages of using Visual Studio 2010 all come down to performance and quality. Features such as hierarchical work item tracking, branch visualization, and IntelliTrace—and our ability to build older versions with the new system—all contribute to an overall improvement in our efficiency. And this, ultimately results in a better quality product." **- Russ Agrusa,** President and CEO, ICONICS

Understanding Your Systems with Visual Studio 2010



Most projects start from an existing code base – whether you are enhancing an existing application or re-writing an application to modernize it. In either case, the documentation for the existing system is most likely non-existent, stale, or just wrong. The people who wrote the system are no longer available to the new team, which is left trying to understand a system with no starting point. Visual Studio 2010 gives you that starting point and allows teams to be productive in a fraction of the time it takes today.

Visual Studio 2010 Ultimate introduces new architecture tools that can help you improve the return on investment of developing systems and significantly decrease the cost of maintaining and updating existing systems. The architecture tools focus on allowing team members to visualize applications and understand the flow of information. Because of this, the cost and effort of adding new team members and understanding applications is greatly decreased. In using the architecture tools, teams will be able to focus more on new features and spend less time fixing existing issues. Now your existing applications can yield additional benefits at lower cost because they will last longer and are easier to maintain.

REDUCE THE COST OF MAINTAINING EXISTING SYSTEMS

Maintaining existing systems is expensive. More effort and money are put into maintaining existing systems than developing new systems. As systems get older, the cost of maintenance rises. Even the cost of maintaining a recently released system can be high if there are new developers working on it who have to understand the application. And that is what maintenance is all about – understanding the system and the potential impact of any change. Visual Studio 2010 reduces associated maintenance costs because it reduces the amount of time it takes to understand your system.

Traditional problems involved in any system are magnified when trying to perform maintenance on that system. Many of these problems involve documentation, or the lack of documentation. Code comments are non-existent, incomprehensible or inaccurate. System documentation is seldom updated, so the initial technical designs rarely match the finished product, and the documentation is never updated as part of the maintenance work. Sometimes, a far larger problem is "spaghetti" code – code that is just a mess and winds through many parts of the system. These problems can cause developers to spend days or weeks trying to reconcile the documentation with the application in a time-consuming exercise. Microsoft Visual Studio 2010 Ultimate provides two features to reduce the amount of time this takes: Architecture Explorer and Sequence Diagram.

Architecture Explorer

The problem with making changes in an existing system is that you often don't know what you are going to break until it's too late. This is a time-consuming and costly situation which can result in numerous bug fixes being required immediately after a release. To address this issue, Visual Studio 2010 Ultimate includes Architecture Explorer. Architecture Explorer provides a number of ways to quickly and easily determine what the application looks like from any level (method, class, namespace, assembly or solution). It also provides a dependency view to help you determine which items are connected to the one you are changing.

Architecture Explorer lets a developer either start small and gradually expand the view to get a full understanding of the application or drill deeply into those areas of the application that are of interest to the developer. This powerful visualization solution also allows developers to communicate risks and potential impacts in a graphical format. This process is shown in Figures 1-4.

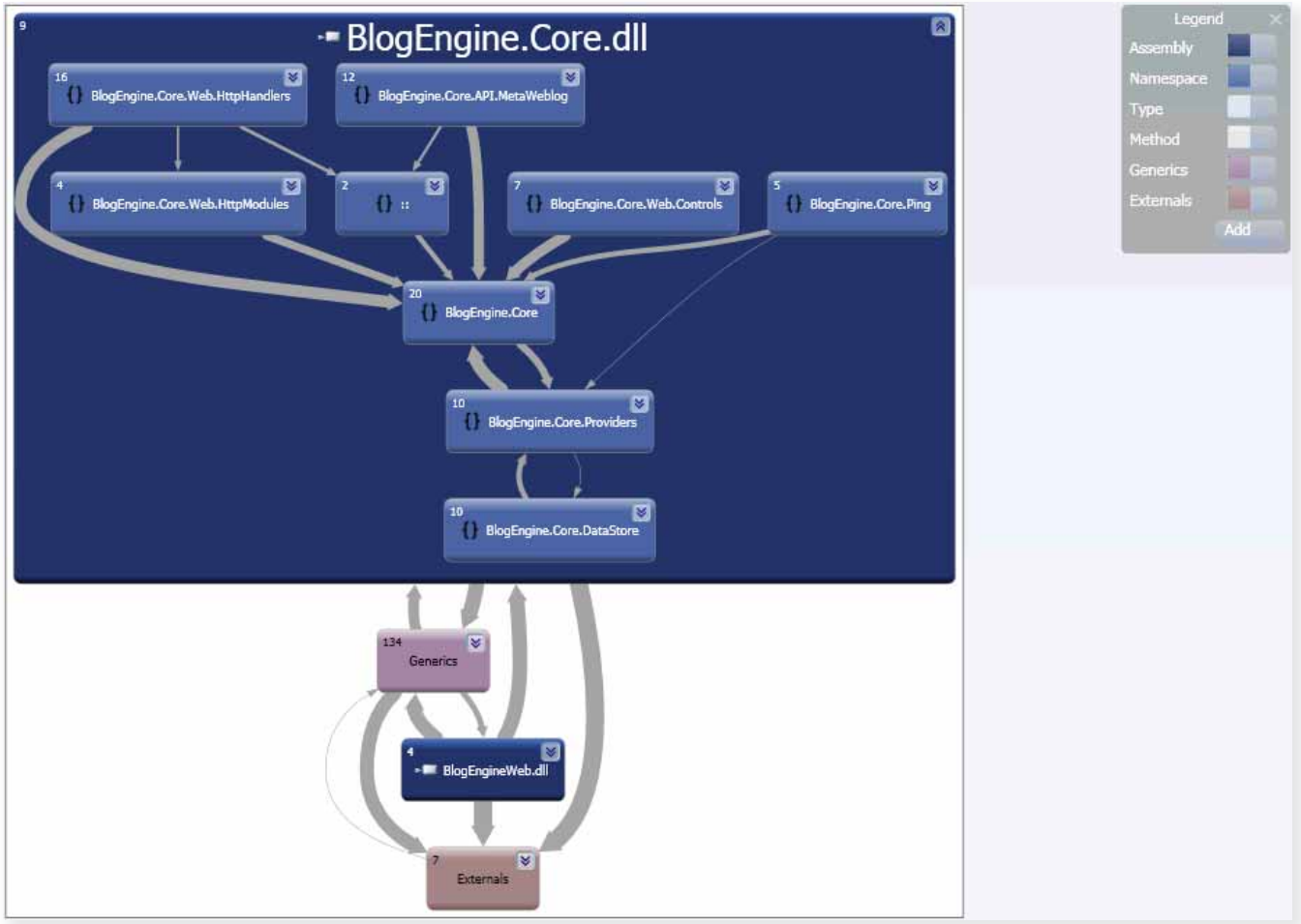


Figure 1 - High level view of the application.

Figure 1 shows a top-level view of the application as the developer begins to drill into the BlogEngine.Core.dll. In this case, the developer wants to focus on a change needed in the BlogEngine.Core namespace. Drilling further down, they see the view in Figure 2.

Now the developer can view the classes which make up the namespace and drill into the class or classes they want to change to analyze the potential impact. Drilling down even further, the developer sees the view shown in Figure 3.

At this point the developer can drill down further still or switch the view to the Dependency graph as shown in Figure 4. The dependency graph allows the developer to take any code they have visualized in Architecture Explorer and generate a grid of dependencies. Essentially it shows which

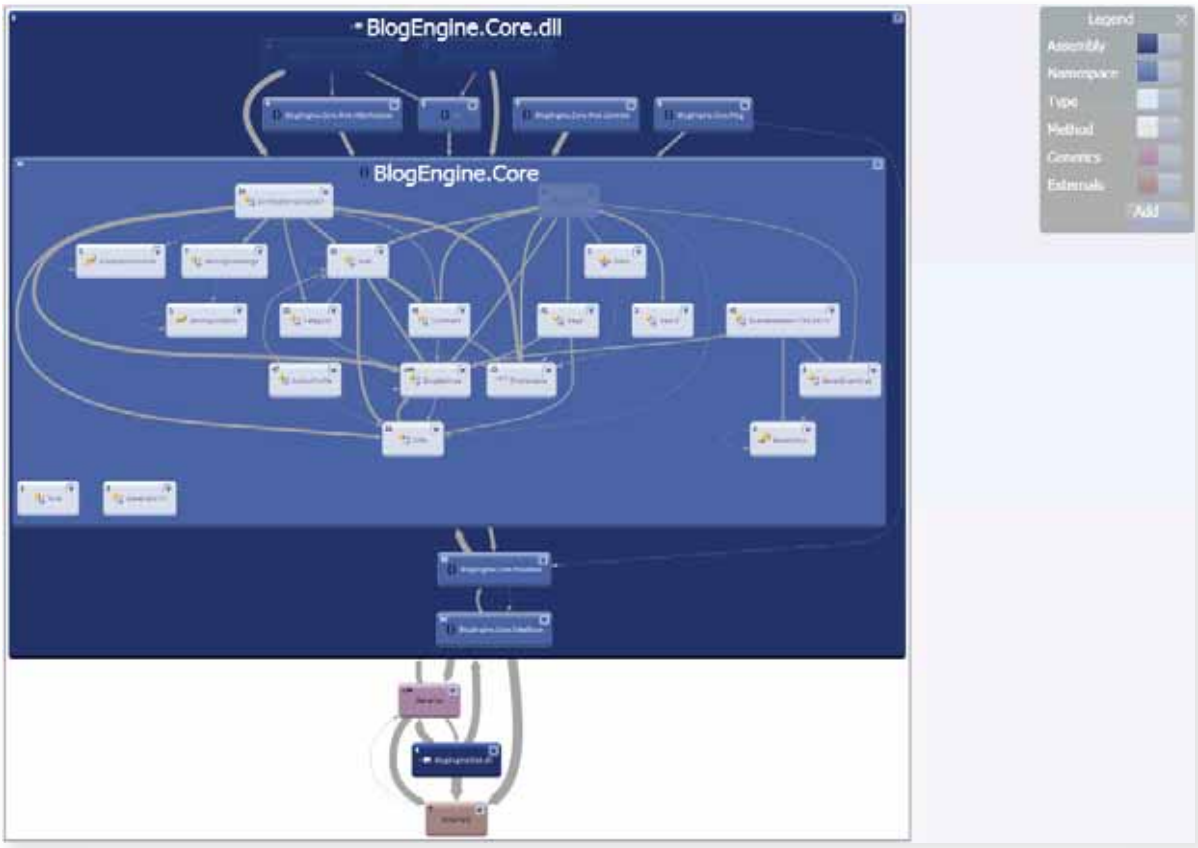


Figure 2 - Exploded view of the BlogEngine.Core namespace.

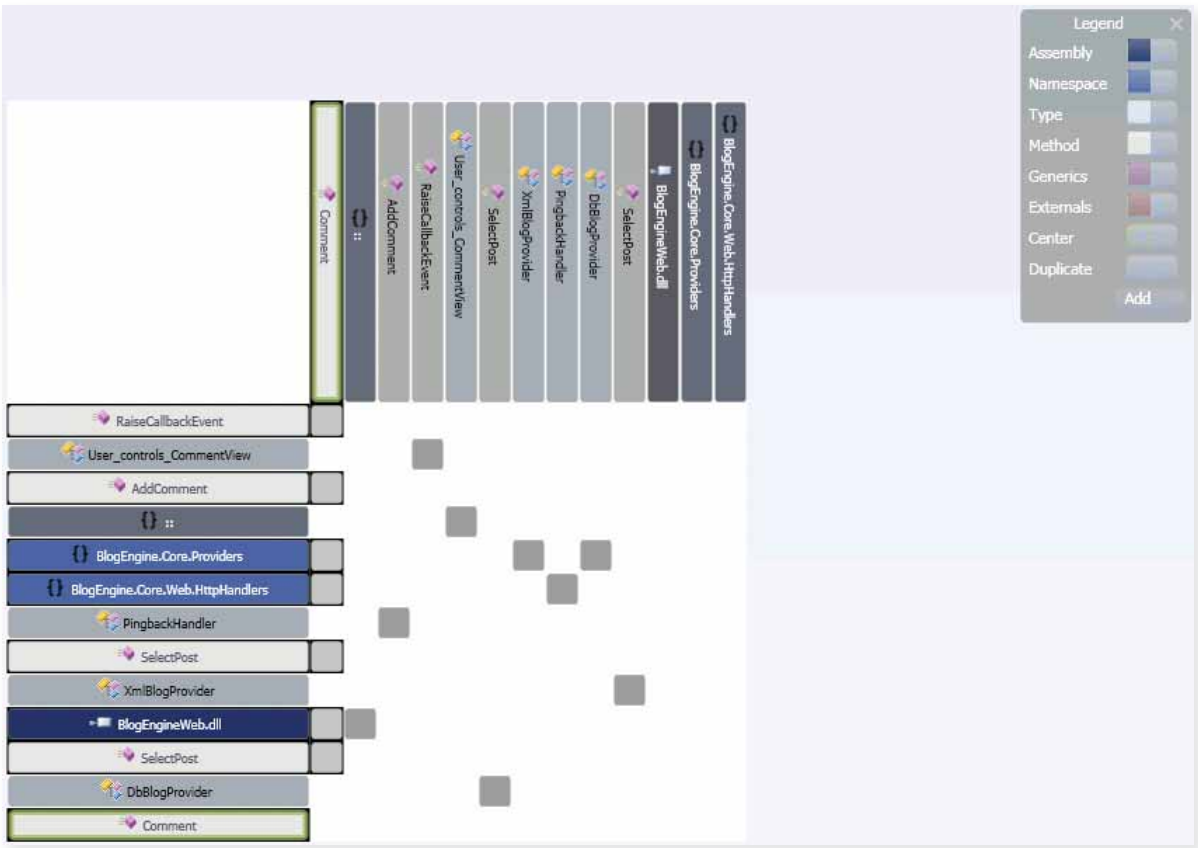


Figure 3 - The Comment method and related classes.

other classes (or methods, assemblies, namespaces, etc.) are in some way related to every other class. This allows developers to see the relationships and impact of potential changes before they make any modifications. This has not been possible before and can help reduce the risk of any given change.

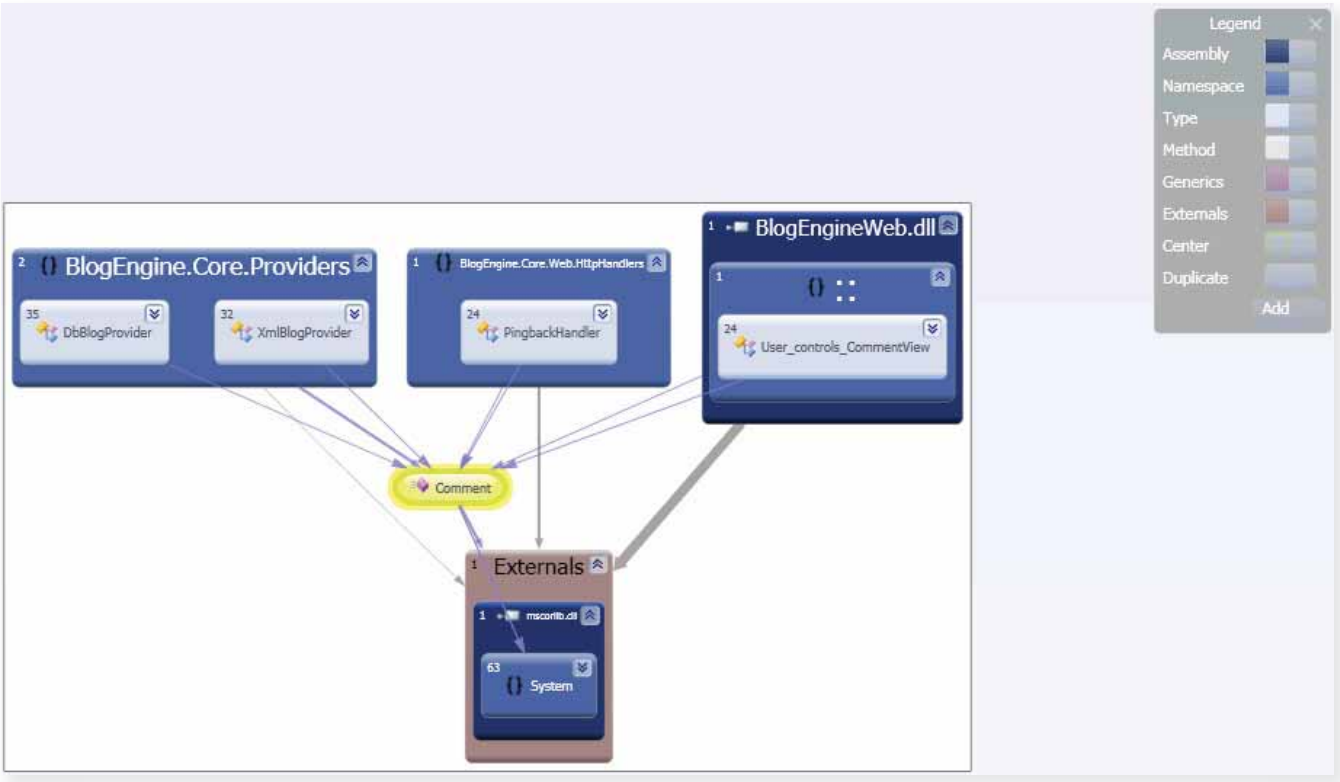


Figure 4 - Methods which call the Comment method in the Dependency View.

Sequence Diagrams

Once a developer determines where a change needs to be made, he or she needs to understand the sequence of calls to the method so they can make related changes. Prior to Visual Studio 2010, the code had to be executed and breakpoints set before a developer could step through the code. Not only did this take time but, in some cases, it was difficult to do. The Sequence Diagram allows developers to right-click a method to generate a full-sequence diagram showing all of the different classes that the method interacts with. Figure 5 shows an example of this.

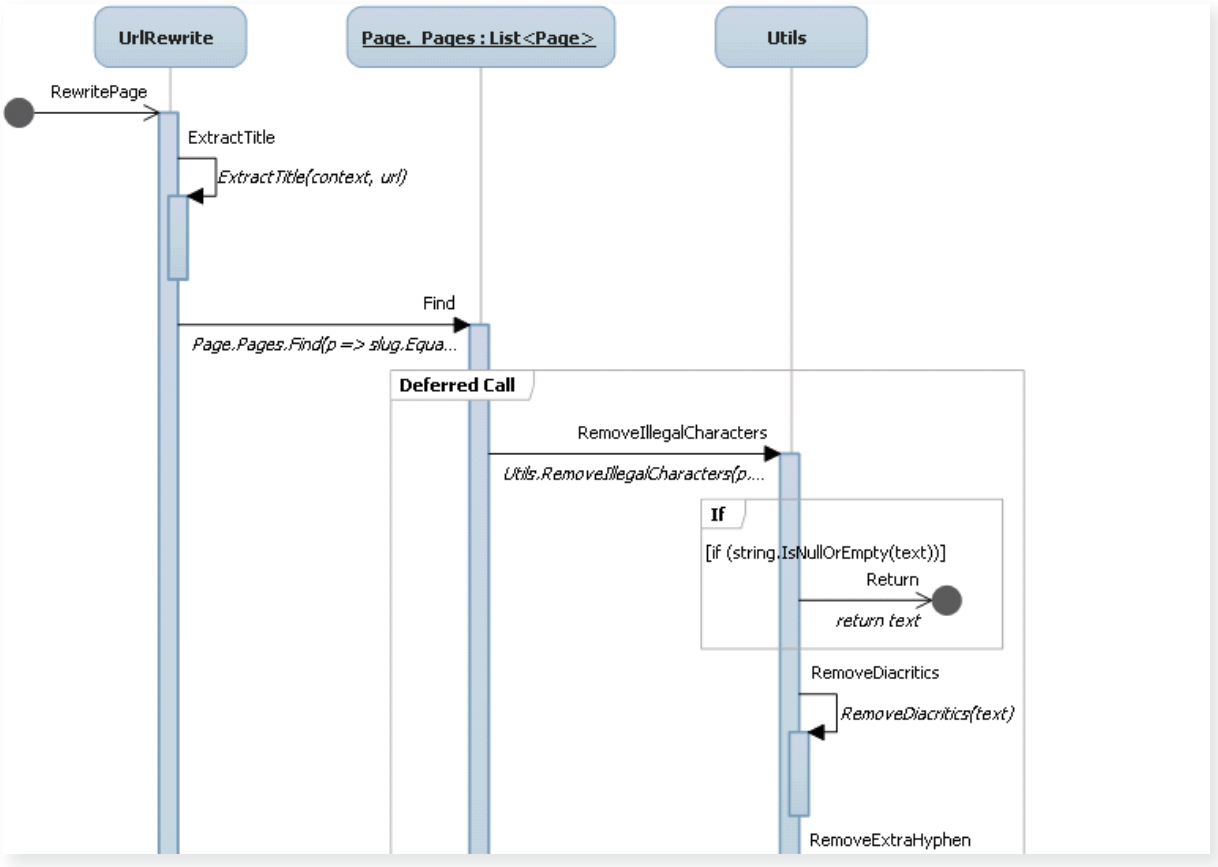


Figure 5 - RewritePage generated sequence diagram (partial).

UML Modeling

You have seen how -- without documentation or code comments -- a developer can dive into and understand an existing application in a fraction of the time it currently takes. The new architecture tools can also help to minimize the lack of system documentation in the first place. Typically, documentation is separate from implementation and it is difficult to tie the two together. Because of this separation, there is a very good chance that they will get out of sync, which leads to the situation experienced in many legacy systems. Visual Studio 2010 and Team Foundation Server 2010 work together to eliminate that problem and provide an upfront design experience that can be tied directly to the documentation so that you can visually drill down from the design to the documentation to the code. This level of traceability helps ensure that the investment in documentation is not wasted and can be leveraged during maintenance work to help reduce the overall cost of system ownership and increase the ROI of any given system.

Visual Studio 2010 introduces UML 2.1 compliant modeling with some features that are only available because of their integration into Visual Studio. Use case diagrams are often the starting point for many applications because they provide a visual guide to the users and their interaction with the system and the functions that the system will provide. One of the benefits the UML models

gain through the tight integration with Team Foundation Server is the ability to link directly to one or more work items or generate a work item from an artifact in a model. This turns the models into living assets that continue to provide benefit instead of taking up space on a shelf.

Figure 6 shows a use case diagram with the Add Post use case circled to show a related work item. This allows developers to drill in from a graphical perspective to the detailed requirements that they will code against and ultimately allows seamless movement from the requirements to code and back. This plays a large part in maintaining the system as well, because a developer can look at the code and determine not only why it was added to the system but how it relates to other features in the system from the user's perspective.

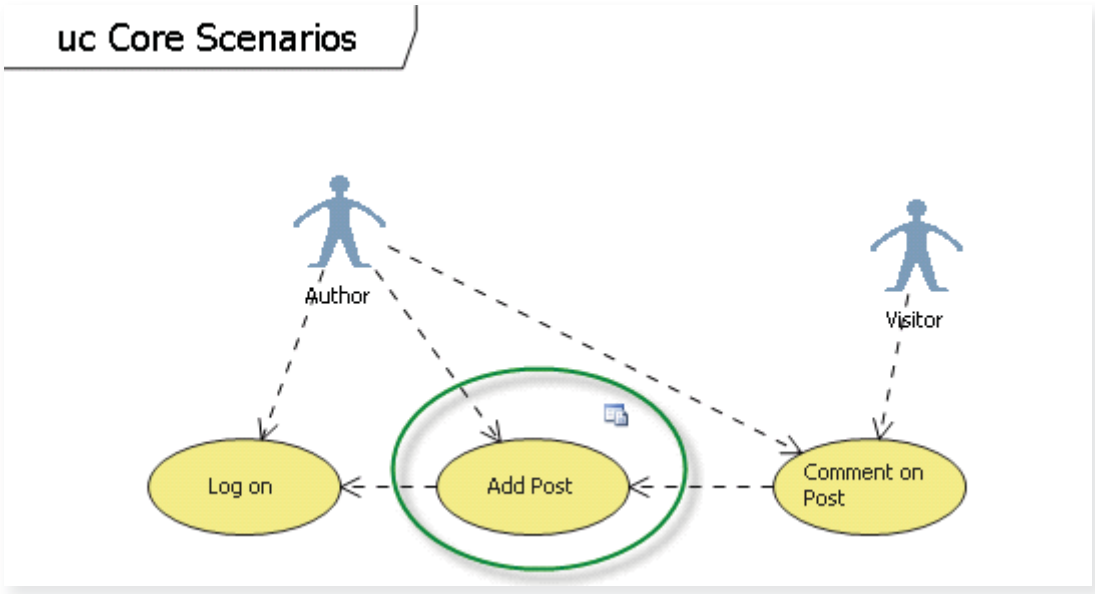


Figure 6 - Use Case diagram with related work items.

You also can add links to other design diagrams such as the component diagram shown in Figure 7.

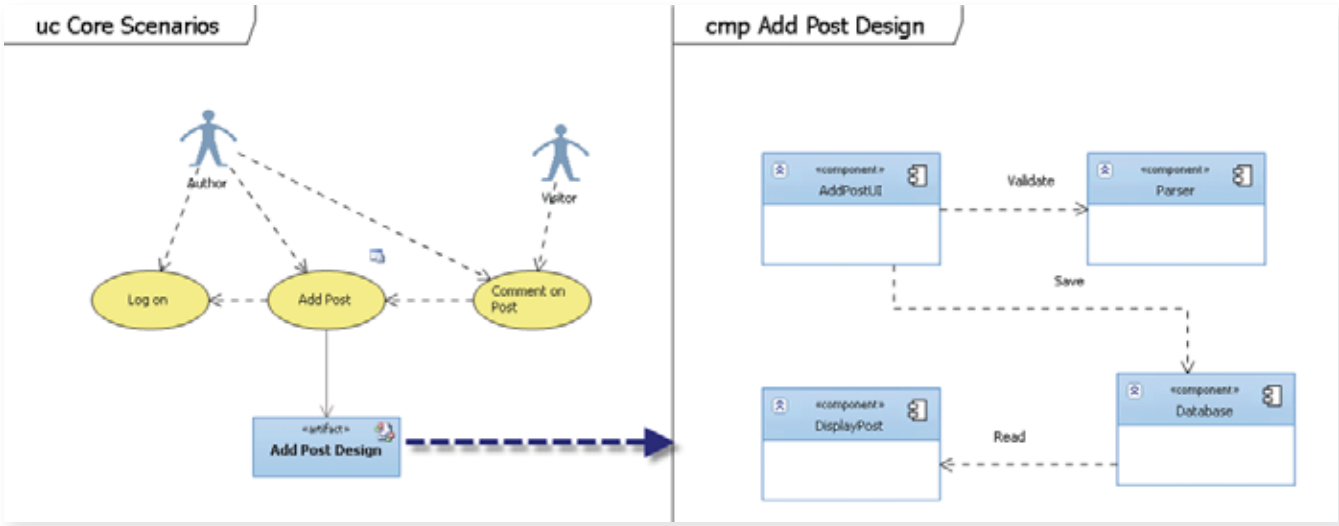


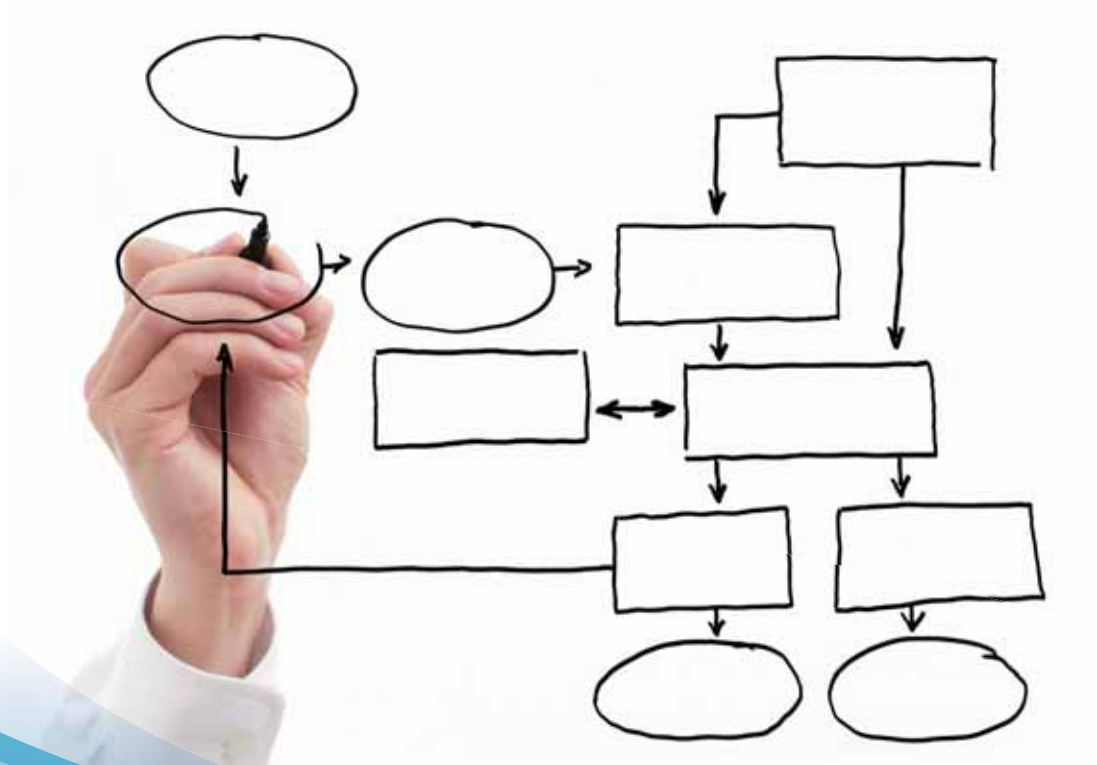
Figure 7 - Linking Use Cases to requirements and designs.

This provides a simple way to link design diagrams to requirements and code, providing an integrated solution that allows you to traverse design models and documents. Because of this, you don't have to create file shares on the network that get stale or worry about someone not being able to find the model or document to update ¬-- it's all centrally located. Plus, if the desire is to link design documents rather than other models to a model, you can do that just by providing a link to the document – either on the web or on a file share. In this way, all of the artifacts you create and that you need to work with are available in the context in which they were created. This decreases the time to understand the applications and allows team members – during development or maintenance – to become productive much more quickly and with less chance of regression bugs occurring.

START UNDERSTANDING YOUR SYSTEMS TODAY

Visual Studio 2010 Ultimate provides teams and organizations the ability to immediately increase the ROI for existing systems because much of the time spent understanding and maintaining these systems can be reduced. The architecture tools allow developers to spend more time writing code and less time trying to figure out what the impact of any given change is going to be. It also allows new developers and teams working on legacy code to quickly and easily understand the application they are working on and the context of that application. So much of the time spent bringing on new team members or working on existing .NET applications is getting developers to understand the application at a technical level. With these tools, the magnitude of that problem is greatly decreased and the productiveness of the developers will be greatly increased.

Streamlining the Development Process with Visual Studio 2010



Most organizations fall into the trap of optimizing a part of their development process while failing to look at the end-to-end development process. Although this low- hanging fruit approach can sometimes result in positive change, it very often misses the big gains that can be made when focusing on the entire end-to-end process. Visual Studio 2010 and Team Foundation Server 2010 provide deep visibility across the entire end-to-end process, creating real change that impacts an entire organization instead of just a single team.

WHY SHOULD YOU CARE?

The term process has, for some reason, become synonymous with not getting work done. It denotes a series of endless documents and rigor that, at the end of the day, means you spent more time doing busy work and less time being productive. This can be the result of poor processes that do not focus on the problems you are trying to solve. Smart processes with appropriate tooling, however, can be used to reduce your workload, increase efficiency, quality and visibility and ultimately increase customer satisfaction. Visual Studio 2010 provides flexible tools that work with, not against, you to deliver your projects on time, on budget, and with high quality. Visual Studio 2010 provides a platform for making improvements that positively affect the overall software development process. At its core, it streamlines the development process by breaking down silos and making communication more efficient and effective.

AN INTEGRATED SOLUTION

One area that tends to slow teams down and increase overhead is tool integration. Teams have one tool for version control, another for bug tracking, a third for project management, one for architecture, other tools for testing and the list goes on. The list of tool combinations is endless, causing teams to spend a lot of time and money maintaining all of the disparate tools. Plus, it makes it difficult to bring on new developers because they have to learn all of the tools. In many cases, the tools don't work well together and have their own set of workarounds. Even if different tools do coexist peacefully, ensuring that all of the different versions of the tools that you are installing are actually compatible is a management headache that can lead to a constant stream of upgrades. Team Foundation Server 2010 (TFS) and Microsoft Visual Studio 2010 eliminates this entire set of problems with a suite of integrated tools.

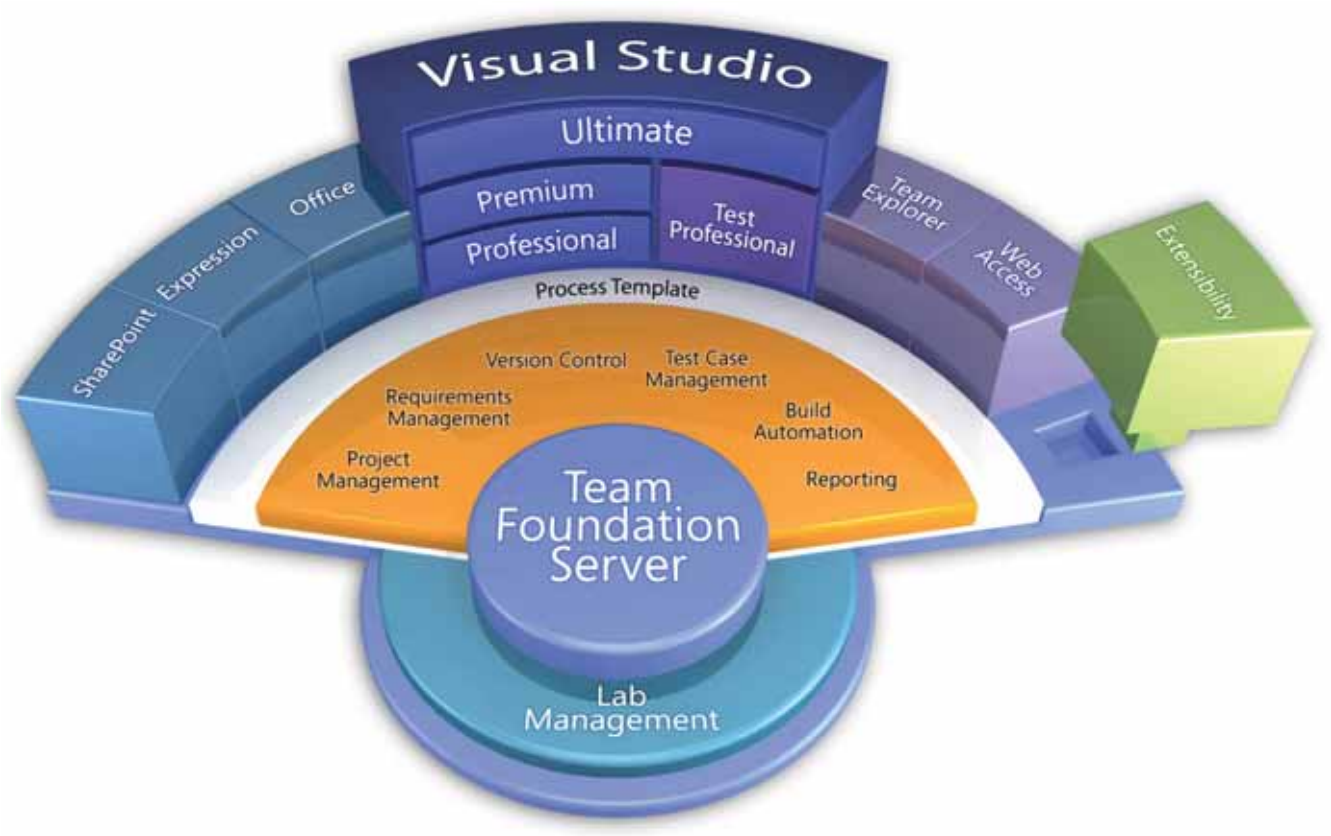


Figure 1 - Team Foundation Server 2010 and Visual Studio 2010, the heart of an integrated solution.

Team Foundation Server provides work tracking, bug tracking, change management, test case management and overall project management as shown in Figure 1. It is backed by SQL Server 2008 and integrates with Visual Studio, Microsoft Office SharePoint Server, Microsoft Test Manager and Microsoft Expression Blend. With the addition of the Eclipse plug-in, even teams using Eclipse can get the same power and benefits as your Visual Studio developers. And if these tools don't meet all your needs, there is a full application programming interface (API) so you can write your own add-ins to work seamlessly with Visual Studio and Team Foundation Server.

Integrated Work Tracking

The Work Item Tracking System in TFS allows teams to easily meet a frequent requirement: traceability. A Work Item is the basic unit for tracking work; it can represent a requirement, task, bug, change request or any other type of artifact your organization needs. By assigning work to developers using the work item tracking system, developers can associate code directly to their assigned work.

Many development teams work under some type of regulatory compliance guideline –HIPAA or SOX, for example – that requires evidence of work streams for audits. This typically takes a large amount of time and work to get right, but with Team Foundation Server it is built-in.

This integration provides additional benefits. For instance, developers and testers can work in a single system instead of many different systems. Because of this integration, TFS has the ability to report on multiple dimensions of any given piece of information. For example, teams can look at the requirements, determine how much work has been done and how much work is left, how many bugs have been filed against a requirement, and how many tests have been executed against the requirement. The out of the box Stories Overview report shown in Figure 2 shows this type of information.

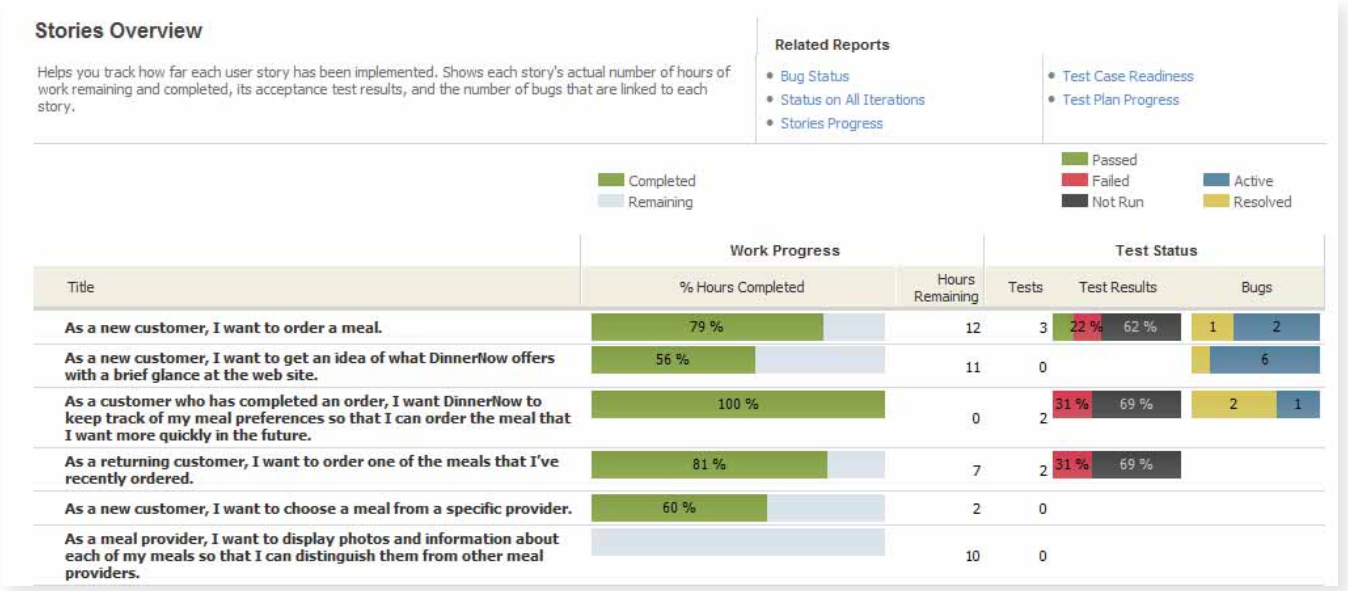


Figure 2 - Stories Overview Report.

This type of report is only available in a fully integrated system.

Break down silos

Teams are about people working together towards a common goal. When organizations streamline one process they sometimes miss the effect on other parts of the team or never realize the full benefit of an improvement. Take architecture, for example. There are many standards out there and many ways of documenting architecture but how many ways are there to make it relevant over the lifetime of the application? It isn't enough to create awesome architecture diagrams if no one else on the team ever sees them and the diagrams sit on a shelf gathering dust. One of the key benefits of an integrated system is that information can be used at the right time by the right people and that it has relevance.

Team Foundation Server is the integrated core that helps project managers, architects, developers and testers work together and communicate effectively. Architects can attach their models to work items, while developers can examine the architecture of the system to help them understand what to build. Not only does this ensure a greater level of compliance between design and reality but it also provides a communication platform through which developers and architects can interact. Testers can gain the same level of integration with the development team. The new test case management features allow testers to create test scripts that developers can access as well. And when a tester files a bug, all of the information that the developer needs to find the cause of the bug is included and attached to the test case. There is no need to export the data from one system and import it into another or have developers work in multiple systems.

Managers are also negatively impacted by the use of multiple systems because they have to gather data from all of these systems and pull it together into a single status report. This is not only time-consuming but frequently inaccurate. With TFS work item tracking, managers get near real-time status on what everyone is doing and when they were doing it. They will know who completed what work on a given day, week or month and what's being worked on at the moment. There is no more need for e-mails requesting status or for project managers to spend hours each week gathering information to create status reports.

Provide visibility

In many organizations, development teams and customers have trouble communicating effectively. They talk frequently yet valuable information is often missed. And, often, the team spends an inordinate amount of time answering customer questions when the information should just be easily available to the customer. By integrating with SQL Server Reporting Services and Microsoft Office SharePoint Server, TFS provides a mechanism to answer many user questions without bothering the team. Figure 3 shows the default dashboard in MOSS 2007. From this dashboard, teams can determine if they will make their release date. The Task Burndown and Task Progress graphs show two views of team progress – by hours and by the amount of work they need to perform. The User Story Progress shows how many requirements are being completed (i.e. are ready for delivery to the customer) and the 7-Day Issue Trend Rates show the amount of work for a team that is Active (currently being worked on), Resolved (finished but not checked off) and Closed (done).

Views such as these provide accurate and timely information that helps answer questions about any aspects of the development process. No longer do teams have to spend hours per week creating these reports. They can all be customized as needed to report on any information

captured by TFS. Comprehensive and detailed reporting also provides information on where the bottlenecks are in the overall process.

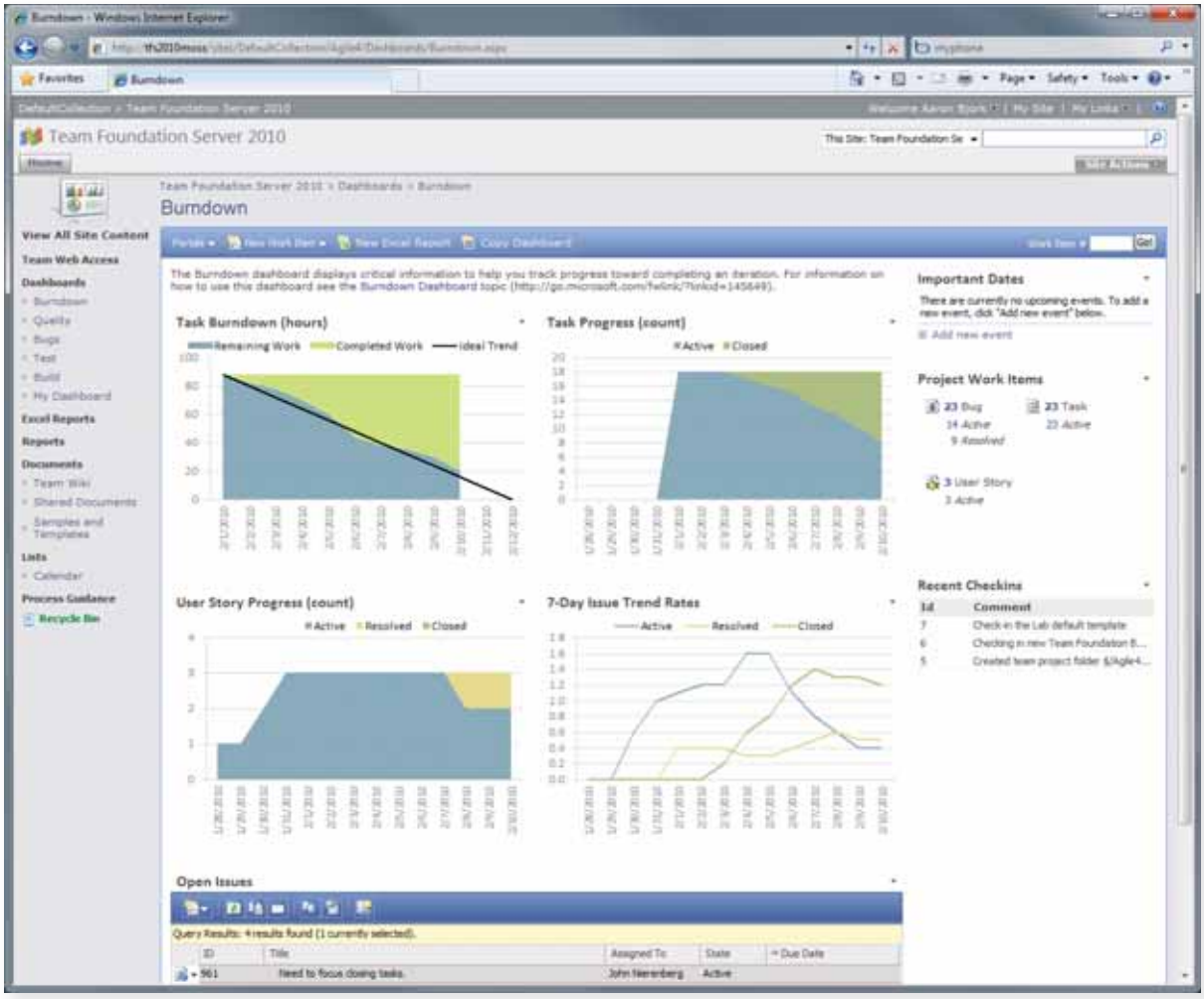


Figure 3 - SharePoint Dashboards

Find and eliminate bottlenecks

Are you going to meet your development deadline? If not, where is the hold up? Do you need to bring more developers on or maybe more testers? Information like this is hard to determine in many situations but TFS makes these questions moot because the answers are right in front of you. Figure 4 shows a typical burndown report with the ideal trend and actual trend.

Looking at this report, teams can determine what their trend is from the very beginning of a project and when they are likely to complete it. Here, the actual trend line is steeper than the ideal trend line which means that the team will make their date – and this could be determined as early as the first few days into the process. Other reports show which state each requirement is in so teams can either add developers or testers as needed. The clear presentation of information lets teams take action to correct problems early in the process.

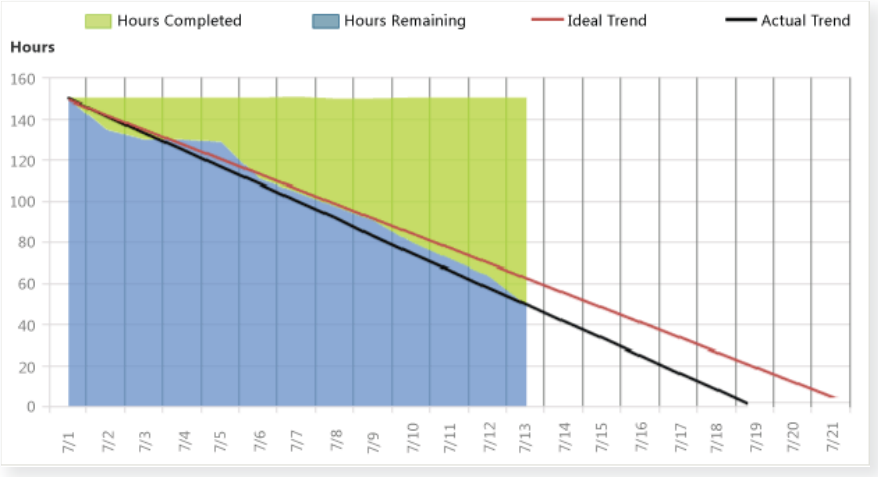


Figure 4 - Burndown Report

Integrate Testers

In too many organizations testers and developers are separated into different organizational groups which prevent them from working together effectively. One reason for this is that developers often don't know what testers are doing (they don't know what the testers expectations of their code is) and testers have no insight into what developers are producing until it is given to them for testing. This does not need to happen, though.

The new test case management features of TFS and Microsoft Test Manager (included with VS Ultimate and VS Test Professional) go a long way toward integrating teams. Now developers can see the test cases that the testers are creating and, when a tester files bugs, they are providing detailed trace logs for direct attachment to the bug associated with the test case. This has the capability to really change the game with how developers and testers work together. With the 2010 release, developers and testers can work together as a team to efficiently and effectively find and eliminate bugs.

STREAMLINED PROCESSES BENEFIT EVERYONE

When teams communicate and work together effectively, everything is better. A smooth process actually reduces the amount of work that teams have to do and improves productivity. A reduction in disparate tools means redundancies are eliminated and efficiency is improved because everything can be found in one place. The streamlining goes far beyond a simple reduction in tools: It is the integration of tools that provide the power. Tools should be purposeful, but they also should be supported by the same platform with results available to everyone who needs them. Team Foundation Server 2010 and Microsoft Visual Studio 2010 deliver on the promise of a platform for improving your process.

Eliminate Bugs with Visual Studio 2010

An enormous amount of time is spent debugging problems in code – even simple problems take time. The new debugging tools in Visual Studio 2010 Ultimate give developers a successful outcome to any debugging session. But more than that, developers can do it in a fraction of the time previously required. Elusive non-reproducible and intermittent errors can be eliminated quickly, saving time and money and increasing customer satisfaction.

Quality considerations are of paramount importance in the software business. It has long been understood that the costs of defects rise significantly when they are not caught early. The agile community, for example, refers to "failing fast" as an important rallying cry and, as a result, sees continuous integration and testing as fundamental practices. Visual Studio 2010 provides an aggressive set of innovations around test and quality [see the white paper, "Building Quality Applications"] to help software teams deliver superior results.

A key theme of the Visual Studio 2010 release is "no more no repro" – the concept of ensuring that bugs found by testers can always be reproduced by the people fixing the bugs. Why is the inability to reproduce a bug such a big issue? Because it costs time and money to fix bugs. Or, in worst case scenarios, bugs simply aren't fixed because the developers can't re-create the bug in a controlled environment. Much of this problem comes because of how bugs are reported. For example, a phone call might come in to a developer indicating a bug was discovered with just vague information about what the user was doing. Other times, a bug occurs on one machine and not another-- the infamous "it works on my machine" syndrome. With Visual Studio 2010, Microsoft has taken huge strides in eliminating this problem altogether.

THE ACTIONABLE BUG

How do you solve the problem? There are a number of approaches. The first one is ensuring that bugs filed against a system are reproducible through whatever means are available. Accurate information that is actionable must somehow be passed on to the developers. Vague notions of what caused the bug are not good enough. With the introduction of Microsoft Test Manager in Microsoft Visual Studio Test Professional 2010 (also available in Visual Studio 2010 Ultimate) that vision is now a reality. Whether running a manual test using the Microsoft Test Runner or executing automated or functional tests, testers can record videos of the tests being run and, when a test fails, the video can be provided to the developer to show exactly what led up to the bug and what the user was doing at the time. This is critical information for fixing any bug, and it's acquired through the use of Data Collectors.

Data Collectors allow the testing infrastructure to pull any data necessary from the system under test to help developers. There are several built-in Data Collectors such as the Video Recorder and Test Impact Analysis collector, as well as more comprehensive tools such as the System Information collector and the IntelliTrace™ collector.

Not only are the built-in tools capable of providing data the developer does not currently have access to, but the collectors are extensible. Individuals or teams can create their own collectors that gather whatever information is necessary for them. This allows for unlimited possibilities when collecting information to understand what went wrong. The Data Collectors can also be used in testing runs from Visual Studio 2010, as in a long-running performance test or a test launched from Team Build. Data Collectors are not limited to use in Test Manager.

Finding the Bug

Once a developer understands what is going on, they have to debug the problem and find the specific place in the code that needs fixing. Debugging code can be hard. It's even more difficult if that code is unfamiliar legacy code. This makes debugging time-consuming as well. Too often, problems are reported to developers that can't be recreated so bugs are left unfixed until they become frequent enough that they can be easily tracked down. IntelliTrace™ provides a new tool for developers that allows them to "replay" sessions. A session can be a prior debugging session or it can be the result of a tester running tests cases against code. This means that developers can go back in time to see what an application was doing at any point in the session. Developers might even be able to isolate failure detail without access to the source.

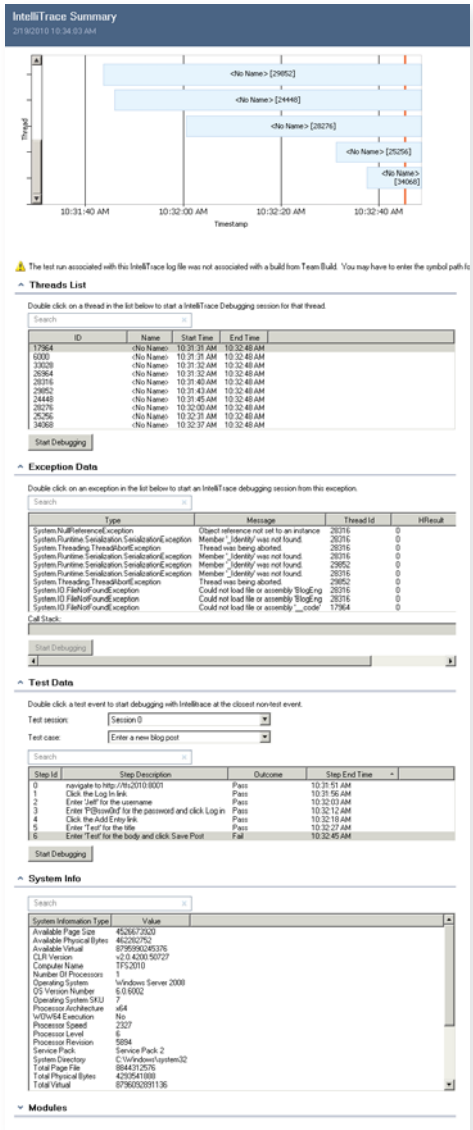


Figure 1 – The IntelliTrace™ Log

The increased efficiency for developers is huge. No longer will they have to run the application many times to get the error to re-occur. Instead, they simply replay the session in which the error occurred. Better yet, developers can start a new application instance, attach an IntelliTrace™ log to the running application, and step through it so they can debug against the recorded session. Figure 1 shows the summary page of an IntelliTrace™ log which gives an idea of how much information is collected and returned to the developer.

The IntelliTrace™ log shows all of the threads that were running and lets developers choose the desired thread. If the bug was reported from a manual testing session using Test, the developer

can go to the section of code that was executing and just examine that instead of stepping through everything. Not only does IntelliTrace™ reduce the amount of time it takes to find bugs, it also reduces the frustration involved in finding them.

Fixing the Bug

The IntelliTrace™ capability in Visual Studio 2010 Ultimate provides quick, rich information, expediting developer-tester coordination on defects. A lone developer might even feel like he has acquired his own dedicated test expert. Figure 2 shows an IntelliTrace™ debugging session.

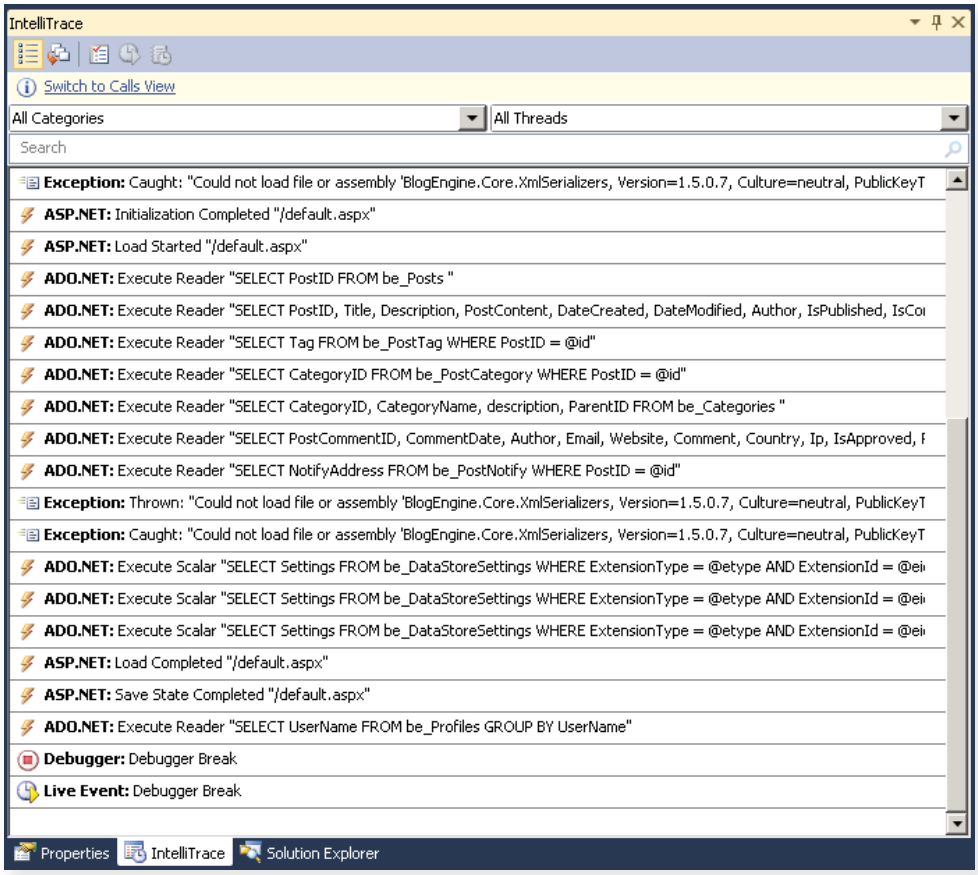


Figure 2 - An IntelliTrace™ Debugging Session

Being able to see everything that happened during a testing or debugging session means that developers don't have to set 50 breakpoints and continuously hit F5 (Run), examine the data provided at a breakpoint, and then move on to the next breakpoint. This process is especially painful if the developer misses the spot in code where they needed to stop—because they have to start all over again!

An interesting side effect of having all this information is that it presents the opportunity to find bugs that otherwise wouldn't have been found. In Figure 2 there are other Exceptions being thrown as part of the normal course of the application running. So while the developer was not looking for this particular issue, they found it anyway! A normal debugging session would not show you what happened prior to the point at which you broke into the code; having this view means you will be able to find bugs that would not have been found previously.

Using IntelliTrace™, developers can search for methods or classes that have been executed as part of the test or simply look through the IntelliTrace™ logs to discover where the problem occurred. In Figure 3, the developer is trying to discover why incorrect information is being loaded as part of a user control.

Figure 3 shows that by simply looking through the log, the developer pinpointed where the user control was being loaded and saw that it was reading from the cookie information when it shouldn't be.

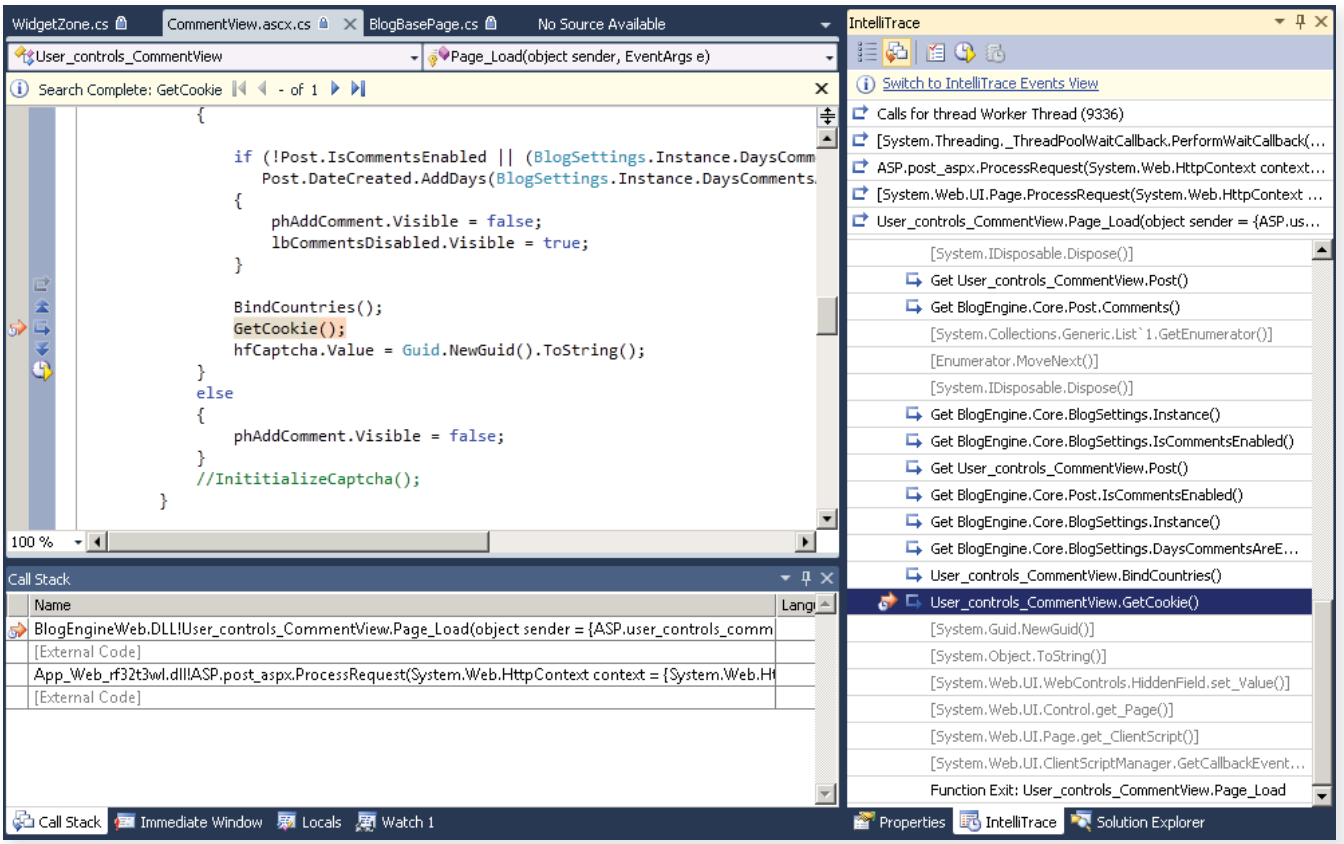


Figure 3 - Debugging with IntelliTrace™

At this point, the developer can step into the method and see what is being loaded and why – and what the values were during the debugging run.

INTELLITRACE™ AS A TIME SAVER

An enormous amount of time is spent debugging problems in code – even simple problems take time. The new debugging tools in Visual Studio 2010 Ultimate give developers a successful outcome to any debugging session. But more than that, developers can do it in a fraction of the time previously required. Elusive non-reproducible and intermittent errors can be eliminated quickly, saving the time of testers and developers. Team leaders can know that practitioners will be quickly back on task with the next feature or release when IntelliTrace™ is in their toolkit.

SEE WHAT OTHERS ARE SAYING: Minitab & Wintellect

Minitab®

Software Developer Reduces Costs and Streamlines Development with Integrated Tool Set

Minitab, a developer of statistical software, wanted to simplify and accelerate development. It deployed an integrated solution that includes Microsoft Visual Studio 2010 Ultimate, Visual Studio Team Foundation Server 2010, and Visual Studio Team Lab Management 2010. Now, Minitab can take advantage of features such as IntelliTrace and Microsoft Visual F# to boost productivity, reduce development and licensing costs, and speed the time-to-market for new offerings.

"IntelliTrace will be a significant debugging tool for us because we can use it to capture a testing workflow and then attach the workflow to a bug report. In addition, IntelliSense in Visual Studio 2010 offers new capabilities in error reporting, parsing, and background processing that will certainly speed development. We can also use Visual Studio Team Foundation Server 2010 to manage software check-in and share test workflows recorded in IntelliTrace log files." - **Matt Kowalski**, Manager of Database Development, Minitab

Wintellect® Know how.

Consulting Firm Expects to Accelerate Debugging by at Least 25 Percent with New Tool

Software consulting firm Wintellect wanted more innovative processes for debugging that could help accelerate the resolution of bugs and enhance the quality of code. The company found what it needed in the Microsoft Visual Studio 2010 Ultimate development system, which includes the new IntelliTrace debugging feature. Wintellect now has the tools it needs to debug applications 25 percent faster, improve collaboration, and reduce costs.

"To my knowledge, nobody else has any kind of debugging tool like IntelliTrace in Visual Studio 2010 Ultimate." - **John Robbins**, Co-founder, Wintellect

Becoming more effective & efficient at testing with Visual StudioTest Professional 2010

There are two key goals for any Quality Assurance and Test Professional: to validate the quality of an application and to improve overall software quality. Yet testing effort is often wasted on necessary repetition, unclear test prioritization, and downright human error. Visual Studio Test Professional 2010 lets you achieve these two goals efficiently and effectively using the right testing and diagnostics tools to create superior solutions and interactions. It also eliminates waste across your application lifecycle by breaking down the silos within teams and delivering key integration capabilities to ensure teams are more productive.

Professional testers have two key goals: to validate the quality of an application and to help improve the quality of that application. Many issues come into play, which make these two straightforward goals very difficult to accomplish. Much testing effort is wasted on necessary repetition, unclear test prioritization and downright human error. These common issues make testing ineffective and inefficient. For example, what makes a good bug? Why is one bug actionable and another isn't? Why is information that is good enough for one problem not good enough for another? Bugs filed without enough information cause deterioration in the relationship between developers and testers, which causes a downward spiral in communication and collaboration. This is just one aspect of testing. Testers often have a difficult time reporting on the quality of an application because there is no holistic view it. Visual Studio Test Professional 2010 is designed to bridge these types of gaps that exist in many software development environments and to help ensure that software is released with the highest quality.

Today

One of the major causes of frustration between developers and testers is the fact that many bugs cannot be reliably reproduced. If they can't be reproduced, they cannot be easily fixed. This leads to animosity between developers and testers and wastes time as testers try to reproduce the bug and developers try to find a non-reproducible bug. When testers cannot provide actionable bugs, developers do not value their contributions. In the majority of situations, Microsoft Test Manager 2010, the key testing tool in Visual Studio Test Professional 2010 (also included with Visual Studio 2010 Ultimate), ensures that actionable bugs are created and that they can be tracked to resolution.

What is an actionable bug?

An actionable bug is one that has enough data with it for the developer to reproduce the bug and determine where the failure occurred. In addition, the desired behavior (that is, what the code is supposed to do) should be identified so the developer knows when they have fixed the bug.

A common reason bugs are not reproducible is the quality of the bug report. Bug reports take time to properly create even when there is sufficient information. In many cases, there isn't sufficient information or the tester cannot remember everything he did. This is particularly noticeable when a tester is performing exploratory testing and doesn't have a defined series of steps. Time is a critical component of the testing process.

When a bug is caused by specific environment settings, it becomes that much more difficult to reproduce. When one setting in one environment causes a problem, the chances of reproducing it are very slim without access to that specific environment. There are many reasons for this but the most obvious is that the developers don't know any of the characteristics of the machine on which the test was run or have access to that machine.

One test manager noted in regard to testers, "in a time when they are required to do more they are spending a significant amount of time retesting the same features over and over again as they are forced to regress and verify bug fixes." With the limited amount of time available to testers, they need to be as efficient as possible and run only those tests most likely to reveal bugs. When developers make small changes to code, testers need a way to sift out the test cases which likely haven't been affected by code changes. With so many test cases and so little time, testers may not get to those test cases which are most critical.

TOMORROW

Microsoft Visual Studio Test Professional 2010 provides the tools to solve each and every problem mentioned here. It provides powerful yet easy to use tools which allow teams to manage test plans, run tests, file actionable bugs, and verify that bugs are fixed. It also provides the ability to report on many different aspects of the testing effort. Because these features are deeply integrated with Team Foundation Server 2010 (TFS), they leverage the capabilities of TFS to provide full traceability from requirements to test cases and bugs to code. Plus, it offers the advanced reporting features of the data warehouse and analysis cube. But how does it make testers more efficient?

At the core of Microsoft Visual Studio Test Professional 2010 is Microsoft Test Manager (Test Manager). Test Manager allows testing teams to manage test plans, test suites, and test cases down to individual test run results. It does not require Visual Studio but instead plugs into Team Foundation Server 2010 which integrates the testing effort directly into the development process and provides for easy communication between developers and testers. This is the tool through which testers execute test cases, file bugs and verify bug fixes. It helps testers become more efficient when filing actionable bugs.

In addition, the modern and lightweight Test Manager interface gives you the ability to focus on your most frequent tasks through a streamlined plan-test-track workflow design. Figure 1 shows Test Manager in its ***Plan*** mode.

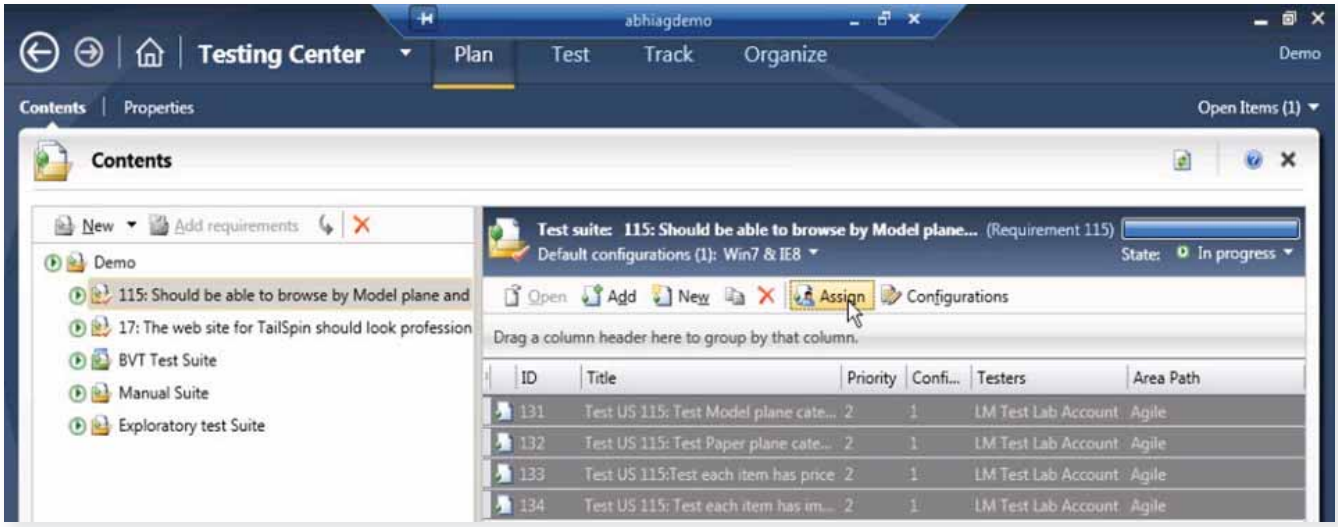
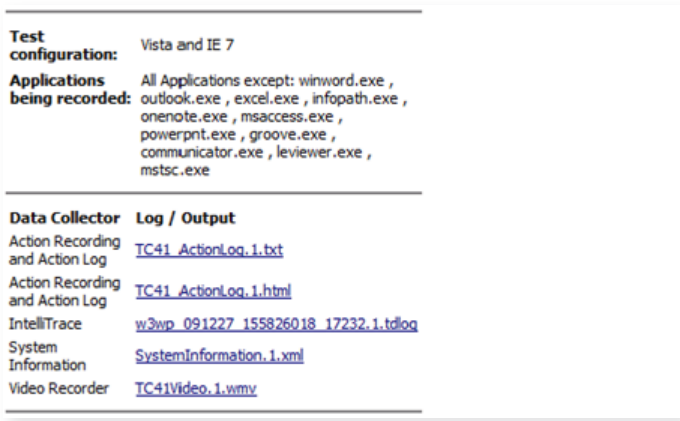


Figure 1 – Microsoft Test Manager 2010's Plan–Test–Track Workflow Style Interface.

What makes an actionable bug? From a developer's perspective, knowing what the tester was doing that led up to the bug is critical. They need to know the data entered into the application and the conditions of the environment in which the test was being executed and the build of the software the test was executed against. Manually entering all of this information is time-consuming and error-prone, which is why traditional methods fail. Test Manager takes the work out of the process by doing it for you. Figure 2 shows the reproduction steps of a bug filed with Test Manager.

Step no.	Result	Title	Video links
1	Passed	**Log on as Jeff (Editor)**	
1.1	None	**Navigate to http://vs2010beta2:8001** Expected result: Blog Engine welcome page is displayed	
1.2	None	**Click the Log In link** Expected result: Log in page is displayed	
1.3	None	**Enter the username as 'Jeff'**	
1.4	None	**Enter the password as 'P@ssw0rd' and click Log In** Expected result: Welcome page is displayed and you are logged in as an editor. The administration box is displayed and it does not have an entry for 'Users'	
2	Failed	**Click the Welcome to Blog Engine .NET 1.5 blog post** Expected result: The post details are displayed and the Name = 'Jeff' and the e-mail address = 'Jeff.Levinson@nwcadence.com' Comments: Previous users information was displayed Attachments: TC41IterationIStep2Screenshot1.png	Video:00:00:31



Test configuration:	Vista and IE 7
Applications being recorded:	All Applications except: winword.exe , outlook.exe , excel.exe , infopath.exe , onenote.exe , msaccess.exe , powerpnt.exe , groove.exe , communicator.exe , leviewer.exe , mstsc.exe

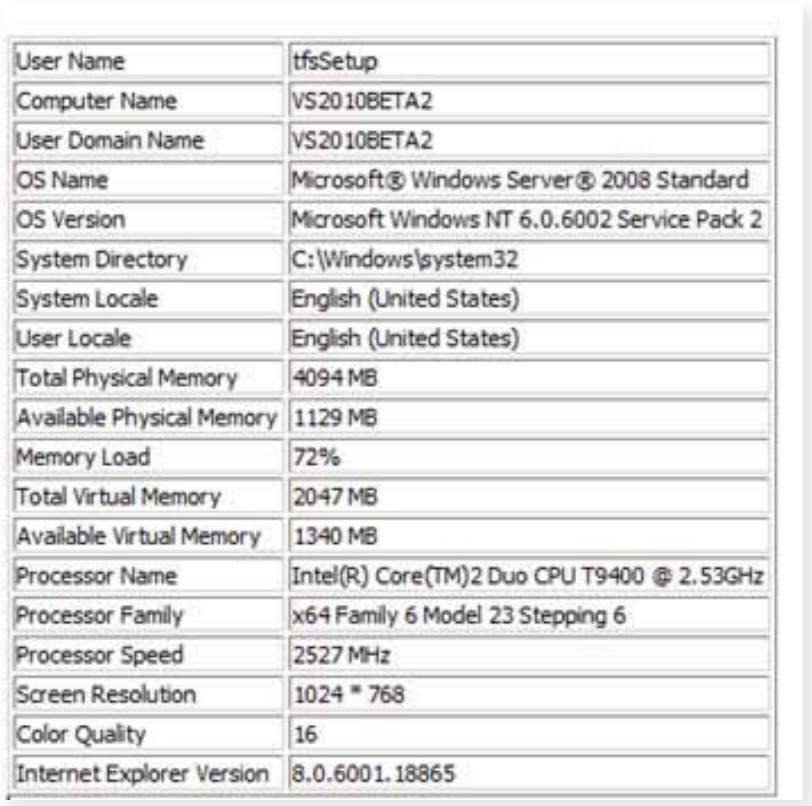
Data Collector	Log / Output
Action Recording and Action Log	TC41_ActionLog.1.txt
Action Recording and Action Log	TC41_ActionLog.1.html
IntelliTrace	w3wp_091227_155826018_17232.1.tdlog
System Information	SystemInformation.1.xml
Video Recorder	TC41Video.1.wmv

Figures 2 & 3 - Repro steps of a bug filed with Microsoft Test Manager 2010

Here, a developer knows exactly which test case was being executed (the bug is linked to the test case), which steps passed and which step(s) failed. A video was recorded as part of this test case (video recording is just one of the data collectors available to testers) and is time-indexed to the point of failure so the developer can see exactly what went wrong. The tester has also attached a screenshot of the failure associated with the specific step in the screenshot.

When testers file bugs, developers have access to additional information gathered behind the scenes courtesy of data collectors. Figure 4, for example, shows information about the system that was being tested.

If this information isn't enough, additional data collectors can be created to gather the information that is important to your organization or test process. Data collectors provide advanced diagnostics capabilities so you can capture any information you want from a target machine through a simple extensibility mechanism.



User Name	tfsSetup
Computer Name	VS2010BETA2
User Domain Name	VS2010BETA2
OS Name	Microsoft® Windows Server® 2008 Standard
OS Version	Microsoft Windows NT 6.0.6002 Service Pack 2
System Directory	C:\Windows\system32
System Locale	English (United States)
User Locale	English (United States)
Total Physical Memory	4094 MB
Available Physical Memory	1129 MB
Memory Load	72%
Total Virtual Memory	2047 MB
Available Virtual Memory	1340 MB
Processor Name	Intel(R) Core(TM)2 Duo CPU T9400 @ 2.53GHz
Processor Family	x64 Family 6 Model 23 Stepping 6
Processor Speed	2527 MHz
Screen Resolution	1024 * 768
Color Quality	16
Internet Explorer Version	8.0.6001.18865

Figure 4 - System Information is automatically caputred into the bug.

Test Manager also provides a new feature called IntelliTrace™, which records the entire debugging session of the application being tested. This allows a developer to literally play back the testing session so he can step through the code without running the application. Not only does the developer see the end result which is the filed bug, the IntelliTrace™ log lets him see all of the values of all of the variables during the test run and any exceptions thrown during the test run.

A further refinement of this process is the addition of Lab Management. Lab Management uses virtualization technology to accelerate the set-up, tear-down and restoration of complex virtual environments to a clean state. It solves the no-repro problem by allowing testers to file rich bugs with links to checkpoints that developers use to recreate bugs in complex environments. With one click of a button, a developer can launch a virtual environment that exactly matches the one in which the bug was found. Finally, it extends build automation dramatically by automating virtual machine provisioning, build deployment and build verification in an integrated fashion. This approach lets teams embrace change and be more agile in an ever-demanding world.

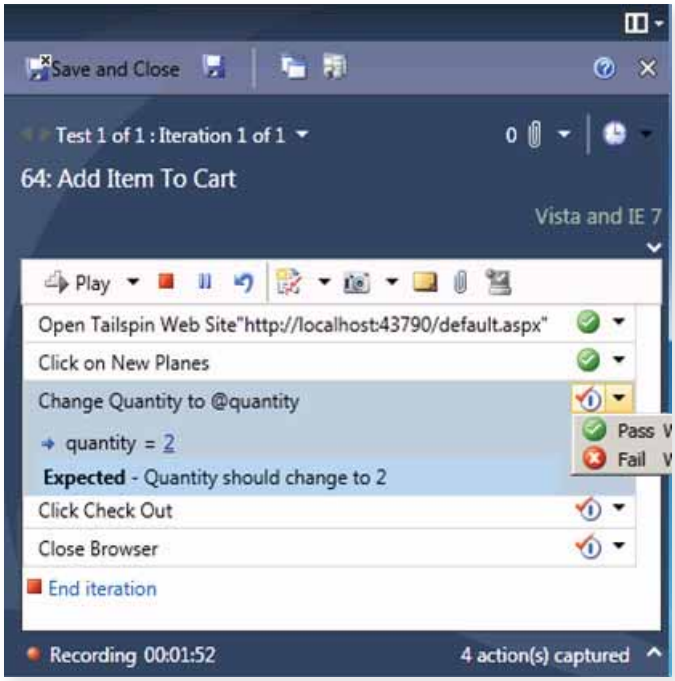


Figure 5 – Data-bound test iterations are captured for Fast Forward for Manual Testing uses.

As mentioned previously, testers spend an inordinate amount of time testing the same things again and again because of the need to verify bug fixes or perform regression testing. Test Manager works to reduce the amount of time this takes. There are two key features to improve efficiency and effectiveness – Fast Forward for Manual Testing and Test Impact Analysis. This reduces the time needed to verify bug fixes while still being repeatable and allows testers to verify many more bugs in a given period of time.

Test Manager allows you to easily capture data-driven test iterations on your first pass using Microsoft Test Runner. These recorded and data-bound actions are used in subsequent passes for Fast Forward for Manual Testing, a powerful feature to help you quickly get to where you need to be.

Fast Forward for Manual Testing uses a pre-recorded action log of manual tests to navigate to the right place within your test case to continue verification steps in subsequent passes. There is no need for user interaction when replaying the test. Fast Forward for Manual Testing will play back the test case as fast as the application will respond.

What about the situation where developers may inadvertently introduce regression bugs? How do testers know what to test? This is where the integration between the testing tools and development tools shines with a feature called Test Impact Analysis (TIA). Instead of spending hours or days determining which tests should be re-run, you simply access a list of recommended tests due to code changes and know which bugs are addressed in a given build. Now you can drive the right decision about when to pick up a new build, greatly reducing wasted time and effort.

Test Impact Analysis (TIA) correlates the code executed for each test within the overall test. When a change is made to the underlying code TIA identifies the tests (or test cases) that have previously executed that code. This allows testers to perform risk analysis on the test cases and prioritize them for use in regression testing. This applies to Manual Testing, Automated Testing and Unit Testing.

Reporting on the quality of applications is difficult. In large part, this is because the code, test results and requirements are not typically integrated into a single system. Visual Studio Test Professional 2010 takes full advantage of this integration along with the Microsoft Business Intelligence tooling to provide comprehensive dashboards on the overall quality of your application – and they are built-in. Leveraging the power of SharePoint, these reports can be exposed to those stakeholders who need to see this information on a regular basis. Standard reports are available, such as number of failed tests versus the number of passed tests and the number of bugs created. There are fifteen standard reports dealing with application quality. One of the most compelling is the Stories Overview report (Figure 6) because of its multi-faceted approach to data consolidation.

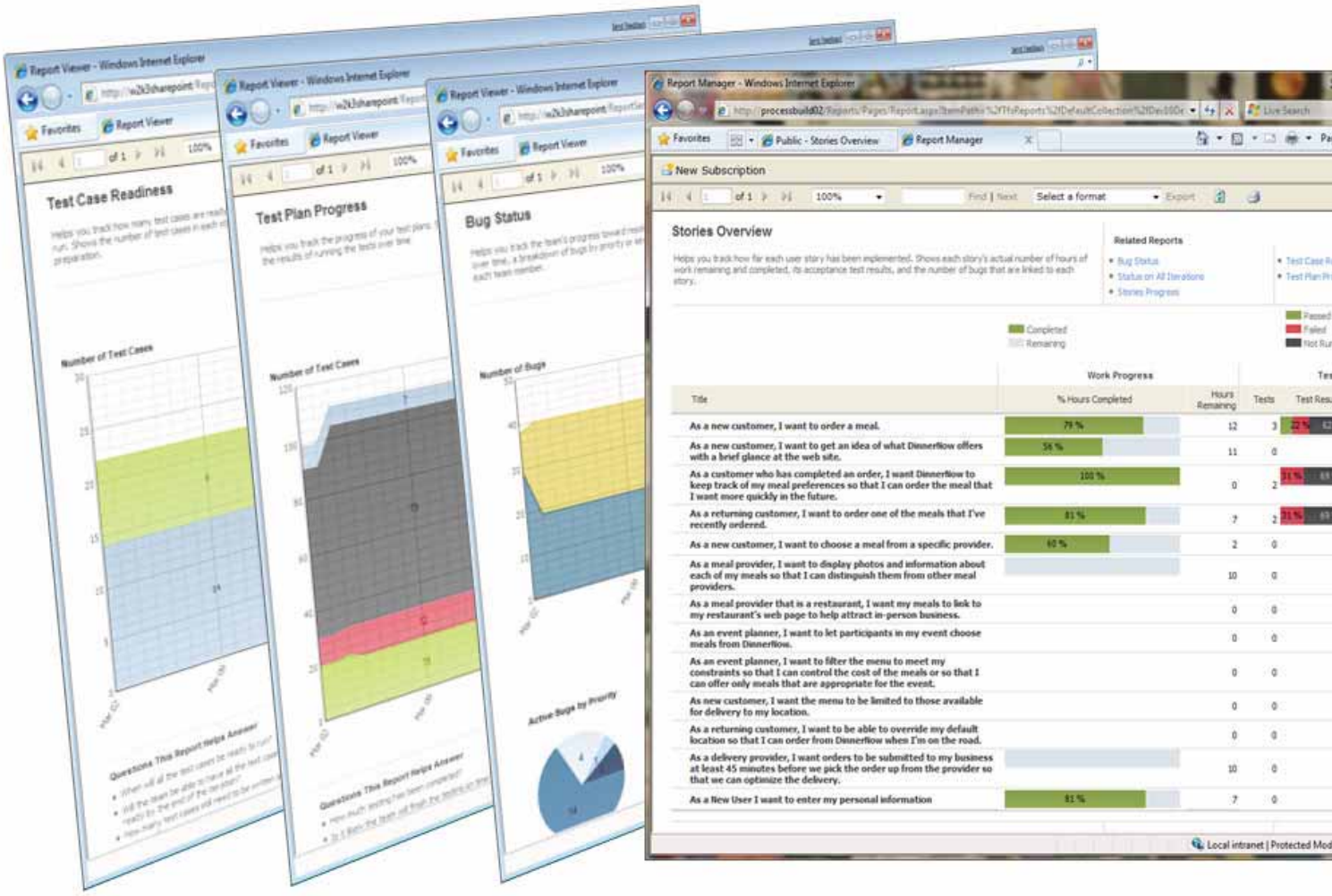


Figure 6 - A sample of testing reports

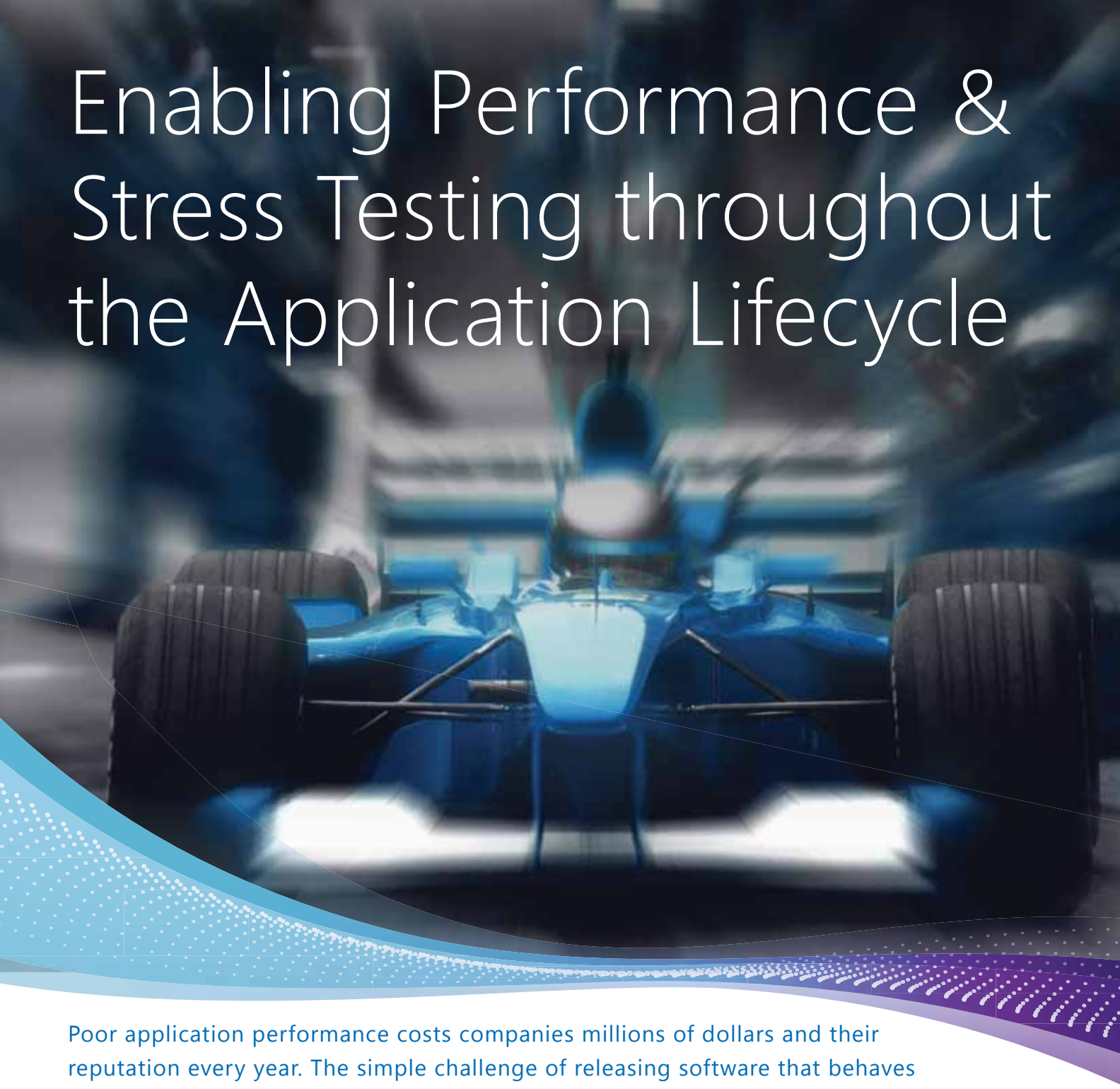
This report maps the testing status to the individual requirements of a system, the number of bugs filed against those requirements, their status, and the progress in completing the requirements. Users can see at a glance the number of test cases for a requirement, how many have been

executed, how many have passed and failed, and the number of active, resolved and closed bugs against each requirement.Combined with reports on builds and code churn, an overall picture of application quality can be obtained. This gives testers better visibility into where they should spend their testing effort to ensure that the testing offers the greatest benefit for the application.

SUMMARY

Microsoft Visual Studio Test Professional 2010 gives testers the ability to do more with less. It allows testers to focus their efforts on what needs to be tested and gives them the tools to create actionable bugs when they find a bug. It reduces the back and forth between developers and testers, improves communication, and ultimately improves productivity. Activities that require a large amount of testing time – such as writing bug reports, verifying bug fixes, and determining which tests to run as part of a regression suite – are now greatly reduced. As a result, testers have more time to spend on complex tasks such as exploratory testing or increasing the number of test cases to cover more areas of the application. Your testers contribute even more to releasing quality applications for your customers with Microsoft Visual Studio Test Professional 2010.

Enabling Performance & Stress Testing throughout the Application Lifecycle



Poor application performance costs companies millions of dollars and their reputation every year. The simple challenge of releasing software that behaves predictably, consistently and responsively continues to be a big one. Internal applications with poor performance add another layer of cost - both in lost productivity and missed deadlines. Microsoft Visual Studio 2010 provides the tools to measure, improve and verify application performance under the most demanding conditions so that your application performs predictably regardless of the situation.

Poor application performance costs companies millions of dollars and their reputation every year. Why this happens is fairly straightforward – customers can't get to the website they want or they can't purchase products because the application either responds poorly or doesn't respond at all. And that's just for web-based applications. Internal company applications that suffer from poor performance waste time, and time is expensive – both in lost productivity and missed deadlines. In the past, poor performance might not have had as visible an impact but today poor performance is extremely expensive. It doesn't have to be that way though. Microsoft Visual Studio 2010 provides the tools to measure, improve and verify application performance under the most demanding conditions so that your application performs predictably regardless of the situation. Performance testing, in many cases, is not performed at all. The chief reason for this is not the time available but the cost and complexity of the tooling required to undertake performance testing.

A FAMILIAR ENVIRONMENT

One of the key benefits to using the integrated performance testing tools in Visual Studio 2010 is that they work in an environment that developers are used to. Visual Studio provides an intuitive interface for constructing Web Tests, Unit Tests and the associated Load Tests. And all tests are based on the same extensible testing framework which has matured over the previous five years into a robust and well-supported framework.

In addition to the integration with Visual Studio, customizing tests do not require you to learn a new language – they are backed by the Microsoft .NET framework. Many other industry standard performance testing tools require you to learn a new scripting language or development language and a new Integrated Development Environment (IDE). With the Visual Studio testing tools, you can be up and running in a short period of time with a minimal learning curve.

Finding Performance Problems Early

Performance problems can be introduced into applications in a variety of different ways. Often, performance issues occur because of the chosen architecture. Poor architectural structure leads to bottlenecks in code which can slow the entire application down – even the addition of faster hardware doesn't help when the software isn't architected correctly. Fixing a problem that originates with the application architecture can be difficult and costly; finding the problem early in the process can save a considerable amount of time and cost.

Modeling User Behavior Using Web Performance Tests

Web Performance tests are one of the key components of performance testing in Visual Studio 2010. This feature was introduced in Visual Studio 2005; Microsoft has taken an already solid tool and enhanced it to provide greater ease of use and better functionality.

Web Performance tests record browser traffic at the HTTP layer, which can then be run in a load test to simulate user behavior. These tests are lightweight and very efficient at generating a large amount of load.

Web Performance tests are rich in functionality. Ajax and page resources are handled automatically, and you can easily add conditions and looping to your tests without writing code. In addition, dynamic parameters, cookies and authentication are handled for you and tests can be bound to data to create flexible data driven tests. In addition to these features, Web Performance tests are extensible. If, for example, you need to add complex validation rules that are not met by the built-in rules, you can add your own. Need to do custom dynamic parameter handling for the application you are testing? You can create a plug-in using minimal code. There are numerous extensibility points which let you customize any aspect of the Web Performance testing experience. Figure 1 shows the output of a Web Performance test.

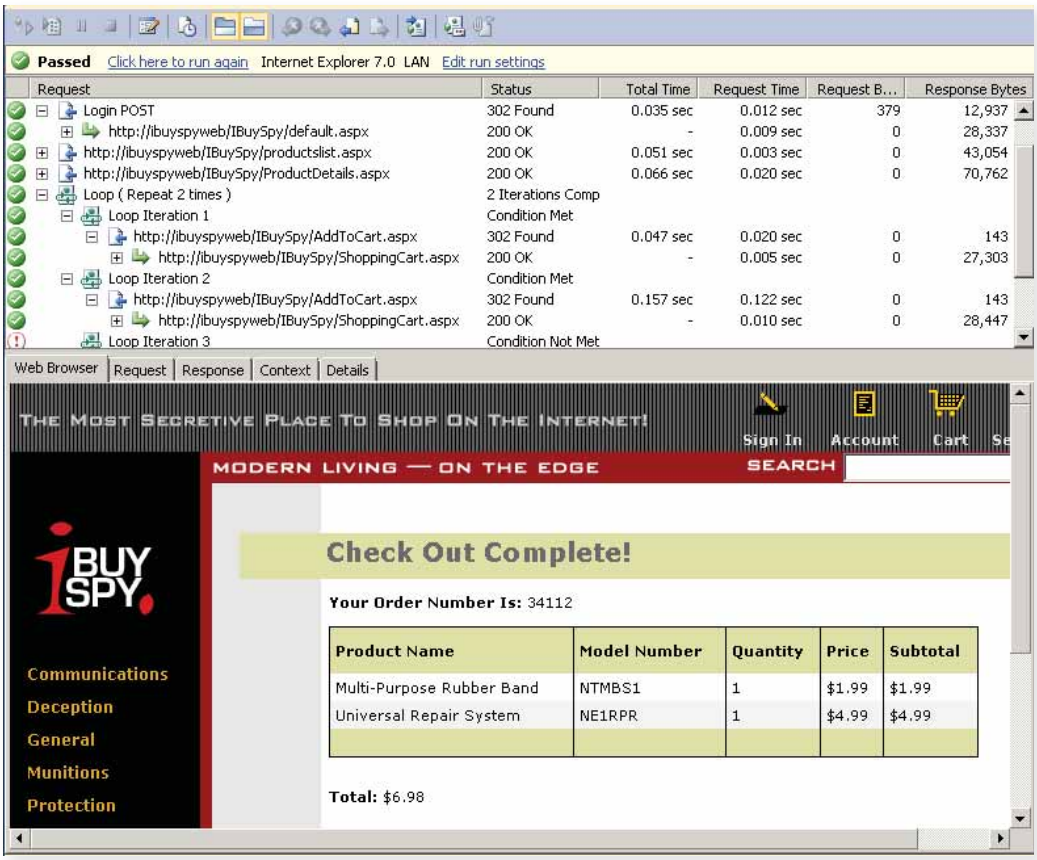


Figure 1 - Web Performance Test Output

Using Unit Tests to Drive Load

One of the great benefits of the VS 2010 performance and load testing tools is the ability to work with unit tests. This means that you don't need a full-up user interface or even all of the pieces of an application to test the performance of an application. In addition, you can perform tests earlier than normal in the process if there is a suspected performance bottleneck. This is also another benefit when working with web services: As more applications begin working with web services which back user interfaces, the performance of those services can be independently tested to ensure they perform well.

Creating Load Tests

How does performance testing work in Visual Studio 2010? First, you can create either unit tests or web performance tests or a combination of both types of tests. Next, using a simple wizard, you can construct complex load tests which have different network bandwidths, load patterns, test mixes and use a variety of different browsers. In addition, you can specify the warm up time, how long the test will run for, whether or not to use think times, and you can determine which machines you want to gather performance counters for. While the interface for selecting these options is simple, taken together they can be used to specify virtually any performance profile you want. Want some sample users on the standard 10MB/sec network bandwidth while others are using dial-up? How about gathering detailed performance information not only of the middle-tier system being tested but of the data tier, proxy server, network load balancer and various other machines along the communication path? You can set these up through the wizard as well. Figure 2 shows the Load Test Wizard

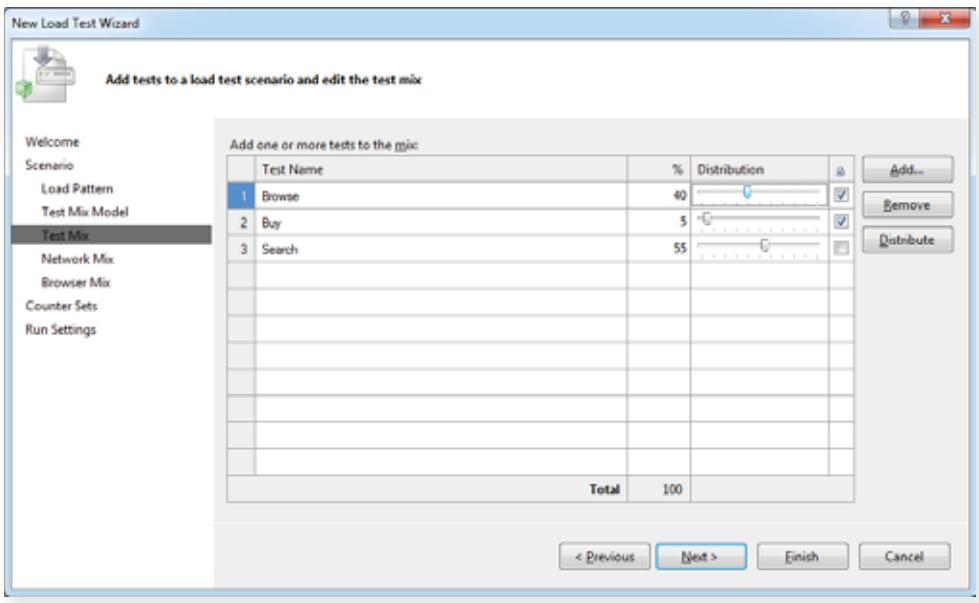


Figure 2 - Set the text mix to predicted usage.

You can easily collect performance counters from the system under test using counter sets, which come with pre-configured thresholds to warn when resources are over-utilized (Figure 3).
You can also extend the testing framework by creating custom data collectors that gather

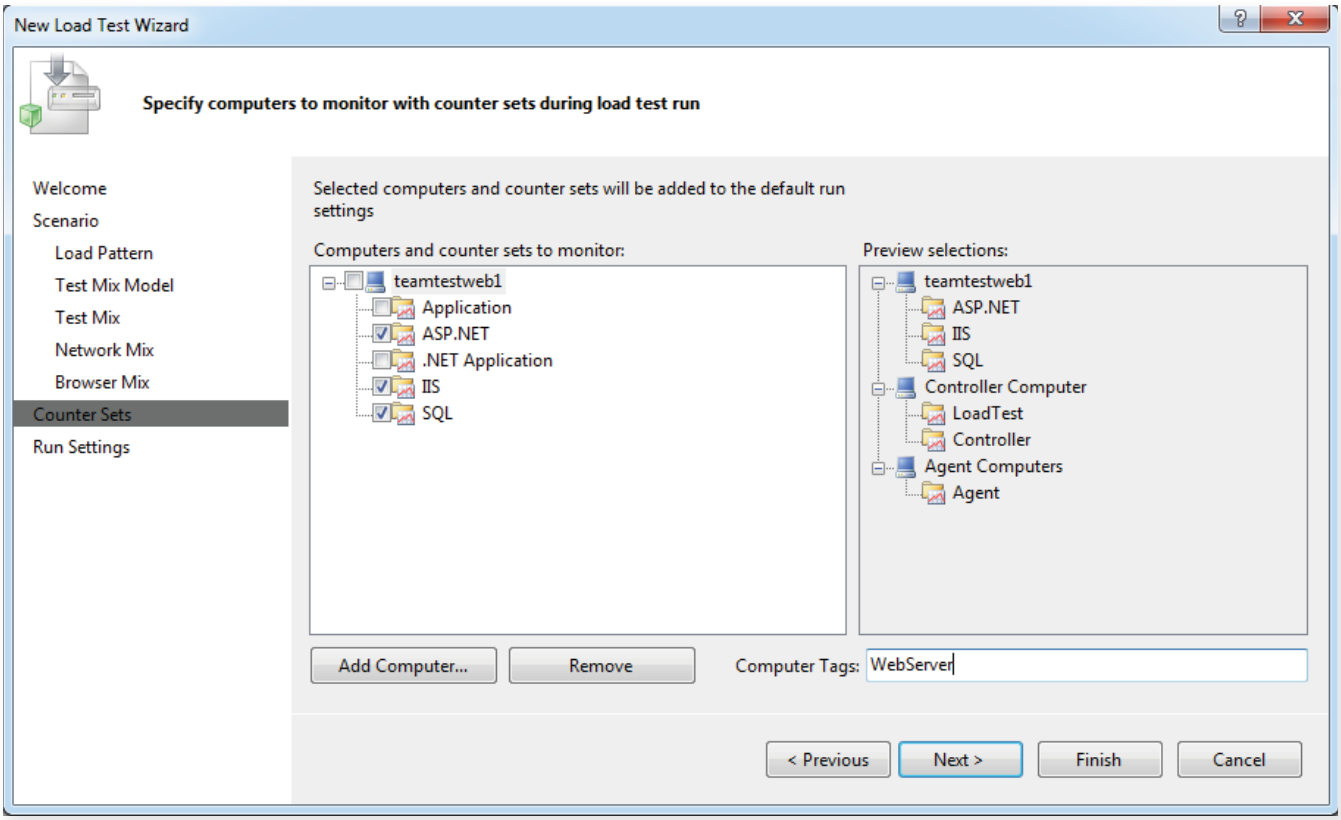


Figure 3 – Select Performance Counter Sets.

information specific to your needs. Using these capabilities, you can capture custom application logs, network usage, SQL Server calls or virtually any other needed information.

In addition to all of the options open to you – far too many to list here – it scales, scales, and scales. The load testing controllers and agents scale to support true enterprise scenarios with tens of thousands of users at a lower cost than previously available. The best news here is that the licensing is simple – you add additional users through the purchase of user packs. There are no different types of users, additional costs for more controllers, or any hidden surprises. This gives you flexibility and predictability in the cost of your performance testing.

Analyzing test results

Load testing is great but if you can't perform an analysis of the test results, the tests themselves are not very useful. The Visual Studio Load Test Analyzer graph view lets you correlate performance slow-downs with activity and conditions on the server, such as errors reported in the event log or excessive

resource utilization (Figure 4). Performance Counter thresholds are automatically configured for you in the load test, and allow you to quickly identify resources on the server that are under pressure.

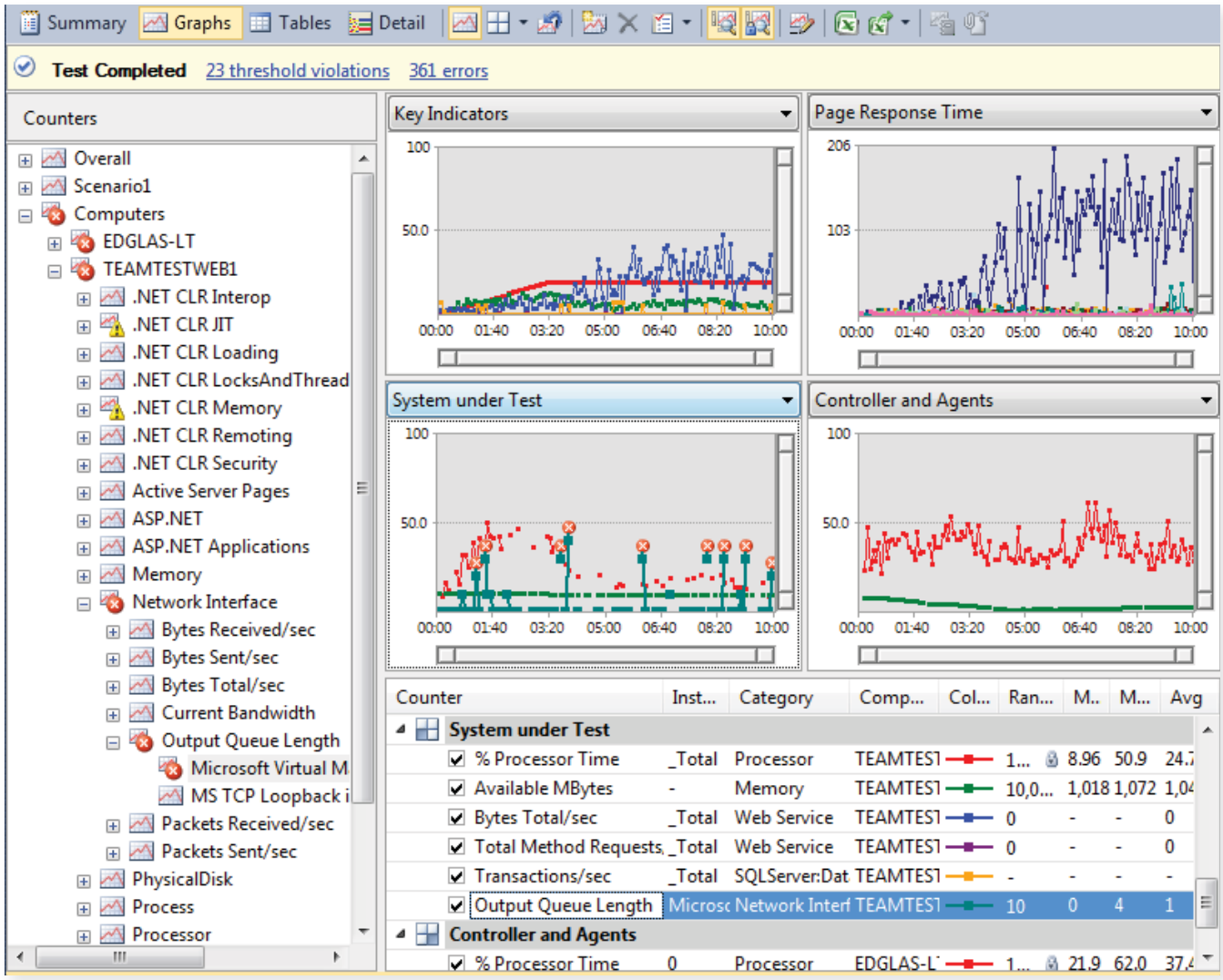


Figure 4 - Load Test Result Summary.

Using the Virtual User Activity Chart, you can determine what individual virtual users were doing during a performance slowdown (Figure 5).

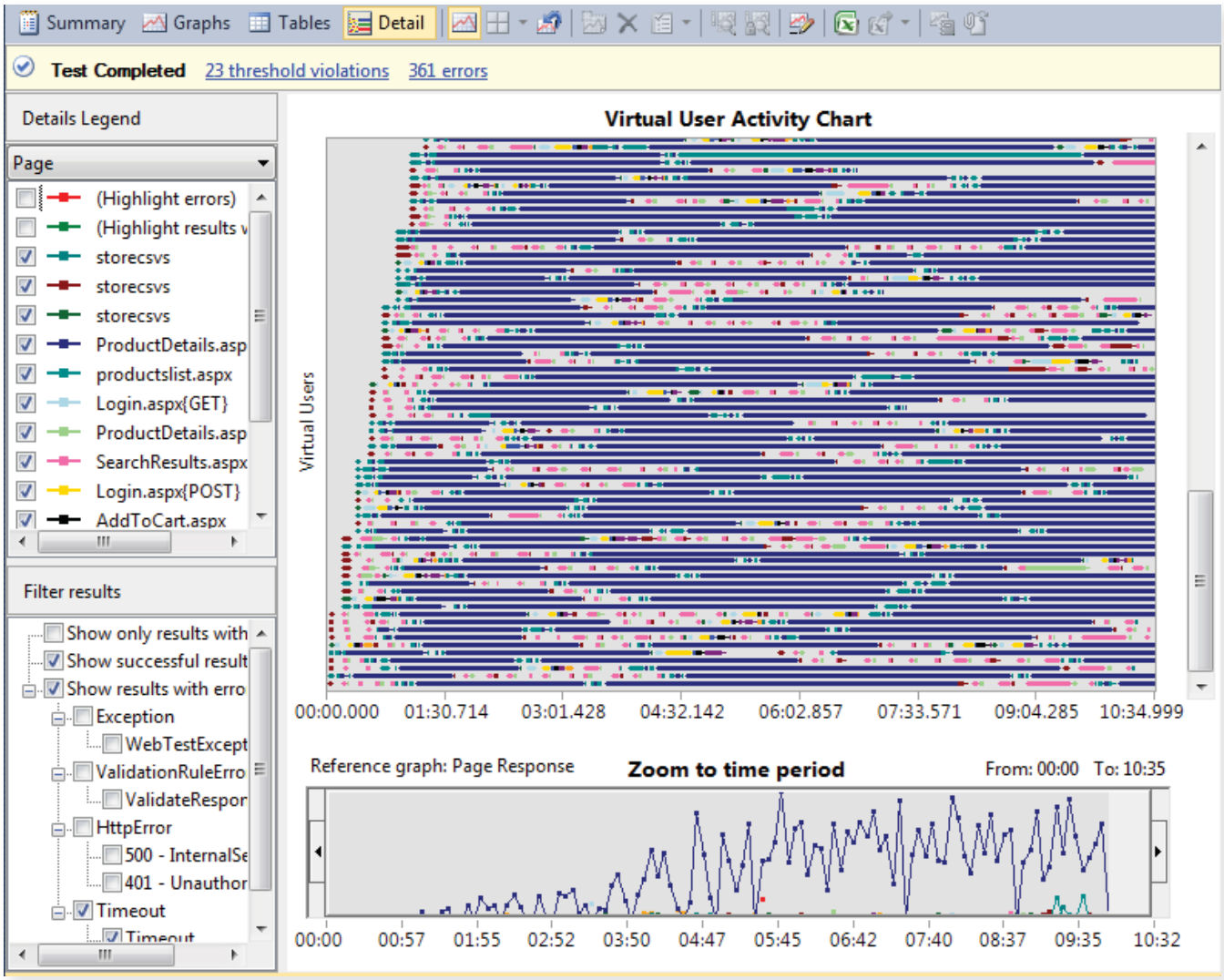


Figure 5 – Virtual User Activity Chart.

Fixing the Bottlenecks

Once you have identified a problem you need to be able to fix it. For this, Visual Studio 2010 uses the Microsoft ASP.NET Profiler data collector. The data collector allows you to sample or instrument your code and even the code in the .NET Framework to analyze it for bottlenecks down to the method level. A typical scenario might involve running a Load test and discovering that one of your scenarios, Order

What is the ASP.NET Profiler? This data collector profiles the Internet Information Services (IIS) process. It will monitor the overall performance of IIS relative to a specific application, record performance counters and monitor your code down to the method level. You can also gather information on calls made to a database to determine the effect of those calls on your overall application performance.

Item, is running slower than it should and you want to discover why this is the case. Once you've identified the problem scenario you can drop down into the ASP.NET profiler to get detailed information on every method called during a session (Figure 6). This lets you identify the hot path which is the slowest path through your code.

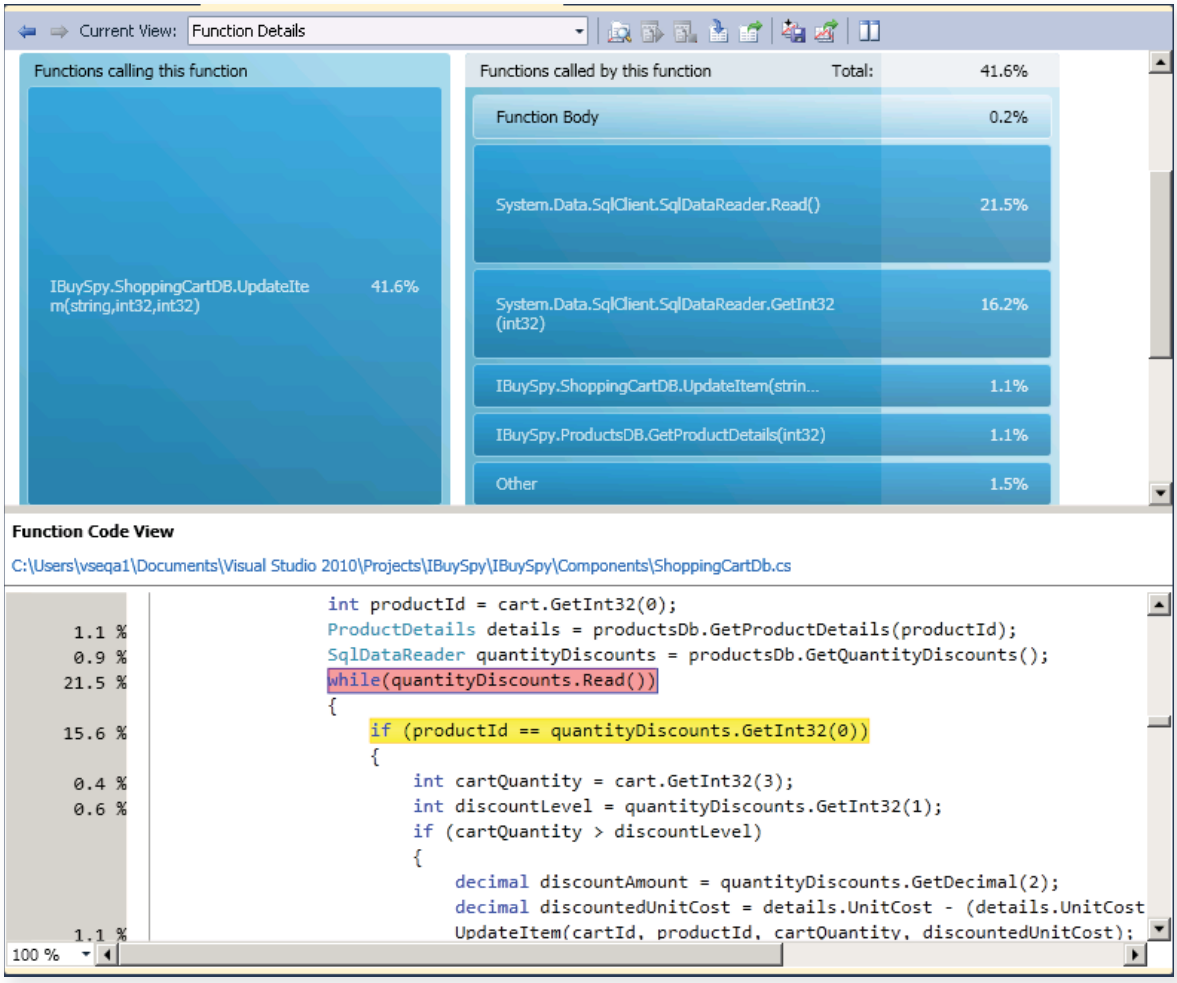


Figure 6 - Function Details view

Find Performance Problems for Global Customers Before Deployment

The ASP.NET Profiler works well with application performance issues. Have you ever rolled out an application that worked great for local customers, but performed poorly for global customers accessing it over a WAN? Many performance issues are due to chatty clients or clients that send and receive large amounts of data. These applications work fine on a LAN, but have poor performance over a WAN. VS 2010 enables you to run applications while simulating a WAN, enabling you to find these types of performance problems early.

True Network Emulation Microsoft Visual Studio 2010 uses software-based true network emulation for all test types. This emulation simulates network conditions by direct manipulation of the network packets. This allows for easy simulation of wired or wireless networks and allows for filtering at the packet level.

Easily Create Large Data Sets Using Data Generation

An area that often goes overlooked is the amount of data in the database. The more data there is, the more likely it is for queries to execute slowly and introduce deadlocks. One of the features of Visual Studio Premium and Ultimate is the ability to generate test data. Using this feature you can generate and test large amounts of realistic data with very little effort (Figure 7).

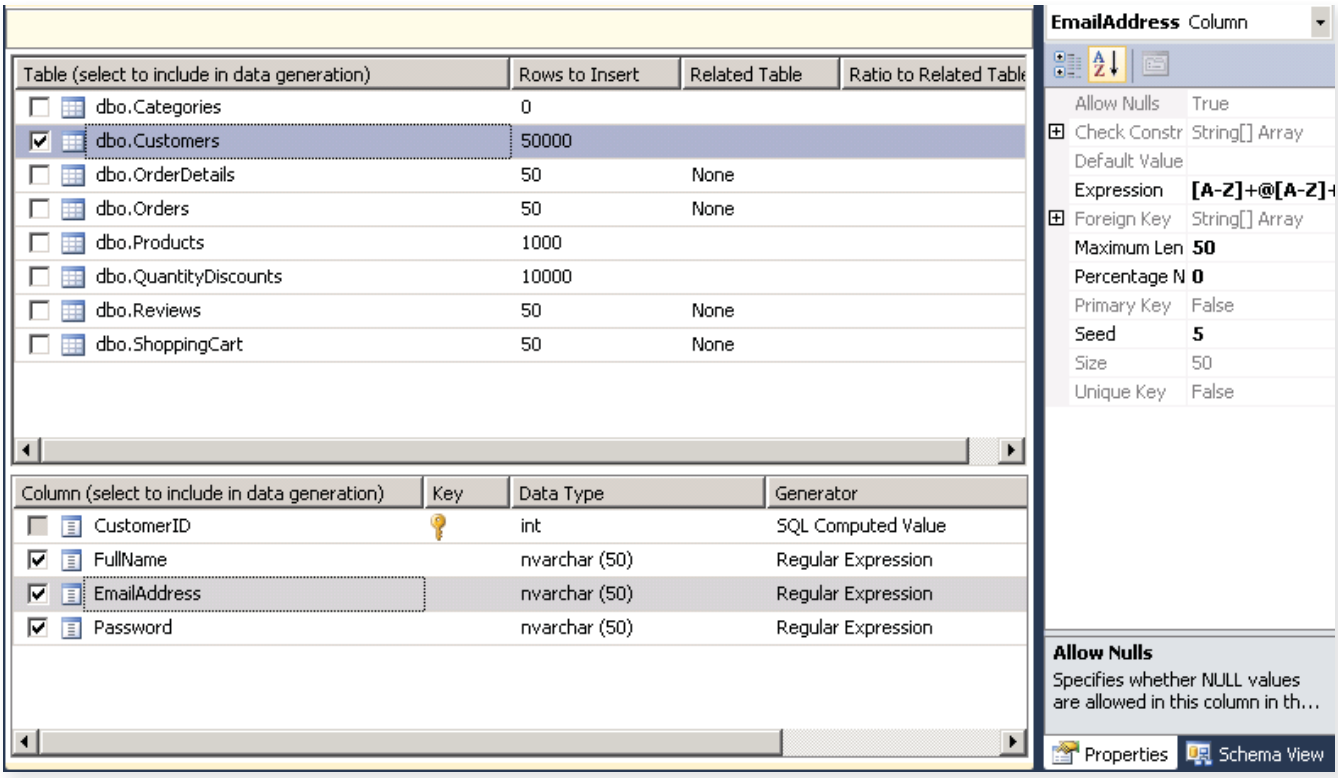
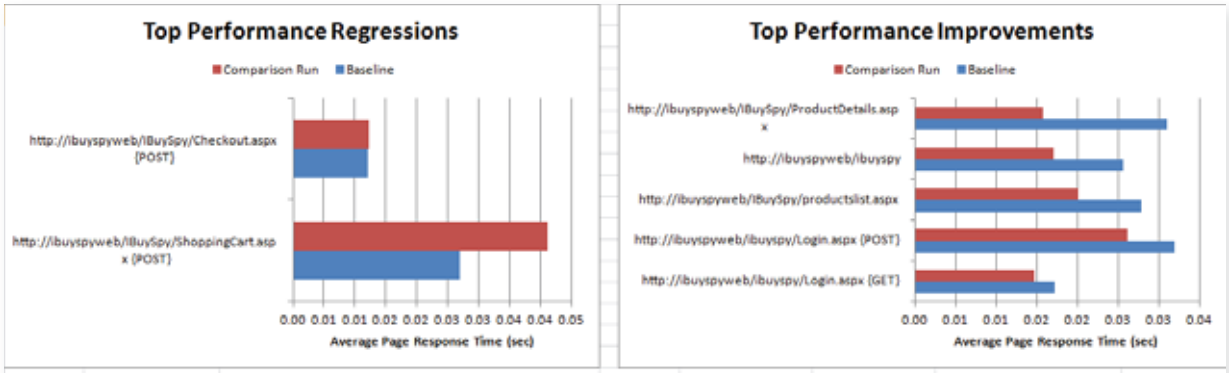


Figure 7 - Data Generation

Reporting

Visual Studio 2010 introduces a new set of reports which help you analyze test runs and share results with stakeholders. Using the analytical power of Microsoft Excel you can quickly and easily perform comparisons between test runs as shown in Figure 8. In this view you can see the page response time for a given operation – both where the application became slower and where it improved.

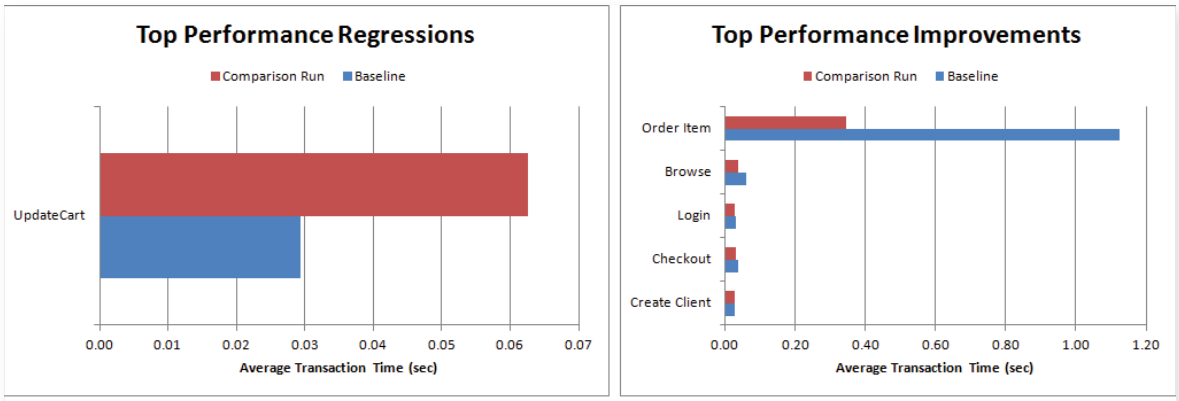


Scenario	Test Case	Request	Baseline	Comparison Run	Response Goal	% Change from Baseline	% from Goal
Customers	ShoppingCustomer	http://ibuyspyweb/IBuySpy/ShoppingCart.aspx (POST)	0.03	0.04	---	-53%	---
Customers	ShoppingCustomer	http://ibuyspyweb/IBuySpy/Checkout.aspx (POST)	0.01	0.01	---	-1%	---
Customers	BrowsingCustomer	http://ibuyspyweb/ibuyspy	0.02	0.02	---	0%	---
Customers	ShoppingCustomer	http://ibuyspyweb/IBuySpy/AddToCart.aspx	0.02	0.02	---	3%	---
Customers	BrowsingCustomer	http://ibuyspyweb/ibuyspy/ProductDetails.aspx	0.02	0.01	---	12%	---
Customers	BrowsingCustomer	http://ibuyspyweb/ibuyspy/productslist.aspx	0.01	0.01	---	12%	---
Customers	ShoppingCustomer	http://ibuyspyweb/ibuyspy/Login.aspx (GET)	0.02	0.01	---	15%	---
Customers	ShoppingCustomer	http://ibuyspyweb/ibuyspy/Login.aspx (POST)	0.03	0.03	---	18%	---
Customers	ShoppingCustomer	http://ibuyspyweb/IBuySpy/productslist.aspx	0.03	0.02	---	28%	---
Customers	ShoppingCustomer	http://ibuyspyweb/ibuyspy	0.03	0.02	---	33%	---
Customers	ShoppingCustomer	http://ibuyspyweb/IBuySpy/ProductDetails.aspx	0.03	0.02	---	49%	---

Figure 8 - Page run comparison

Figure 9 shows transaction comparison – that is, how long did it take to perform a set of steps which make up a user scenario? In this figure some of the scenarios are logging onto the system, ordering an item, and checking out.

Performance counter time, errors, overall test results and other reports are also standard reports available and they can be filtered to provide the level of detail that you need. Once the performance bottlenecks have been identified you can drill into the code and make changes as needed to improve performance.



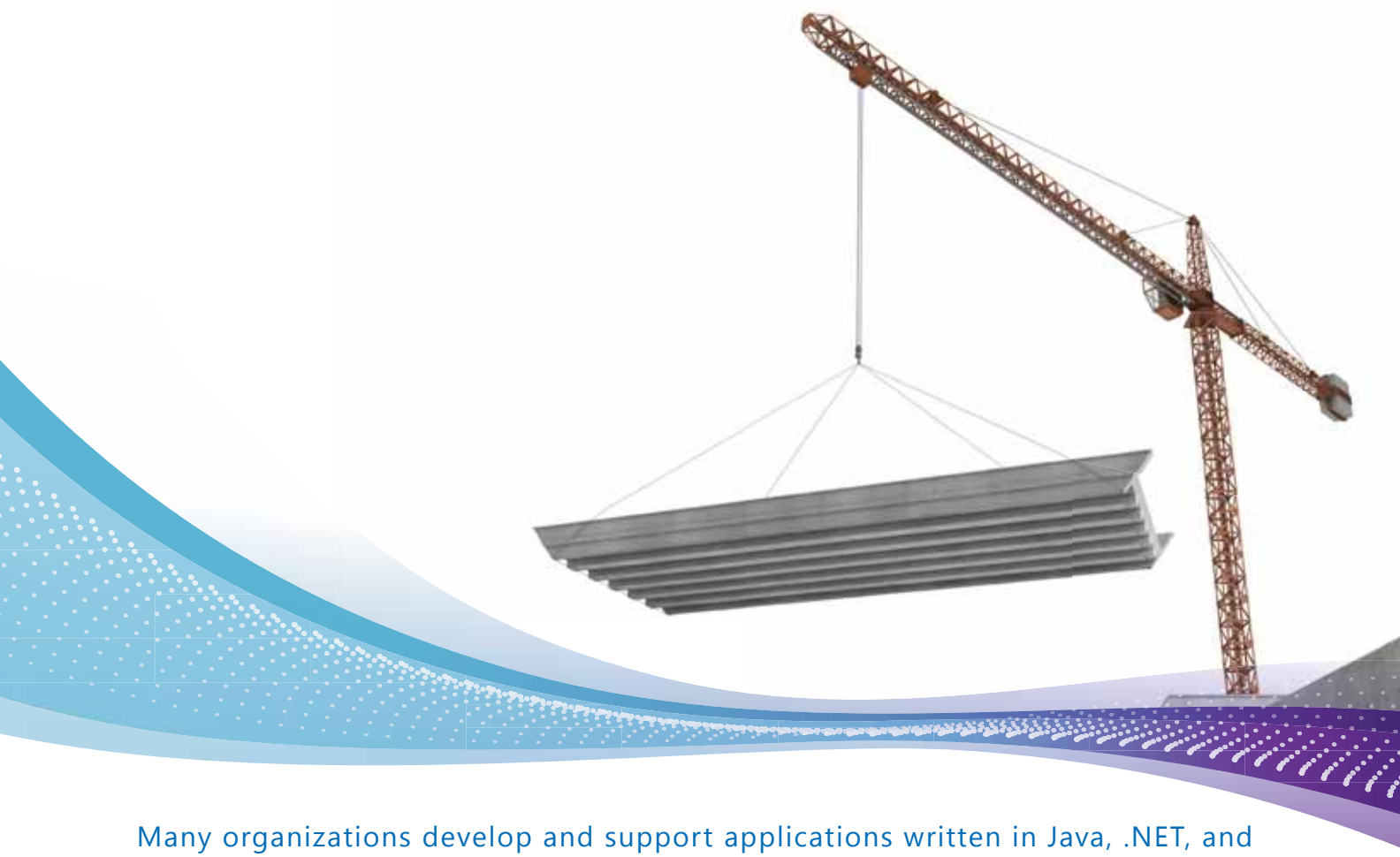
Scenario	Test Case	Transaction	Baseline	Comparison Run	% Change from Baseline
Customers	ShoppingCustomer	UpdateCart	0.03	0.06	-114%
Customers	ShoppingCustomer	AddToCart	0.02	0.02	4%
Services	TestInstantOrderService	Create Client	0.03	0.03	7%
Customers	ShoppingCustomer	Checkout	0.04	0.03	14%
Customers	ShoppingCustomer	Login	0.03	0.03	18%
Customers	ShoppingCustomer	Browse	0.06	0.04	40%
Services	TestInstantOrderService	Order Item	1.12	0.34	69%

Figure 9 Transation Times

PERFORMANCE TESTING FOR EVERYONE

The Visual Studio performance testing tools provide a mature, stable, extensible platform for executing, analyzing and acting on test results. Because of the tight integration with the development environment, performance test results are actionable – they don't just make a good looking report. Teams can analyze the data, drill down to the code causing the bottleneck, fix it, and re-run the test to examine the effects of the change. Whether the application has a small number of users or a large number, is mission critical or not, these tools can help you improve the performance of your application without resorting to costly tooling and a steep learning curve.

Heterogeneous Development with Visual Studio 2010



Many organizations develop and support applications written in Java, .NET, and often other platforms. Today, these disparate platforms have resulted in separate silos, leaving the organization without end-to-end visibility into its development efforts. This separation also means that any process improvements made by one silo are unlikely to benefit the other silos. With Visual Studio 2010, that painful separation is about to become a thing of the past.

A HISTORY OF SEPARATION

For the longest time, working with Microsoft development tooling meant you were working on Microsoft Windows and probably on code dedicated to those platforms. Organizations picked tools based on the technologies involved, because tools and technologies went together. If you were writing Visual C++, for example, you were in a Windows-based tool.

If you were using Java, perhaps yours was an integrated development environment (IDE) from Sun, IBM, or the open source world. You might have been on a Windows-based development machine, but you were often deploying onto some Unix/Linux variation. But choosing a single additional platform usually also suggested a completely different technology stack. It was fertile ground for confusion - and dogma as well; finding economies of scale in such a scattered landscape was seemingly forever elusive.

Sometimes it was little more than professional preference, but organizations often divided teams by these tool choices. With legacy systems and merger and acquisition activities, firms simply had to accommodate many platforms, and often also managed redundant process, teams, and development tooling. It became implicit that tools, people, and languages were largely matched sets that rarely connected with outsiders. You picked a side and it often defined your career.

That was the natural order of things. Big consulting firms, large IT teams, and product developers in the interoperability world eventually required a broad, typically disjointed tooling solution. They had special people and expert teams for the many and varied platforms. The silos became entrenched and most grew accustomed to the separation; people learned how to co-exist, but rarely banded together.

The process tooling was typically siloed as well. When there was common process, it was often disjointed from the coding work itself; companies had separate process personnel and product teams for the different technology stacks. Building and deploying for the mainframe and other assorted systems for internal IT and outfacing data centers bred a range of specialists and cultural walls. So the opportunity to align together on process and tooling went largely unrealized. Tools for construction and tools for other development areas were point solutions; the holistic visual flow of work, and the sharing of minds and goals, went unrealized as the world did not readily support unity.

NEW PARADIGMS

The release of Team Foundation Server 2005 brought some interesting concepts to the table. Combining the process tooling and the IDE and sharing a common view of activities across the project team was a paradigm shift. It was new and different to have all team practitioners on the same holistic page, describing a single shared process with lots of data. Solid Application Lifecycle Management [ALM] with tight integration to the technical teams had emerged.

Now Visual Studio 2010 breaks even more new ground. The concepts that process and tooling must be duplicated for different deployed technologies and that silos are normal are now being called into question. The Team Foundation Server Plug-in for Eclipse (see Figure 1) now brings long-isolated practitioners together, regardless of technology, under one unifying process.

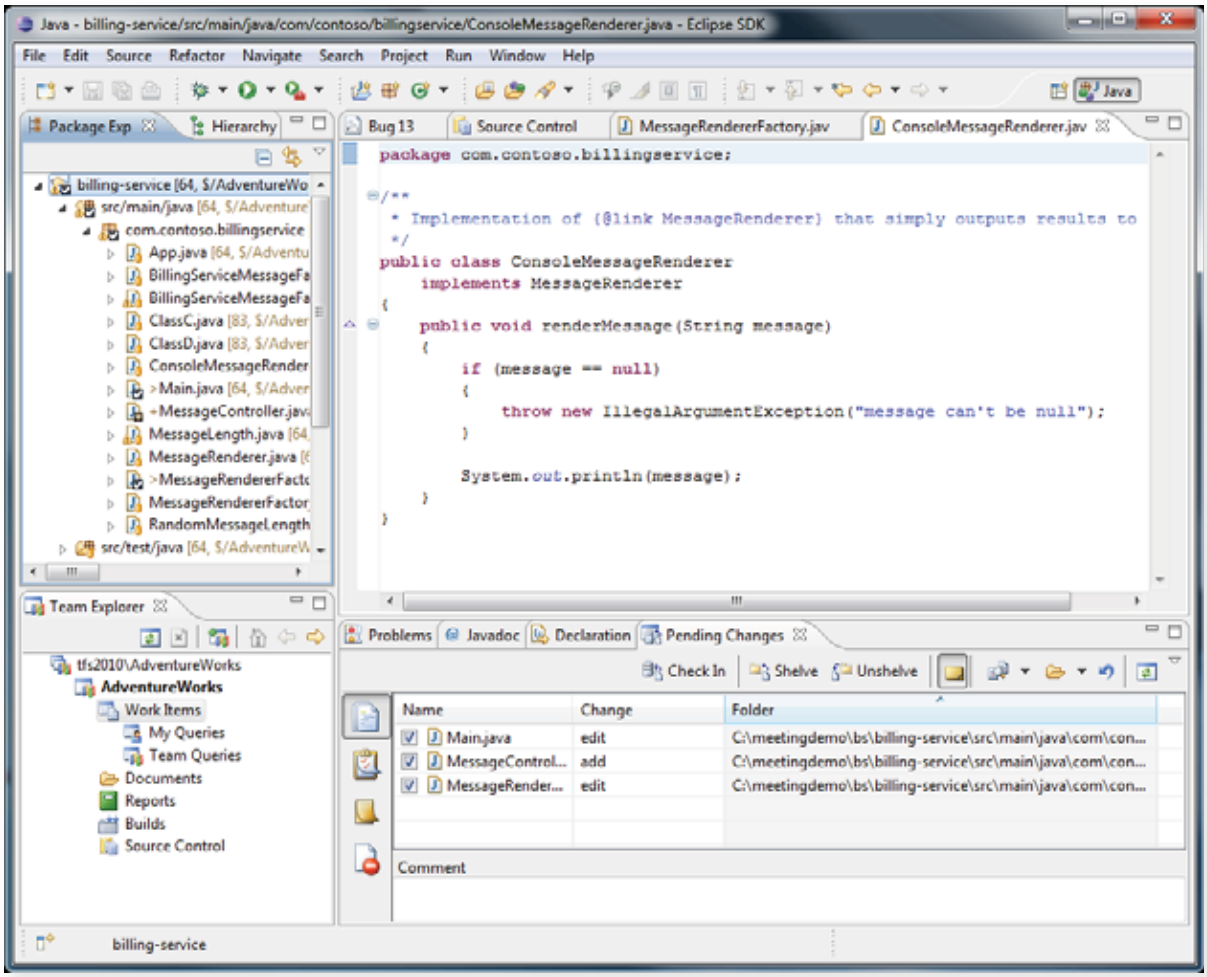


Figure 1 - Editing code in Eclipse using the Team Foundation Server plug-in.

The Team Foundation Server plug-in for Eclipse brings the bulk of the editing and ALM functionality of Team Explorer to the Eclipse IDE. Eclipse has been a common IDE in the Java space for some time; there are tools and extensions for a range of technologies. Developing in other languages (e.g. Java/J2EE, C++, Python, Perl) using Eclipse is commonplace.

Development with Eclipse can now mean development with the process management of Visual Studio Team Foundation Server 2010. Beyond source control in their familiar IDE, Eclipse users now have access to the same work items and process templates, the same project portal, documents, and reports, and the same project collections as the .NET developers but in their existing chosen development environment! If you have code or web infrastructure sharing .NET, PHP, Java/JSP, Python, Perl, Django, or other technologies, this all can be managed and maintained with a single ALM toolset. Developers can continue to use their familiar tools, and at the same time can connect and collaborate through the common Visual Studio 2010 ALM infrastructure. Teams working on different technologies can now act with one voice, supporting common vision and goals, and collaborating in a shared tool.

If you are currently working primarily in Eclipse and are new to Visual Studio, the strong connection between process tooling and code development may be unfamiliar. Depending on your tools, you might have struggled to get decent metrics or a shared collaboration environment. Almost certainly, there was an assortment of tools and a lot of difficulty in seeing the big picture. Mapping change sets in code to features or defects in your tracking and requirements tooling probably meant expensive licensing. With Team Foundation Server 2010, however, powerful collaboration tools can be accessed directly from Eclipse based IDEs.

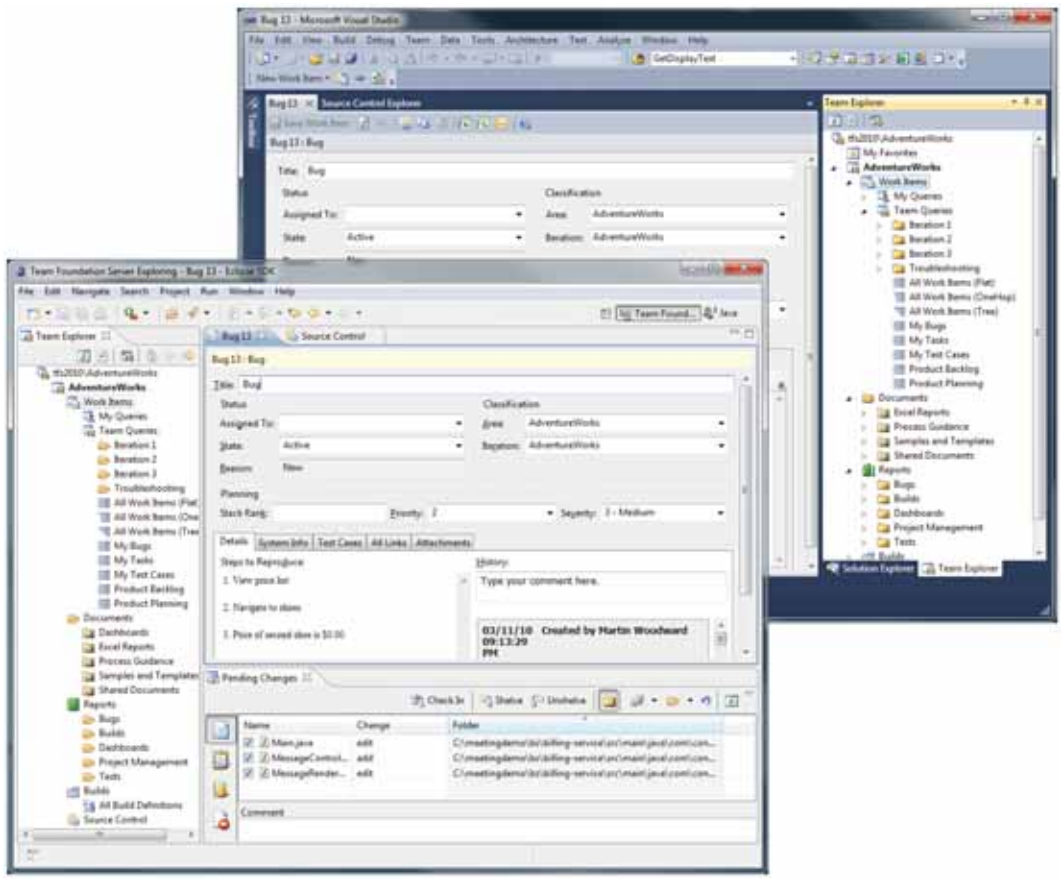


Figure 2 - The Team Foundation Server Plug-in for Eclipse provides the familiar elements for working with resources.

APPLICATION LIFECYCLE MANAGEMENT FOR EVERYONE, EVERYWHERE!

Because Visual Studio Team Foundation Server 2010 is at the heart of the Visual Studio ALM offering, the ability to view project information and statistics is equally available on multiple platforms. True application lifecycle management is now available for the heterogeneous technology organization, with all the deep process visibility and understanding that comes with the data warehouse and SQL Server Analysis Services. It's ALM and shared tooling for the rest of us!

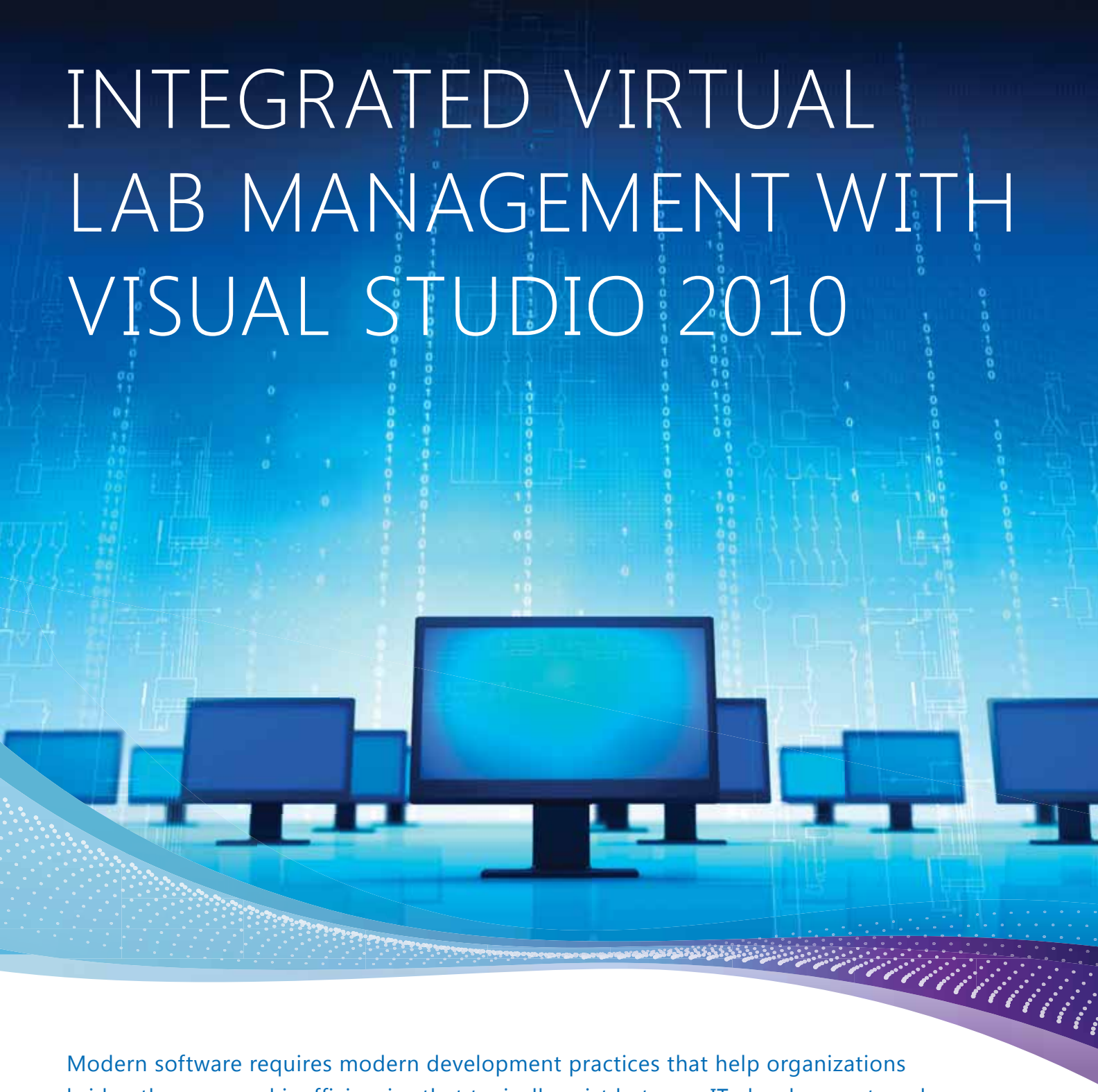
For those developers who actually live in the world of mixed development, the number of tools just went down. Those who regularly work on multiple-language projects can now benefit from a true shared code base, a fundamental principle for those doing agile development. And with the strong collaboration tooling and easy metrics, you can have a solid understanding of your projects regardless of the product technology mix.

Additionally, administrative and process overhead is reduced. Basing IT and product development activity on a single integrated toolset across the organization, instead of across multiple process tools and technologies according to different teams, means no more double duty on bringing process gains to different areas; it also means a reduced skills requirement for process maintenance. The maintainers can focus on one process infrastructure and bring process improvement for the larger organization.

With the Team Foundation Server Plug-in for Eclipse, heterogeneous development is no longer an "us versus them" proposition. Team collaboration is facilitated and development silos can become a thing of the past. Common process and reporting, a single toolset and code base, now make new efficiency and effectiveness possible for the previously isolated and adversarial practitioners. Since your business requirements can now travel easily across the old technology borders, you receive a much higher level of visibility than ever before.

Transparency and collaboration are not just buzzwords with Visual Studio 2010; Microsoft is delivering new ways to work across old boundaries – and new levels of productivity.

INTEGRATED VIRTUAL LAB MANAGEMENT WITH VISUAL STUDIO 2010



Modern software requires modern development practices that help organizations bridge the gaps and inefficiencies that typically exist between IT, development, and quality assurance (QA). Visual Studio 2010 delivers the tools and technology that you need to easily and effectively develop and test your software in environments that mimic production through Lab Management. It helps you create and manage these environments effortlessly with self-service access tools and quick, automated deployment via build workflows that support regression testing as well as interactive use.

Building high quality software demands a lot from your organization and development teams. You're building a range of software solutions from simple client applications to complex deployments with multiple tiers supporting different operating systems and supporting services like databases and web servers. Building these solutions is hard enough. Once you start looking at quality assurance and regression testing, it can sometimes seem overwhelming.

TODAY

If you are building high-quality software today, ideally, you have an effective software development life cycle that includes some type of formal QA process. A typical QA process is made up of people, process, and tools. A great tool to have is a test lab. Test labs today tend to be one of two types.

The first type is the legacy lab where real hardware is often allocated from second-hand development boxes. A team of engineers manages imaging and deployment, which can result in a lack of agility in the overall quality cycle from new build to test pass. Plus, all of these engineers need to be paid and managed. In an era of moving toward Green IT, nothing says "Brown IT" like a large lab of inefficient machines creating their own climate zone.

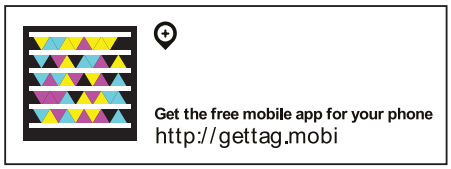
The second type of test lab is the modern one that uses virtualization technology. While a huge step forward compared to the typical legacy lab, it's not without problems. Today's solutions tend to be very expensive when it comes to licensing the lab management software. Many of the products are also overly complicated and not designed to fit into the overall Software Development Lifecycle (SDLC). In fact, many are afterthoughts from vendors whose primary focus is operations, not development or QA. Integrating your development and QA practice with their workflow can be a difficult, frustrating experience.

Regardless of lab type used, there also exists a painful problem created by the typical bug-fixing cycle between developers and testers. With the legacy lab, testers have access to the hardware and resources for only a few specified cycles. This makes it hard, if not impossible, to get developers access to the lab when the bug is easily seen in the lab environment but not reproducible in their development environment. Even with testers filing high-quality bugs, typical test and production environments don't tend to match what developers have. In turn, this leads to a ping pong match between development and test teams about where the problem really lies.

CONTINUOUS DELIVERY THROUGH INTEGRATED LAB MANAGEMENT

With Microsoft Visual Studio's Lab Management solution, you will have a complete set of tools for managing and using your virtual lab. Available with either Visual Studio 2010 Ultimate with MSDN or Microsoft Visual Studio Test Professional 2010 with MSDN, Lab Management helps you eliminate the problems of most lab solutions to reduce developer/tester friction. Built on top of Windows Hyper-V and System Center Virtual Machine Manager 2008 R2, and integrated with Team Foundation Server 2010 (TFS), Lab Management provides a cost-effective, end-to-end solution for testers and developers to find and fix bugs quicker and ultimately help the organization take its quality solution to market faster.

REDUCING OPERATIONAL FRICTION THROUGH A SELF-SERVICING MODEL



Lab Management functionality is already available to you today as part of your MSDN subscription with either Visual Studio 2010 Ultimate or Visual Studio Test Professional 2010. This "per-user" licensing model makes it easy for anyone to adopt Lab Management despite the fact that its functionality is largely enabled by Team Foundation Server 2010. Microsoft Test Manager 2010[1] and System Center Virtual Machine Manager 2008 R2[2] will be your configuration and management tools enabling this built-in technology. Once your Team Foundation Server 2010 domain environment has been set up and configured, your team will benefit from a self-servicing model so that anyone on the team can access production-like virtual environments—on demand!

Because Lab Management is built on top of Hyper-V, you gain all the benefits of virtualization. You can use Hyper-V Server or enable the Hyper-V role on Windows Server 2008 or Windows Server 2008 R2. You gain the benefits of "Green IT", too: reduce the number of physical boxes that are drawing power and generating heat, cut hardware allocation costs and lessen the manpower needed to manage all those servers. Hyper-V supports a number of advanced features including clustering and live migration, so your lab has access to all these features too.

---

1 Available with Microsoft Visual Studio 2010 Ultimate or Microsoft Visual Studio Test Professional 2010.

2 Available with Microsoft Visual Studio 2010 Ultimate with MSDN or Microsoft Visual Studio Test Professional 2010 with MSDN.

Your primary entry point to Lab Management is through the Microsoft Test Manager (Test Manager), a rich Windows Presentation Foundation (WPF) application that makes it easy for users to access and use virtual machines. The Lab Center within Test Manager provides your team with easy access to all the features of the lab needed to quickly create and manage virtual machine environments (see Figure 1). In the Lab Center, you create, deploy, and clone environments. Each environment can have different test settings and be composed from existing virtual machines, templates—also known as "golden images"—or a composition of both types. In addition, you can create a physical environment of real machines. The Library feature provides the management tools to share and move resources where they need to be.

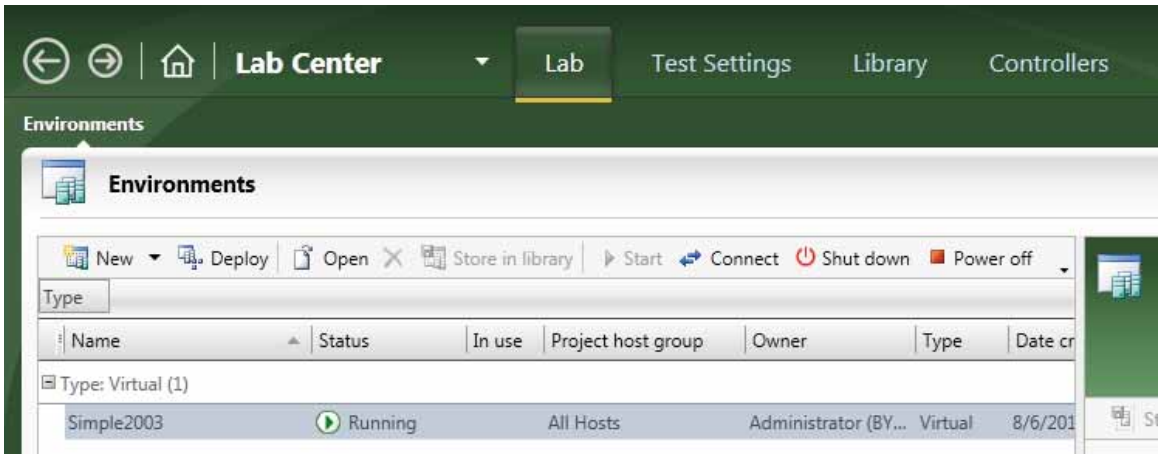


Figure 1 – The Lab Center in Microsoft Test Manager provides access to your virtual test lab.

Test Manager also provides access to the Microsoft Environment Viewer. This tool allows you to connect to all of the virtual machines in an environment as a holistic unit. In addition, you can manage environment snapshots (see Figure 2).

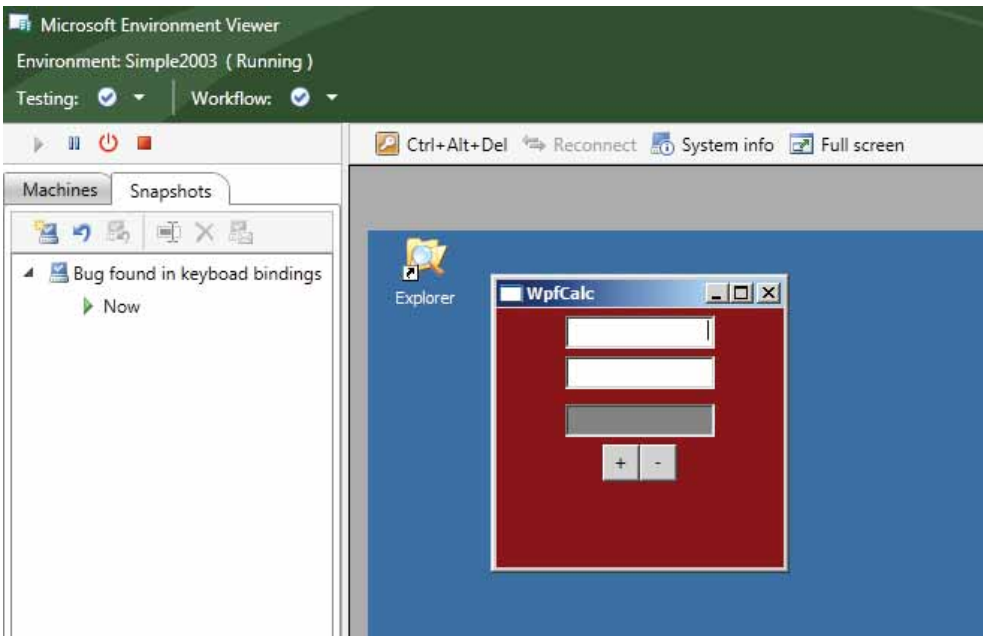


Figure 2 – The Microsoft Environment Viewer provides quick access to virtual machines and their snapshots.

Test Manager puts virtual machines at your fingertips. With full support for role-based administration, users can easily access existing environments for both development and testing. In addition, Test Manager provides simplified access to virtual machines and golden images. This enables team members to provision new environments themselves without having to burden IT or wait for someone from IT to get machines set up.

IMPROVING OPERATIONAL EFFICIENCY THROUGH AUTOMATED WORKFLOWS

Once environments are created, you can easily make them the foundation of a solid build-deploy-test (BDT) workflow. Using the build feature of Team Foundation Server, you simply define a build from Team Explorer and run a wizard to define the entire workflow process. This process can include specifying the environment to use as your install baseline, managing and applying snapshots (see Figure 3), determining which build to use, how to deploy the build, and deciding how to run automated tests.

Environment Snapshots takes the hassle out of testing multi-machine environments. With the click of a button, Lab Management captures a snapshot of all of the virtual machines that make up the environment at a consistent point in time. You can this use the snapshot to move forward and backward through the timeline, making it easy for developers and testers to always work with the virtual machines in a known state.

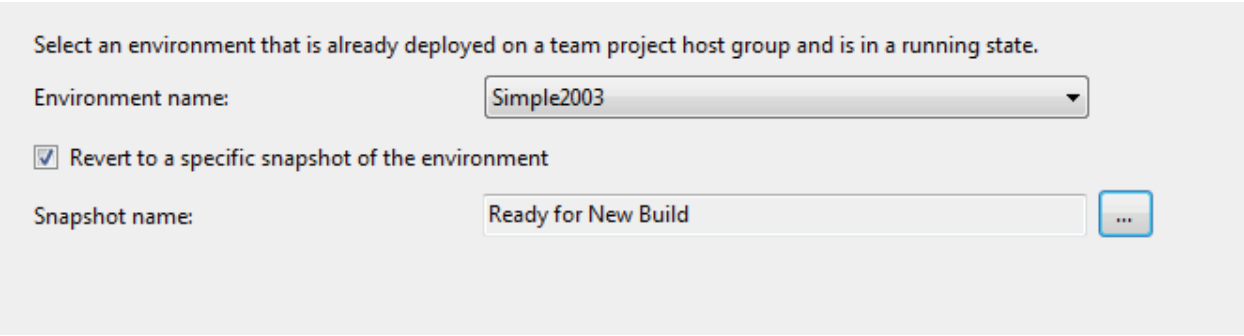


Figure 3 – Specify which snapshot should be used before your application is deployed.

The build workflow is super flexible and supports extensiblity via Windows Workflow 4 activities. It provides you with complete control of how your environments are used. In particular, it makes it easy to deploy to a known configuration (see Figure 4). Through snapshots, you can specify

that the environment should be set to a specific state before deployment. Then, after your application has been deployed, you can have the workflow take another snapshot. This is a great time saver for testers. They can easily start up the environment, run some tests, and reset back to the environment knowing the application is installed and is clean for another round of testing.

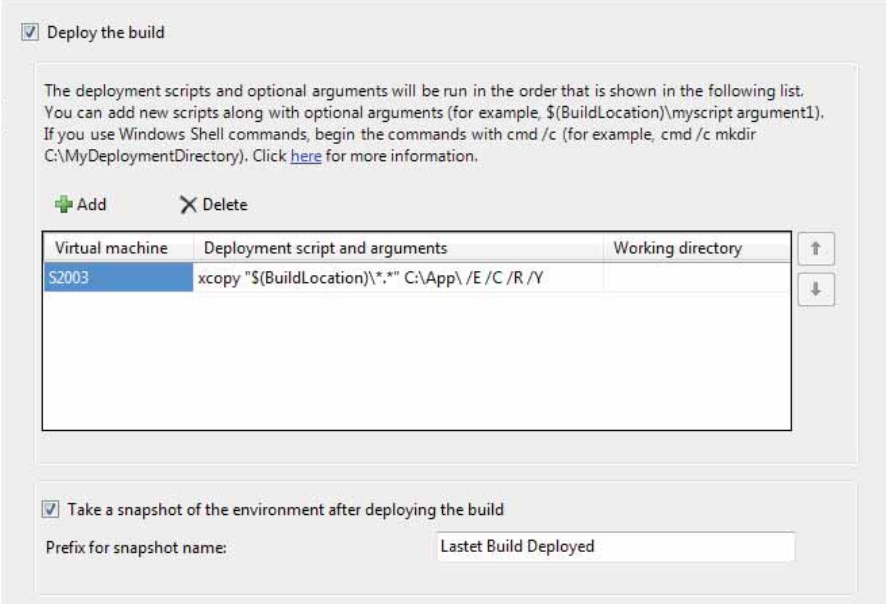


Figure 4 – Snapshot your environment after your build is deployed.

As mentioned, during the build workflow, automated tests can be run in the environment. The Test Agents[1] that run the tests and workflow support gathering rich diagnostic data. An environment can use a common set of settings that defines which diagnostics to gather or use its own custom set (see Figure 5). These Diagnostic Data Adapters support a number of options to make it easy for you to tailor your collection strategy for each build and environment.

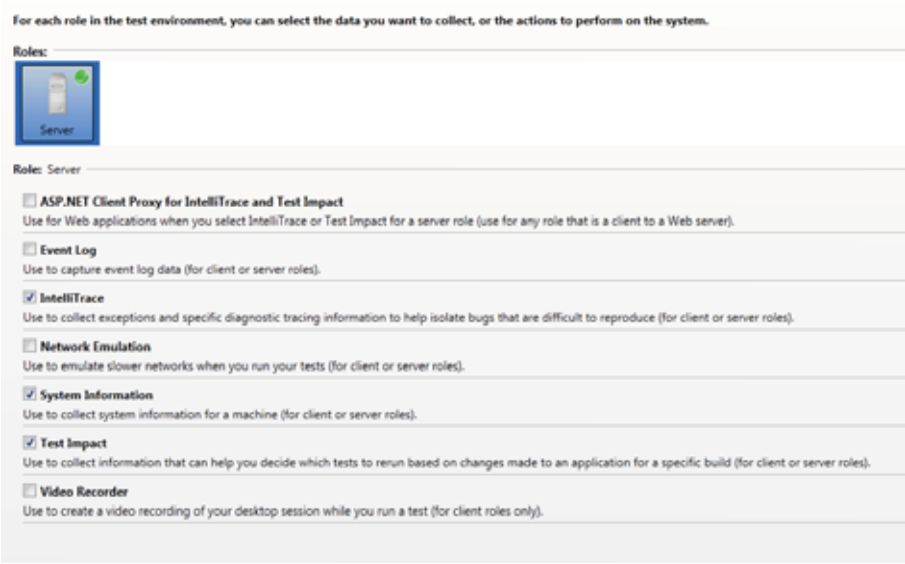
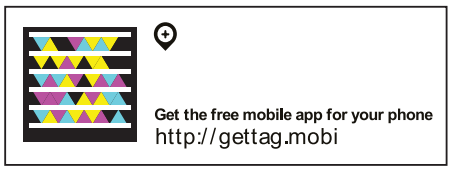


Figure 5 – Specify what adapters should be enabled in an environment.

1 Available with Visual Studio Agents 2010, additional software for Visual Studio 2010 and freely available on Microsoft Downloads.

ENABLING INTERACTIVE USE THROUGH RE-USE AND SHARING



Diagnostic Data Adapters (DDAs) are plug-ins used by Test Agents to gather data during the execution of tests, whether manual or automated. You can specify different combinations of DDAs into logical Test Settings groups. In addition to using the Microsoft-provided DDAs, you can write your own for specialized data collection.

Lab Management's use of virtualization technology accelerates the set-up, tear-down and restoration of complex virtual environments to a clean state providing continuous delivery to your teams. And, if an environment turns out to be useful or in high demand, you can clone it and make it available to other team members. This is even possible with complex, multi-machine environments that contain domain controllers, database servers and more. Lab Management makes this possible via a feature known as Network Isolation (see Figure 6). This powerful feature allows multiple members of your team to spin up the same environment and use it without causing computer name or DNS conflicts on your network.

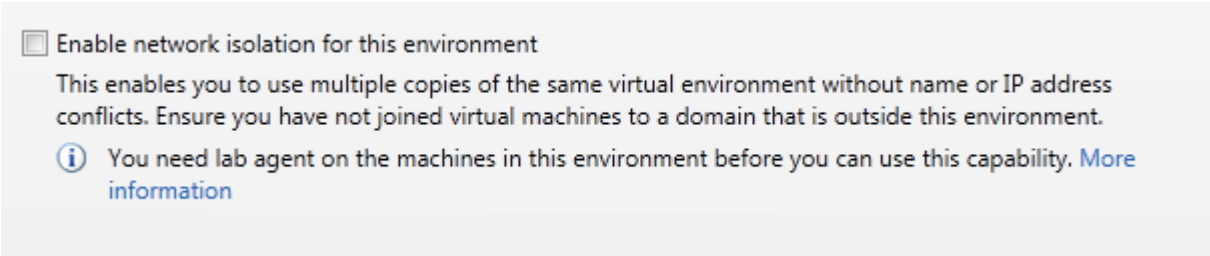


Figure 6 – Enable network isolation for greater environment reuse and sharing.

A great thing about the diagnostic capabilities available during builds and automated tests is that they are also available to testers when executing their manual tests. The automatically collected data can be added to any bug filed, which makes the bug actionable for the developer. And when using Lab Management Environment Viewer, you have the powerful option to attach an environment snapshot link to the bug (see Figure 7). This lets the developer rehydrate the test environment to the exact point in time and condition where the bug was found. This helps solve the no-repro problem by allowing testers to provide bugs that contain links to the snapshots used

by developers to recreate bugs. With a single click of a button, a developer can launch a virtual environment that exactly matches the one in which the bug was found.

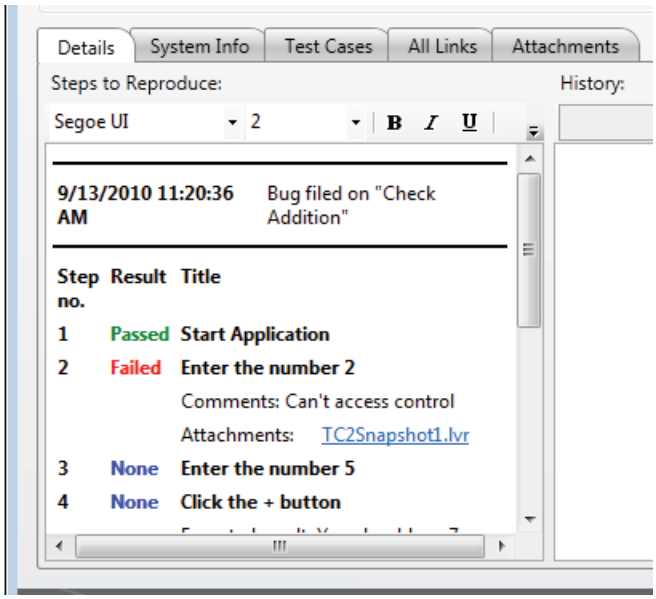


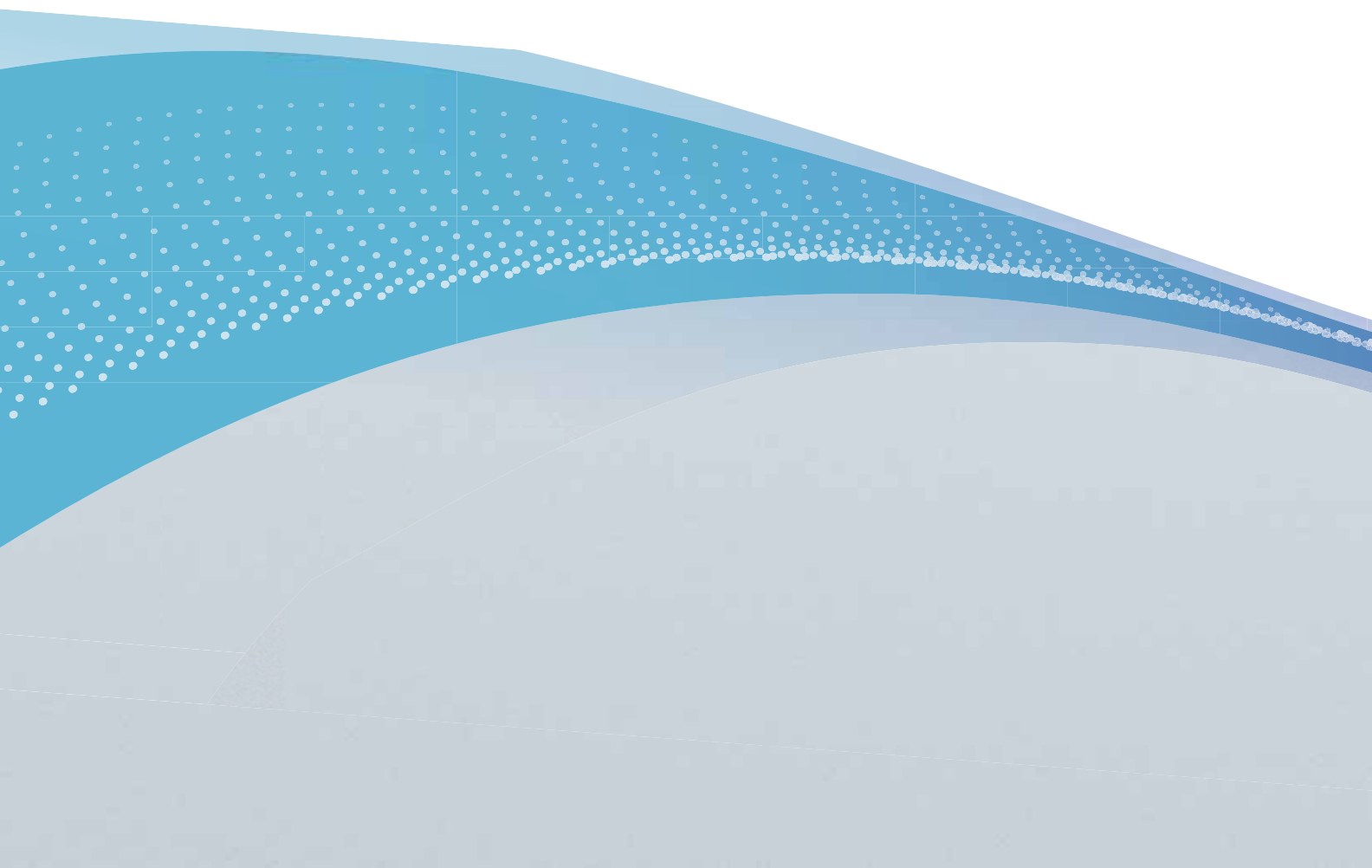
Figure 7 – Attaching a snapshot link to a new bug.

SUMMARY

Microsoft Visual Studio 2010's Lab Management solution gives you the tools you need to easily create, use, and manage multiple Hyper-V virtual machine environments for development and quality assurance. Using Microsoft Test Manager, everyone on the team has an easy-to-use tool for self-service access to the environments they need. Its wizard-based tools make provisioning and cloning of environments fast and simple. Rich integration with Team Foundation Server and build means your environments can stay up-to-date with the latest builds. In addition, the workflow integration with build means that the latest software gets deployed with full support for automated regression testing and validation. Finally, shared access to the same environments means developers load up the exact environment the tester used to find the bug so it can quickly be evaluated and fixed. Visual Studio Lab Management enables the entire team to raise the quality bar for all of your applications.

Understanding Application Lifecycle Management

Read what David Chappell has to say about why ALM is important to your business. In this section you will find 4 short papers that cover why ALM is at the heart of your success and how todays tools are now ready to take on all of your development challenges.

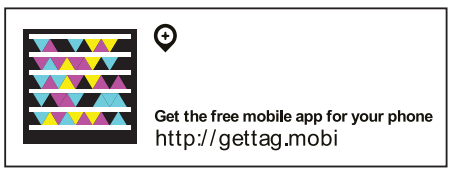



SET OBJECTIVES
IDENTIFY RISKS
DEFINE SUCCESS CRITERIA
ASSESS RISKS
PRODUCE ACTION PLAN
IMPLEMENT ACTION PLAN
MONITOR AND REVIEW

WHAT IS APPLICATION LIFECYCLE MANAGEMENT?

DAVID CHAPPELL

SPONSORED BY MICROSOFT CORPORATION



Defining application lifecycle management (ALM) isn't easy. Different people (and different vendors) take quite different perspectives. Still, ALM is important, and so understanding what it encompasses is also important.

It's common to equate ALM with the software development lifecycle (SDLC). Yet this simple approach is too limiting; ALM is much more than just SDLC. In fact, an application's lifecycle includes the entire time during which an organization is spending money on this asset, from the initial idea to the end of the application's life. To be both accurate and useful, our view of application lifecycle management should take an equally broad perspective. Anything else just isn't right.

THE THREE ASPECTS OF ALM

ALM can be divided into three distinct areas: governance, development, and operations. Figure 1 illustrates this, showing each of these three aspects on its own horizontal line.

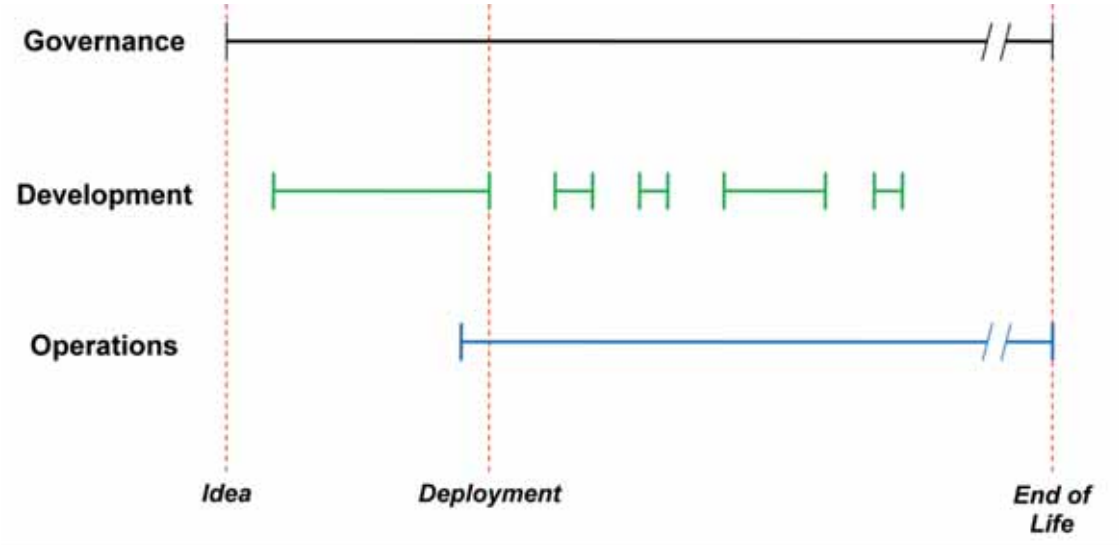


Figure 1: ALM can be viewed as having three aspects.

Like a human life, an application's lifecycle is demarcated by significant events. It begins with an *idea*: Why don't we build something that does this? Once the application is created, the next big event is *deployment*, when the application goes into production. And finally, when it no longer has business value, the application reaches *end of life* and is removed from service.

Like a human life, an application's lifecycle is demarcated by significant events.

Governance, which encompasses all of the decision making and project management for an application, extends over this entire time. Development, the process of actually creating an application, happens first between idea and deployment. For many applications, the development process reappears again several more times in the application's lifetime, both for upgrades and for wholly new versions. Operations, the work required to run and manage an application, typically begins shortly before deployment, then runs continuously

until the application is removed from service. Each of these three areas is important, and so each is worth examining in more detail.

ASPECTS OF ALM: GOVERNANCE

In ALM, the purpose of governance is to make sure the application always provides what the business needs. Figure 2 gives a close-up view of ALM governance, providing a bit more detail on what it entails.

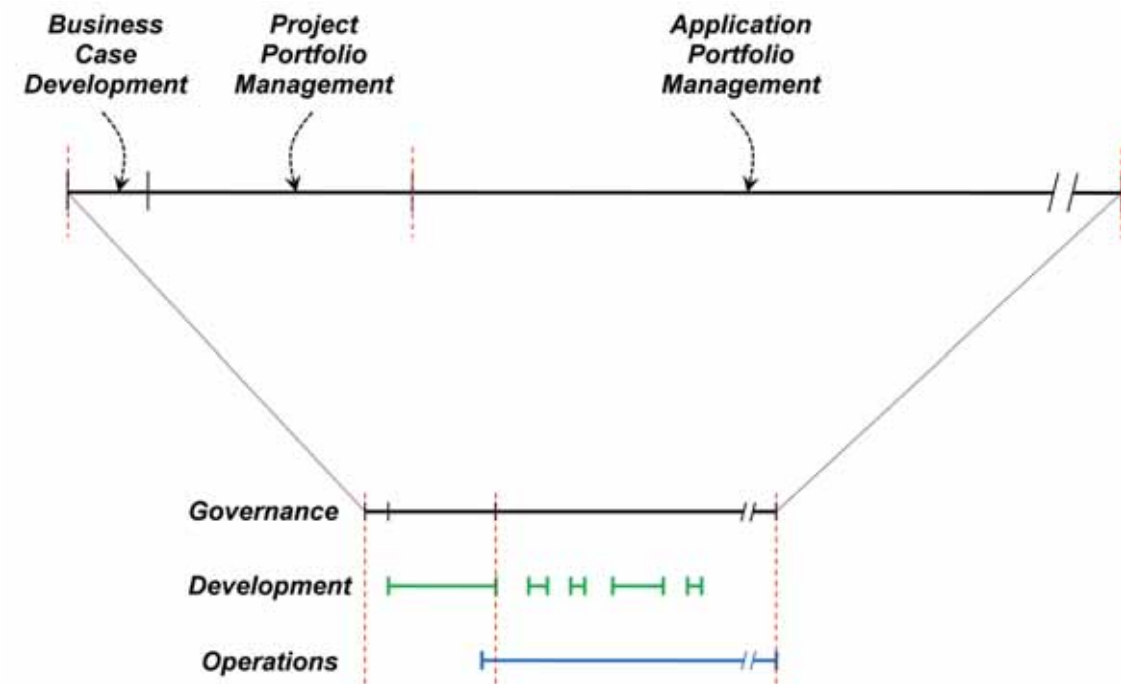


Figure 2: Governance extends over the entire application lifecycle.

The first step in ALM governance is business case development. As Figure 2 shows, this analysis happens before the development process begins. Once the business case is approved, application development starts, and governance is now implemented through project portfolio management. In some organizations, this is simple: A project manager might be attached to the development team, or one of the technical people on the team might take on this role. Other organizations use a more formal approach, relying on a centralized project management office to enforce established procedures. And with some processes, such as Scrum, the development team might not even have a formal project manager. However it's accomplished, some kind of project management is all but unavoidable.

Once the completed application is deployed, it becomes part of the organization's portfolio of applications. An application is an asset like any other, and so the organization needs an ongoing understanding of its benefits and costs. Application portfolio management (APM) provides this, offering a way to avoid duplicating functions across different applications. APM also provides governance for the deployed application, addressing things such as when updates and larger revisions make business sense. In fact, examining the APM section of the Governance line in more detail would show that it contains business case development and project portfolio management for each of the revisions to the application shown on the Development line.

Governance is the only thing that extends throughout the entire ALM time span. In many ways, it's the most important aspect of ALM. Get it wrong, and you won't come close to maximizing the application's business value.

ASPECTS OF ALM: DEVELOPMENT

While equating ALM with the software development process isn't accurate, development certainly is a fundamental part of every custom application's lifecycle. Figure 3 takes a closer look at this aspect of ALM.

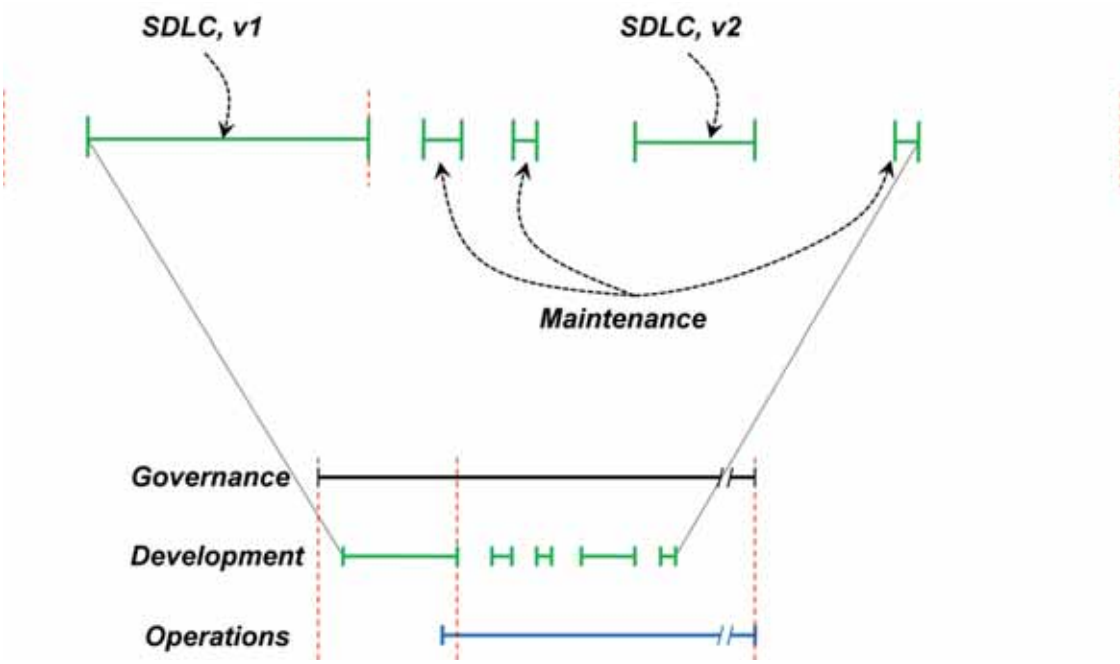


Figure 3: Development occurs in the first part of an application's lifecycle, then happens periodically as the application is updated.

Once the business case is approved, the software development lifecycle begins. If we expanded the SDLC parts of the Development line shown in the figure, a modern process would probably show software development as a series of iterations. Each iteration would contain some requirements definition, some design, some development, and some testing. This iterative style of development isn't always appropriate—some projects are still better done using more traditional methods—but it's becoming the norm in many areas.

Once the SDLC process for version 1 of the application is complete, the application is deployed. For most applications, however, deployment doesn't mark the end of development. Instead, the application needs periodic updates, as shown in the figure, and perhaps one or more full SDLC efforts to create new versions, as in this example. For some applications, the money spent on these updates and new versions can exceed the cost of the original development by a significant amount.

Once again, notice the role of SDLC in the overall ALM process. As Figure 2 shows, this aspect is certainly important, but it's far from the whole story. Viewing ALM as synonymous with SDLC is just wrong—it leads to a misunderstanding of what's really required to be successful in this area.

ASPECTS OF ALM: OPERATIONS

Every deployed application must be monitored and managed. Figure 4 shows some of the important aspects of this operations process.

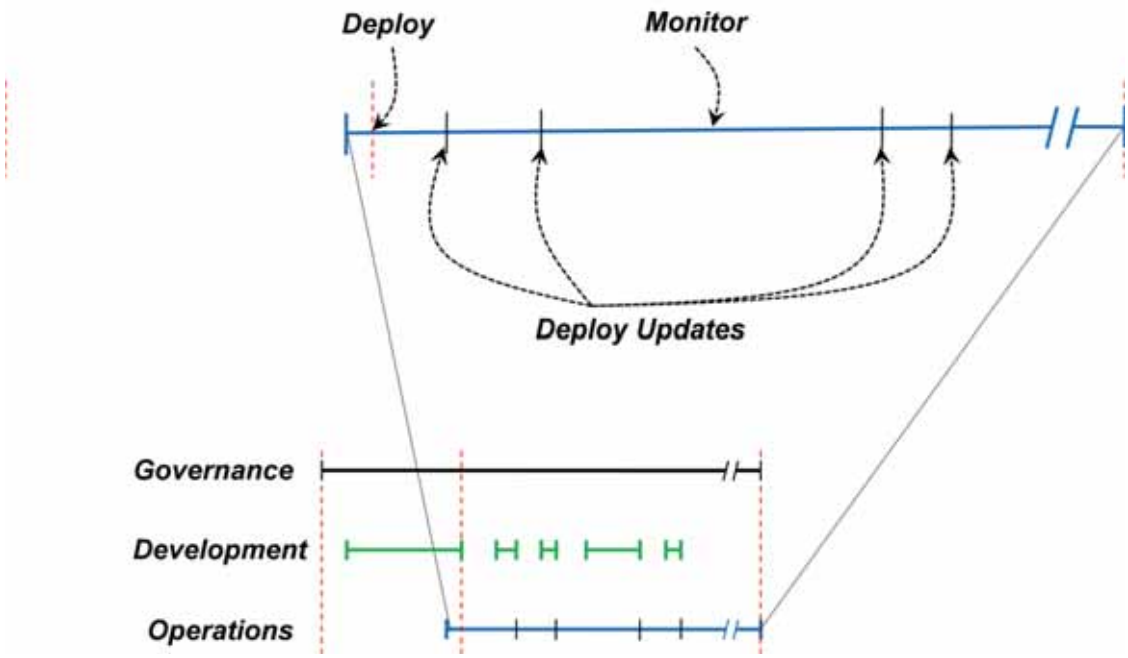


Figure 4: Operations begins shortly before an application is deployed, then continues until the application is removed from service.

As with Governance, the Operations line is intimately connected to the Development line. For example, planning for deployment likely begins shortly before the application is completed, and the act of deployment itself is a fundamental part of operations. Once the application is deployed, it must be monitored throughout its lifetime. Similarly, each update to the application must be deployed once it's completed, as the figure shows.

TOOLS FOR ALM

The three aspects of ALM—governance, development, and operations—are tightly connected to one another. Doing all three well is a requirement for any organization that aspires to maximize the business value of custom software. But this isn't an easy goal to achieve. Each of the three is challenging to get right on its own, and so getting the combination right is even more challenging.

The right tools can make this easier. A number of vendors today provide tools that are horizontally integrated, i.e., tools that work together well on one of the three lines. For example, Microsoft's Visual Studio brings together a range of tools supporting several aspects of the development process. Yet tools

should be integrated not just horizontally but vertically as well, helping organizations make connections across the three lines. For instance, project management tools should be connected to development tools, which in turn should have connections to the tools used for operations.

Tools should be integrated not just horizontally but vertically too.

These connections are beginning to appear. Visual Studio, for example, can connect with Microsoft's Project Server to help project managers get up-to-date information on what developers are doing. There's still plenty of room for improvement, however, and no vendor today offers a set of ALM tools with full vertical integration across all three lines.

CONCLUSION

ALM is much more than just writing code. All three aspects—governance, development, and operations—are important. Think about a project that gets the initial governance aspects wrong, for example, perhaps by not understanding the business needs or failing to get the right stakeholders involved. No matter how well the organization does development and operations, this project won't provide much business value. Similarly, a project that targets the right problems using a first-class development process might ignore operational issues, such as providing enough resources to run the application reliably. Once again, the business value this investment provides won't be as large as it should be. Taking a broad view of ALM can help organizations avoid problems like these.

Maximizing the value of the applications we create means doing all three aspects of ALM well. Achieving this goal isn't easy, especially when today's ALM tools aren't as well integrated as they could be. Yet there's no way around it: Taking a broad, holistic view of ALM is essential for improving this critical business process.

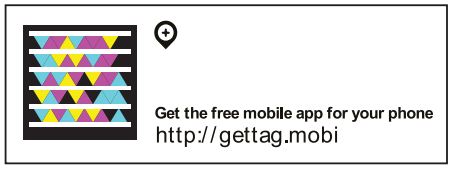
ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technology.

APPLICATION LIFECYCLE MANAGEMENT AND BUSINESS STRATEGY

DAVID CHAPPELL

SPONSORED BY MICROSOFT CORPORATION



Every organization pursues strategies to differentiate it from its competitors. For the vast majority of these efforts, custom applications are essential in providing this differentiation. The uniqueness they make possible might well be the most important way that information technology provides business value.

Yet custom applications don't magically spring into existence. They also don't happily chug along unchanged forever. Instead, both the creation and operation of custom applications are accomplished through the process of *application lifecycle management (ALM)*.

The link between ALM and business strategy isn't always clearly understood. From the right perspective, however, it's clear that for a modern organization to be good at strategy—and thus at providing long-term profitability—it must also be good at ALM.

FROM INNOVATION TO OBLIGATION

The essence of business strategy is being different. A firm might do different things from its competitors, for example, such as offering different products or addressing different markets. Alternatively, it might do the same things in different ways, such as providing a lower-cost service. In either case, what marks an action as strategic is its ability to differentiate an organization from its competitors, letting it provide unique value to customers.

Competitive advantage flows from successful differentiation. Yet keeping good ideas secret is hard, and so remaining different is challenging. Figure 1 shows what typically happens.

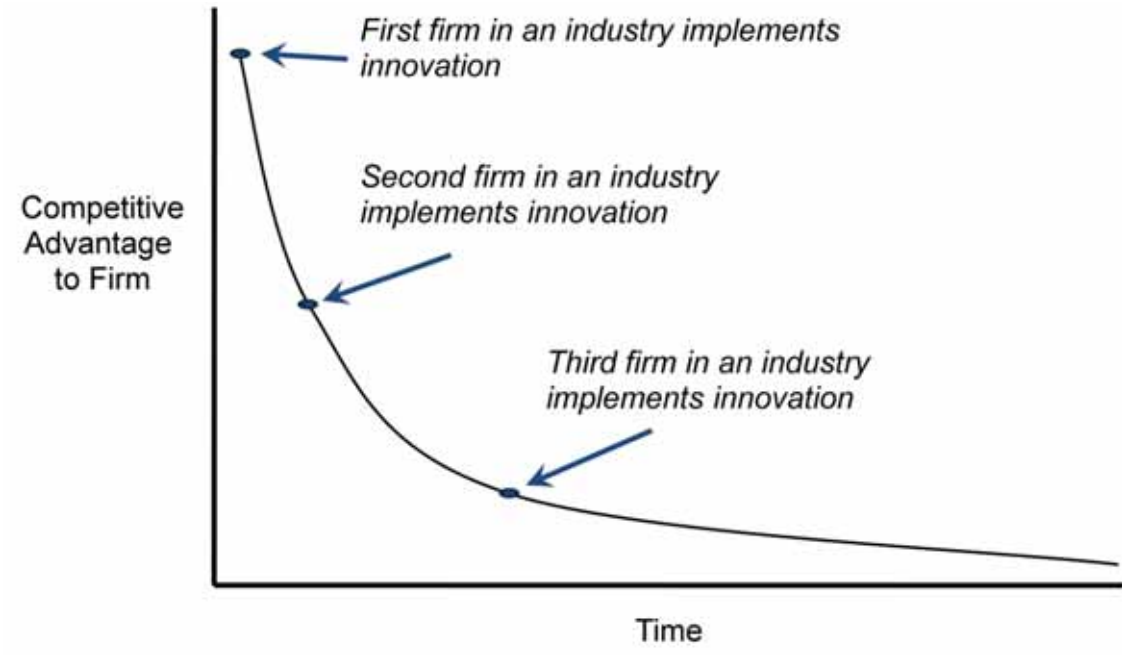


Figure 1: Successful innovations by a strategic leader eventually become obligations for other firms in the industry.

The first firm in an industry to implement a successful innovation gains a significant competitive advantage. The second firm to implement this strategy also derives some advantage from it. By the time the third firm in an industry follows suit, the new approach is usually well on its way to becoming a best practice. Anybody who doesn't implement it is likely to be at a competitive disadvantage. What begins as an innovation becomes an obligation.

What begins as an innovation becomes an obligation.

There's no shortage of examples that illustrate this. The first airline to add self-service check-in kiosks, for instance, gained a significant competitive advantage. The second airline to add self-service check-in kiosks also got some advantage. Yet today, kiosks are a cost of doing business for every airline—not having them would be a competitive disadvantage. Similarly, when FedEx let its customers track their packages directly via the Web, this innovation was rightly hailed as a powerful competitive edge. Today, we expect this from all shipping companies—the innovation is now obligatory.

FROM STRATEGY TO UTILITY

Self-service check-in kiosks and Web-based package tracking are good examples of today's reality, which is that most modern business strategies depend on IT. Yet note what happens: Over time, the IT that underlies every successful strategy becomes part of the IT infrastructure. It's fair to say that virtually all of IT was once an innovation that conferred competitive advantage on its first adopters. Eventually, however, everything becomes a utilitarian IT function that must be supported effectively.

Given this, it's possible to divide IT spending into two broad categories:

- *Strategic IT*, spending on new capabilities that directly support new business strategies;
- *Utility IT*, all other IT spending. To a large degree, the technologies in this category represent the accretion of an organization's strategic innovations over many years.

Figure 2 illustrates how these two categories fit with the competitive advantage curve shown earlier.

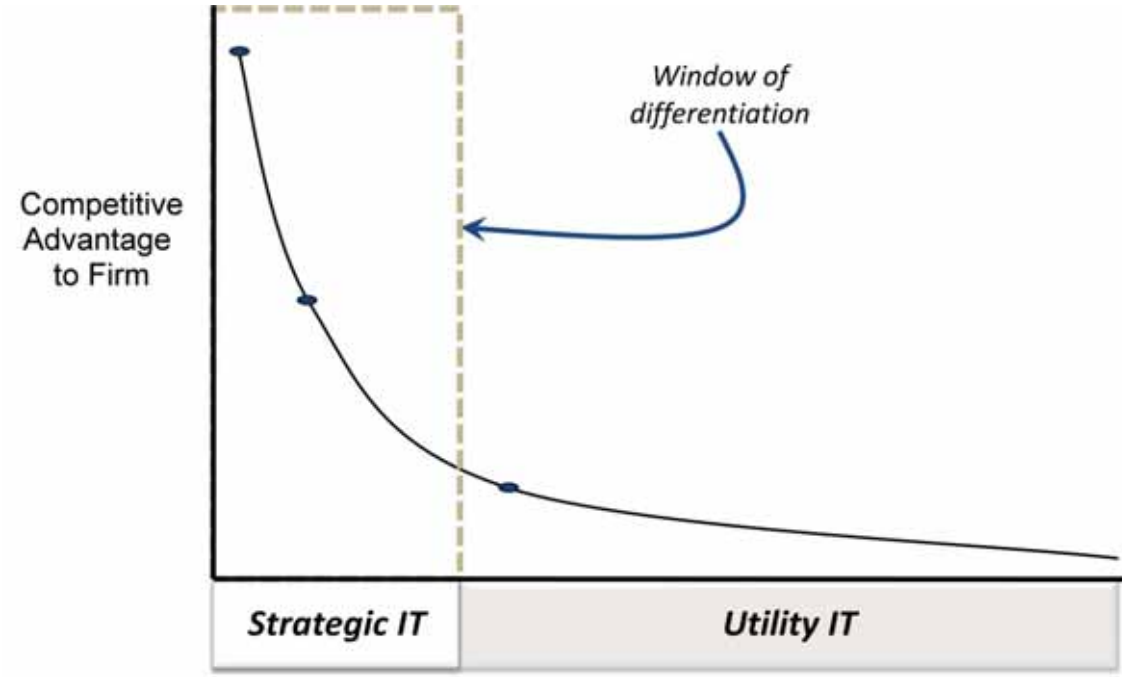


Figure 2: Strategic IT spending creates a window of differentiation.

As the figure shows, strategic IT supports the innovations that provide competitive advantage to an organization. But what does strategic IT really consist of? The answer in most cases is custom applications. While packaged software is important to many businesses, getting strategic differentiation from a generic package is hard. Because of this, innovations such as airline check-in kiosks, Web-based package tracking, and many others depend on custom applications.

Strategic IT supports the innovations that provide competitive advantage to an organization.

Custom applications are rooted in ALM. This implies that how well an organization does ALM can have a powerful impact on its ability to create competitive advantage. As Figure 2 shows, there's a window of differentiation in which innovations can provide a competitive advantage, and so an organization must be able to create custom applications quickly. Yet successful innovations eventually move into the utility category, and so these applications must also be manageable over the long term. Doing this successfully is also part of ALM.

It's worth explicitly connecting all of these dots, and so to summarize, the connection between business strategy and ALM looks like this:

- Business strategy means *being different* from the competition.
- Being different relies on *strategic IT* investments to support that differentiation.
- Strategic IT investments are most often *custom applications.*
- Custom applications are created and managed through *application lifecycle management.*

CONCLUSION

The ability to support new business strategies is at the heart of business/IT alignment. IT decision makers commonly worry about operational risks such as “What if my data center goes down?” Many of them worry less about strategic risks like “What if we can’t support the next new business strategy?” Yet not being able to support a business strategy can have a much bigger long-term impact than a one-time systems failure. And missing a chance to differentiate the business because an IT organization can’t quickly support the CEO’s big new idea won’t help any IT leader’s career. Similarly, when a competitor rolls out a new strategy that lets it be first to exploit a window of differentiation, it’s important to follow suit as fast as possible. All of these things commonly require creating new applications, which depends on effective ALM.

Viewing ALM as a foundation for business strategy isn’t a stretch; it’s just a statement of fact. Because of this, getting good at ALM is an essential part of creating competitive advantage. In any organization where custom software matters—that is, in pretty much every organization today—mastering ALM should be a fundamental goal.

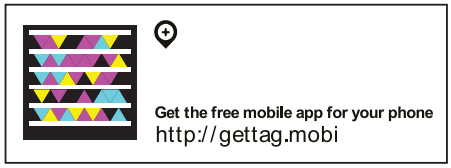
ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technology.

APPLICATION LIFECYCLE MANAGEMENT AS A BUSINESS PROCESS

DAVID CHAPPELL

SPONSORED BY MICROSOFT CORPORATION



Not too long ago, the bond rating agency Moody's disclosed that it had incorrectly assigned a triple A rating to billions of dollars in debt products. In fact, the correct ratings were as much as four levels lower. The error led some of Moody's customers to invest in products that were significantly riskier than they expected. It also led to front-page coverage in the Financial Times and a blot on Moody's reputation.

The incorrect ratings weren't the result of an analyst making poor judgments on these products. Rather, they were caused by a bug in the custom application that Moody's used to model risk. As business processes like this one become more dependent on software, getting that software right gets ever more important. And because the most important software in an organization is typically developed in-house, doing this well has become a foundation for business success.

The set of activities required to create and run custom applications is known as *application lifecycle management (ALM)*. ALM supports many business processes today, and it's also a critical business process in its own right. Any organization that creates custom software should take the ALM process at least as seriously as it does any other important business process. Being better than your competitors at creating custom software can provide a significant competitive advantage.

HOW ALM SUPPORTS BUSINESS PROCESSES

Organizations are defined by their business processes. To understand what an organization does, you need to understand its processes. In a very real sense, improving that organization—making it more responsive to change, more profitable, and more valuable—means improving its business processes.

Supporting those processes with software can help. In a recent Harvard Business Review article[1], Andrew McAfee and Erik Brynjolfsson describe their research into why this is so. One of their key findings is that because software can be easily installed throughout an organization, an improved business process can be quickly replicated in every store, every factory, or every office. Just as important, a software-based process is self-enforcing. Rather than relying on, say, training manuals and hoping that employees follow a new process, the software itself carries out the process in a consistent way. According to McAfee and Brynjolfsson, the ability to embed a better business process in software, then replicate it reliably across an organization, is a primary way to create competitive advantage with information technology.

Yet getting this kind of unique advantage from off-the-shelf software is hard. Almost by definition, packaged software implements well-understood, common processes. And since the same package is available to everybody, differentiating your organization with packaged software isn't easy. Instead, real differentiation comes from unique business processes that are better than those used by your competitors. For unique processes supported by software—which today means nearly all unique processes—custom applications are required. This is exactly why the ALM process is so important, as Figure 1 illustrates.

[1] "Investing in the IT That Makes a Competitive Difference", Harvard Business Review, July-August 2008

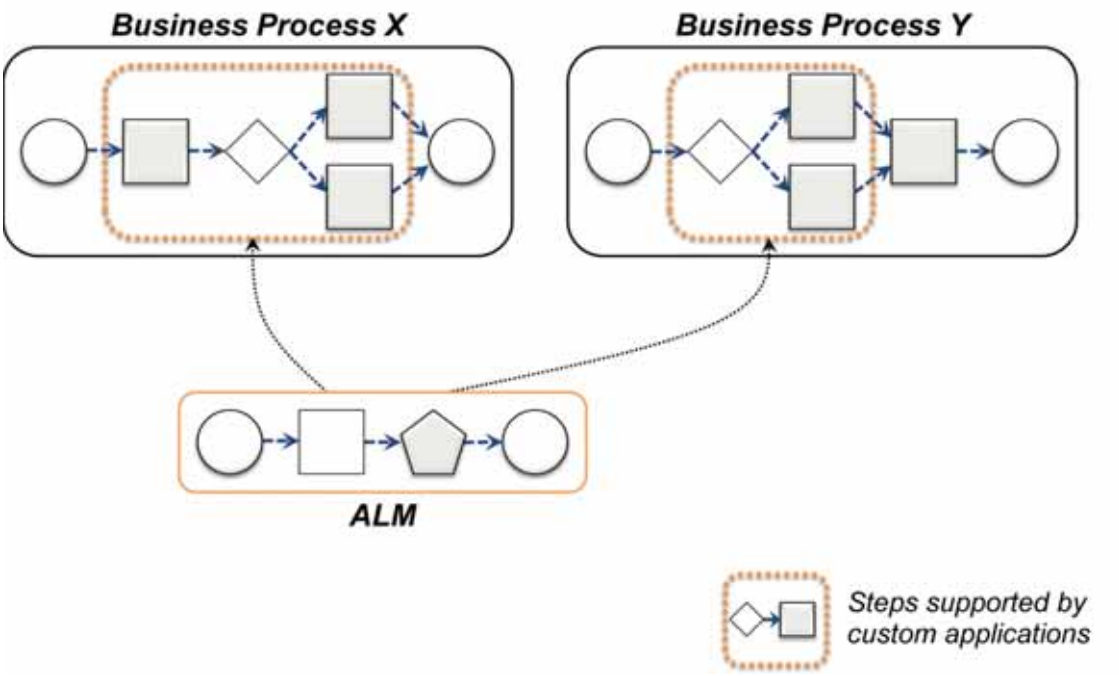


Figure 1: The output of the ALM process—custom applications—provides essential support for other business processes.

As the figure shows, custom applications might underlie all of the steps in a business process, as in process X, or only some of them, as in process Y. In either case, the custom applications that support this process are created, operated, and maintained by the ALM process, as shown here. How fast a new business process can be rolled out and how quickly it can be changed are largely dependent on how fast the applications it relies on can be built and updated. Being good at ALM is a prerequisite for being good at creating and changing business processes.

Being good at ALM is a prerequisite for being good at creating and changing business processes.

Getting better at ALM can even affect the build vs. buy decision for new software. If an improved ALM process makes creating new software faster, cheaper, and less risky, an organization might choose to build more often and so get more customized applications that add competitive value. In a world that runs on code, getting better at creating that code is unquestionably a good thing.

UNDERSTANDING ALM AS A BUSINESS PROCESS

ALM supports business processes, and ALM is itself a business process. It's common to equate ALM with the software development lifecycle (SDLC), viewing it as solely focused on creating applications. Yet every deployed application must be monitored and managed, which means that operations are also part of an application's lifecycle. Just as important, business processes change, and so the ability to maintain and modify an application after it's deployed is also central to ALM. The reality is that ALM is more than just SDLC.

Still, it's fair to say that improving the SDLC aspects of ALM will yield the most benefit in a typical organization. Getting really good at, say, running software efficiently might give you a cost advantage, but it's not going to catapult your business forward. Being really good at software development—creating innovative applications faster than your competitors—can enable this kind of business leap. Yet especially in its development aspects, ALM isn't like most other business processes. Software development can't be a repetitive process like, say, employee onboarding or order-to-cash, and it's certainly not as regular and predictable as a manufacturing process. While these more traditional processes must deal with occasional exceptions, the basic flow and even how long the process takes to complete is typically much the same each time it's executed.

Creating software is effectively a form of new product development.

ALM's SDLC aspects aren't like this. Instead, creating software is effectively a form of new product development. The process looks at least somewhat different each time, and how long it takes to complete can vary. To manage this variation, modern approaches to SDLC use multiple iterations, as Figure 2 shows.

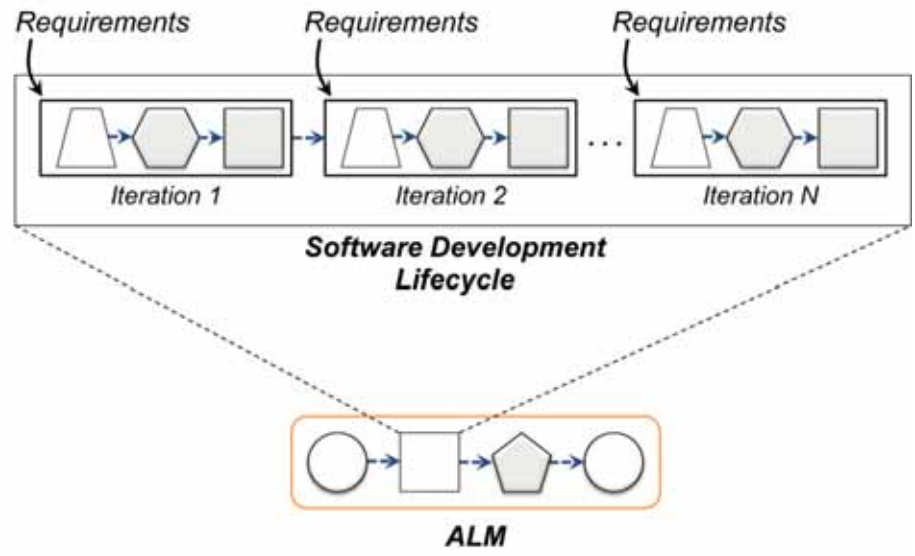


Figure 2: In a modern ALM process, the software development lifecycle is divided into several iterations, with prioritized requirements input to each one.

Rather than rigidly defining all of the requirements up front, then building software that meets these requirements and testing it—the traditional waterfall approach—iterative software development performs each of these steps multiple times. At the start of each iteration, the project's requirements can be changed and re-prioritized, as Figure 2 shows. During each iteration, the development team spends some time on design, some time writing code, and some time testing that code.

While this approach might seem inefficient, it's much better at dealing with the inevitable changes in requirements that crop up during the development process. For example, the people who will use a new application often don't know exactly what they want when development begins. An iterative approach lets the development team show its users how things look at each stage, letting them give feedback that affects the project's direction. Because new requirements are incorporated at the start of each iteration, changes can be made at well-defined points in the process. (Being iterative is also a fundamental

characteristic of what are known as *agile* processes, although there's some disagreement on the precise definition of this term.)

Like other business processes, ALM itself can be supported by software. Just as with any automated process, an organization that creates a first-rate ALM process in one group can use ALM software to help replicate that process quickly across the company. But because ALM is different from many other business processes, the software that supports it doesn't implement a fixed, unchanging set of steps. Instead, tools for ALM support the fundamentals of the process, such as tracking requirements, maintaining code and documents in a centralized place, and keeping track of project status, e.g., the current number of bugs and the rate at which those bugs are fixed. Just as the right software can make other business processes more effective, the right ALM software can help improve the way an organization creates and uses custom applications.

CONCLUSION

Most business people understand the strategic value of differentiated business processes. Yet the importance of ALM in supporting those processes gets much less attention. In reality, any organization that creates custom software should take the ALM process as seriously as it does any other critical business process. ALM might be even more important, since it creates the underpinnings for the others.

At Moody's, for example, the business process for rating debt is surely among the most important in the company. Yet this process relies on a custom application, which means it ultimately depends on Moody's ALM process. Being better at ALM might have helped the company find and fix the bug that led to mis-rating debt before it caused any damage. And Moody's is hardly alone—nearly every organization has similar stories.

Being better than your competitors at creating and using custom software can bring substantial competitive advantage. Similarly, being worse can put you at a significant disadvantage. If your organization doesn't see ALM as one of its most important business processes, it's time to change that view.

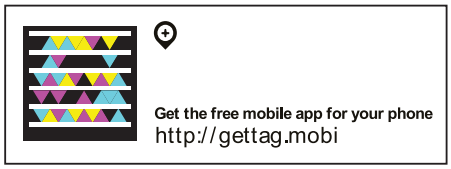
ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technology.

TOOLS FOR TEAM DEVELOPMENT: WHY VENDORS ARE FINALLY GETTING IT RIGHT

DAVID CHAPPELL

SPONSORED BY MICROSOFT CORPORATION



Most software development is done by teams of people. Yet even though tools to support team-based development have been available for some time, they weren't always as useful as they might have been. Today, the vendors who create those tools have reached a consensus on what the real problem is, and they're providing tools to solve it. The result is team development tools that focus on the right thing: optimizing the end-to-end development process.

THE ORIGINAL PROBLEM: COMBINING TOOLS

To understand what today's consensus is and why it's important, it's useful to look first at the history of development tools. Figure 1 illustrates how the tools developers use have evolved over the last few decades.

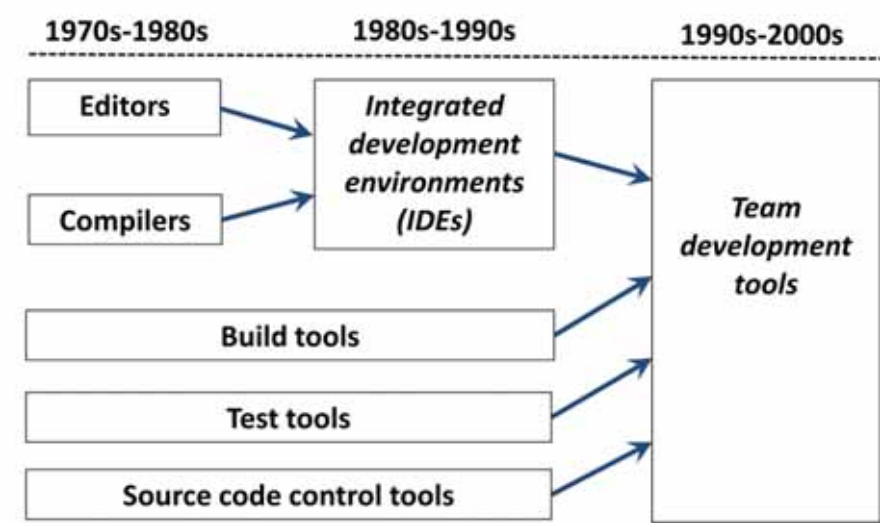


Figure 1: What began as separate tools have evolved into unified team development tools.

In the 1970s, all of the functions performed in the software development process were supported by different tools. Developers created code using an editor, then explicitly invoked a separate compiler when needed. Similarly, the tools used to build complete executables, test code, and manage versions of that code were all distinct from one another.

Over time, these separate tools for software development have been combined. In the early 1980s, editors and compilers were united to form *integrated development environments (IDEs)*. Developers loved IDEs—yoking these previously separate tools together meant, for example, that errors could be fixed immediately right in the source code—and so IDEs caught on quickly. Combining these two tools made sense, because it let developers be significantly more productive.

Over time, what were once separate tools for software development have been combined.

As build tools, test tools, and source code control tools became more widely used, it also made sense to combine them with one another and with IDEs. Just as integrating compilers and editors allowed things that weren't possible when the two were separate, grouping all of these tools together was also a step forward. The result was *team development tools*, an advance that began appearing in the 1990s.

Unlike IDEs, which caught on quickly, organizations were slower to embrace team development tools. This slowness was partly because this kind of unified tool is harder to adopt than an IDE. Moving from a separate editor and compiler to a unified IDE requires only individual developers to change—it's easy. Moving from separate tools for writing and compiling code, doing builds, testing, and source code control to unified tools is significantly harder. More people—and more tools—have to change. Just as important, the development process itself must change.

Making this kind of process change can be challenging, creating various kinds of resistance. People might be reluctant to give up a favorite tool in some area, for example. Another potential roadblock stems from an important benefit of team development tools: their ability to automatically track information about the development process, then generate reports on a project's progress. While this transparency is a great boon for the people who manage the project, it also means that people on the dev team have nowhere to hide. If a developer hasn't checked in any new code in the last week, this problem will show up quickly.

There's also another important reason why organizations have been slow to adopt team development tools: The tools weren't initially as valuable as they might have been. In fact, it's fair to say that, as in most new areas, the real problem wasn't fully understood at the beginning. Today, however, that's no longer true: The challenge has become clear.

THE REAL PROBLEM: OPTIMIZING END-TO-END FLOW

When vendors (and open source projects) first created team development tools, they commonly built or acquired the best tool they could for each area: development, testing, source code control, and so on. This kind of point optimization led to some excellent tools, but it didn't solve the right problem. In fact, the goal in team-based software development isn't creating great tools that optimize parts of the process. It's optimizing the process as a whole.

To see why this is so important, think of a manufacturing process where a part takes three hours to manufacture, then sits in a warehouse for three weeks waiting to be shipped to a customer. Cutting the manufacturing time in half won't make the customer any happier—what needs to be optimized is the end-to-end flow. Similarly, an organization that improves how its developers write and compile software won't see much benefit if, say, testing doesn't also get better. By providing a cohesive environment for creating software, team development tools can help optimize the entire process, not just its individual components.

This idea becomes even more important given how software development has changed over the last few decades. Developers once waited an hour or more to get the results of compiling new code, then waited days or weeks for meaningful test results. Today, creating, compiling, and testing code happen continually—the iterations are much shorter—and the transitions between them are more frequent. Making the process better requires making these transitions as smooth as possible.

Doing all of this requires tools that work together well, with a common way to share information across different aspects of the process. Just gluing together good tools in each area won't work. What's needed is an approach like that shown in Figure 2.

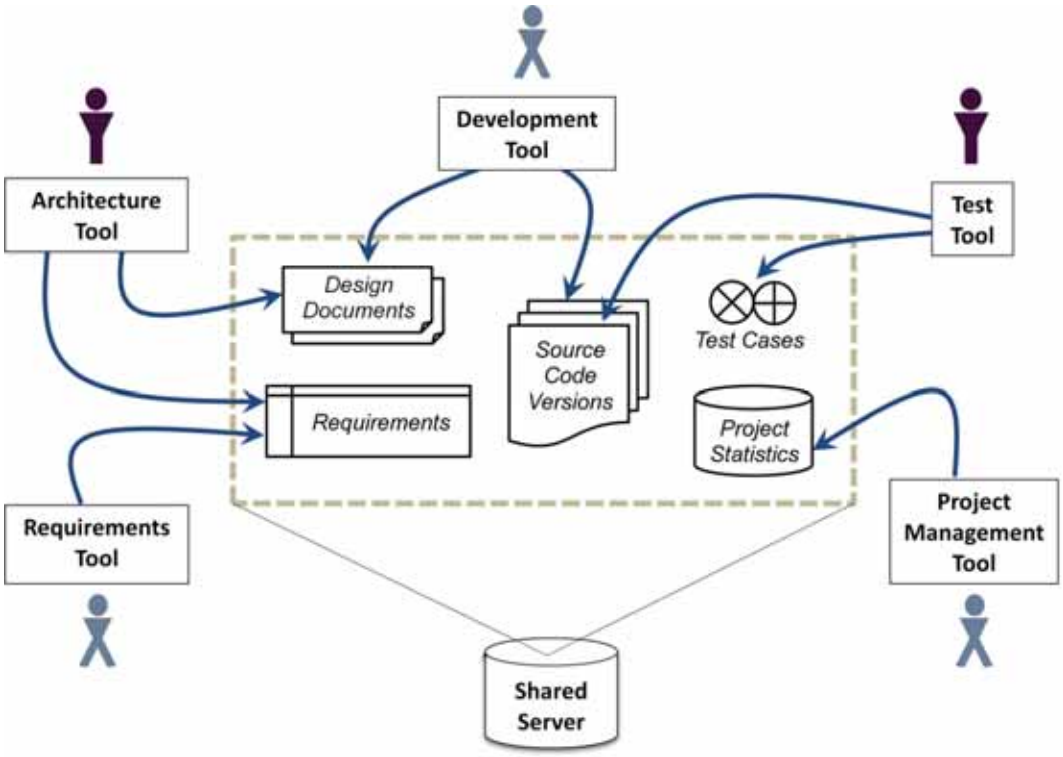


Figure 2: Optimizing the development process as a whole requires a unified approach to storing and working with the artifacts of that process.

Tools used for different purposes—working with requirements, specifying architecture, developing code, testing code, and project management—should all be able to work with a common set of interconnected artifacts stored in a common place. As Figure 2 shows, those artifacts can include requirements, design documents, various versions of source code, test cases, statistics about this development project, and more. This kind of integration allows all sorts of useful things: automatically recording code check-ins, associating test cases with requirements, generating historical reports of bug counts, and more. More important, it allows optimizing the entire process. The transitions between different functions all rely on shared information, and so making those transitions gets easier.

When our industry converges on an architecture, it means there's broad consensus about the best way to do something.

For the most part, the first generation of team development tools didn't take this approach. The vendors have learned from their experience, however, and team-based tools today can provide this broad integration. Microsoft's Visual Studio, for example, provides tools for architecture, development, testing, and more, all of which rely on Team Foundation Server (TFS) to store the artifacts they work with. IBM's Rational offerings take a similar approach, with different tools supporting different functions and Jazz Team Server acting in a role similar to Microsoft's TFS.

When our industry converges on an architecture, it means there's broad consensus about the best way to do something. This consensus has appeared in team development tools. The vendors have figured

out that the real goal is optimizing the development process as a whole, not just individual parts of that process.

CONCLUSION

This is an important moment in the evolution of development tools. Since team development tools are now focusing on the right problem, they have more to offer than they did ten years ago. Organizations doing team-based development—a category that includes almost everybody—can benefit from taking another look at this style of tool.

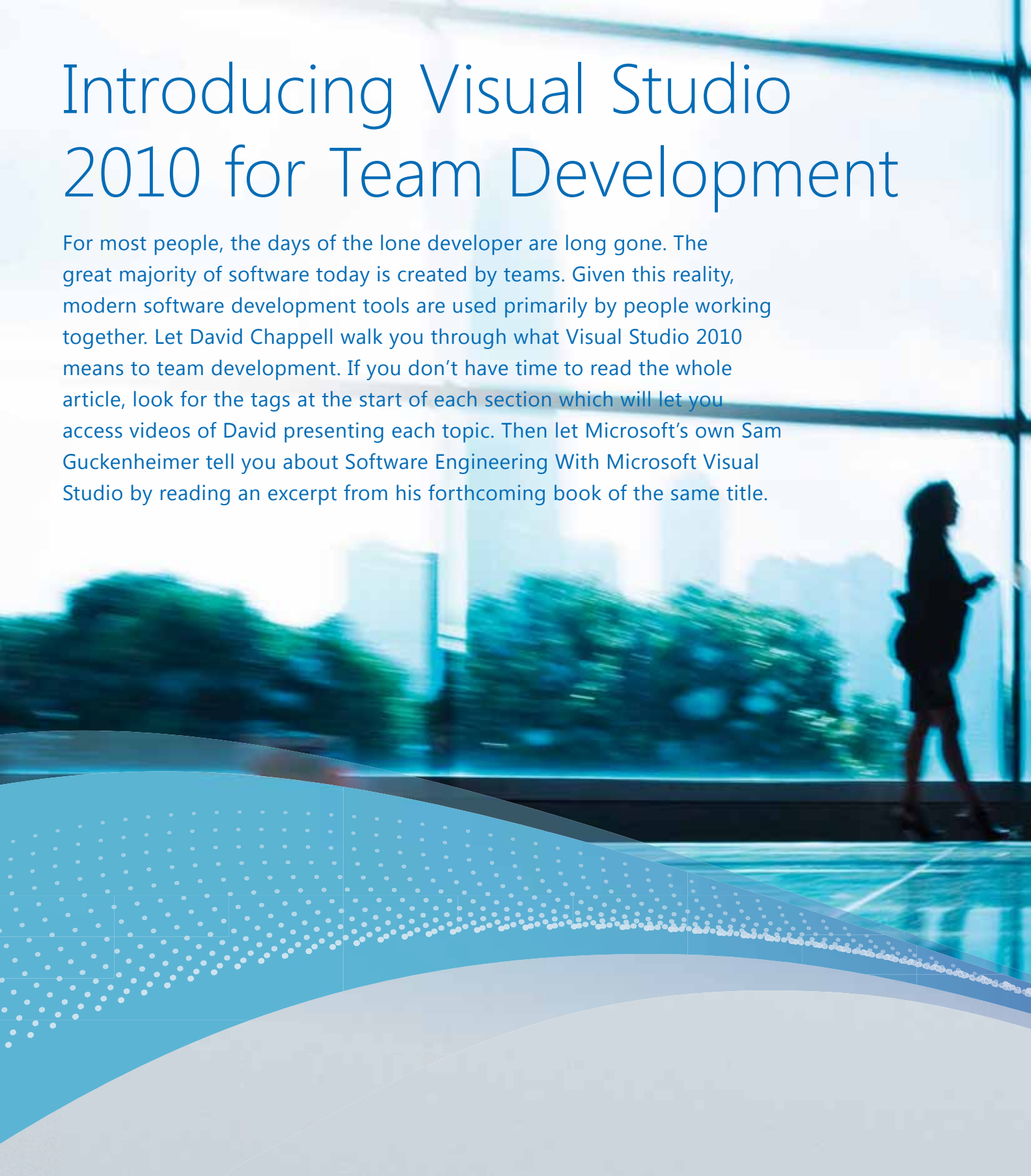
Still, don't expect adopting team development tools to be as easy as adopting IDEs. The challenge of clearly understanding (and perhaps changing) your development process still exists. Nonetheless, just as IDEs made life easier for individual developers by combining what were once separate tools, team development tools can help developers and everyone else on a development team work together more effectively. Given how important software is to most organizations, improvements in how we create it are always welcome.

ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technology.

Introducing Visual Studio 2010 for Team Development

For most people, the days of the lone developer are long gone. The great majority of software today is created by teams. Given this reality, modern software development tools are used primarily by people working together. Let David Chappell walk you through what Visual Studio 2010 means to team development. If you don't have time to read the whole article, look for the tags at the start of each section which will let you access videos of David presenting each topic. Then let Microsoft's own Sam Guckenheimer tell you about Software Engineering With Microsoft Visual Studio by reading an excerpt from his forthcoming book of the same title.

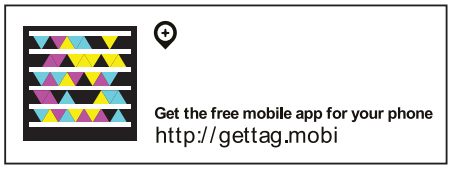


INTRODUCING VISUAL STUDIO 2010

DAVID CHAPPELL

MAY 2010

SPONSORED BY MICROSOFT



TOOLS AND MODERN SOFTWARE DEVELOPMENT

For most people, the days of the lone developer are long gone. The great majority of software today is created by teams. Given this reality, modern software development tools are used primarily by people working together.

One approach to designing these tools is to create a separate tool for each part of the development process. Everyone on the team might then agree to use a particular tool for versioning source code, another for tracking bugs, and a third for managing tests. This approach can certainly work—it's much better than having no tools at all. But what happens when a developer needs to determine which test found a particular bug in code that addresses a specific requirement? Or what if a project manager wants to get a view of the project's overall code quality over time, looking at bug counts, test progress, code churn, and more? Answering these kinds of questions requires integration across the team's tool set. Even though each tool might be great at what it does, development with a set of disconnected tools has limits.

An alternative approach is to create an integrated set of development tools explicitly designed to work together. While a particular member of this tool set might not have every feature found in a standalone version of that tool, the connections between the tools let the team work more effectively. Answering questions that span tools gets easier, as does handing off work between different team members. Since the goal is to optimize the development process as a whole, this integrated approach can make the process of creating software significantly more effective.

Achieving this is the goal of Visual Studio 2010. This latest release of Microsoft's flagship development environment aims at providing a unified set of tools for a variety of development needs. The intent is to be useful in a range of situations, from a large development team spread across three continents to a solo developer working on her own. It's meant to be a modern foundation for what's become known as *Application Lifecycle Management (ALM)*.

Visual Studio 2010 is the successor to both Visual Studio 2008 and Visual Studio Team System 2008. (Microsoft chose to drop the "Team System" label with this release.) The product is large, and so it's available in several different configurations, each with a specific set of functionality. How those configurations look and what's in each one is described at the end of this paper. The goal now is to describe Visual Studio 2010 as a whole, painting the big picture of what this technology family is and how it can be used.

UNDERSTANDING VISUAL STUDIO 2010

Modern software development is anything but simple. Accordingly, the tools that support today's development teams have many moving parts, and Visual Studio 2010 is no exception. One way to get a handle on this product family is to start with a broad look at the components and how they fit together. Once we've done this, we can look more closely at the piece that ties everything else together: Team Foundation Server.

THE COMPONENTS OF VISUAL STUDIO 2010

Visual Studio 2010 has several distinct parts. These parts can connect with one other, with other Microsoft technologies, and with non-Microsoft technologies. Figure 1 shows the main Visual Studio 2010 components and some of the other technologies that are most often used with them.

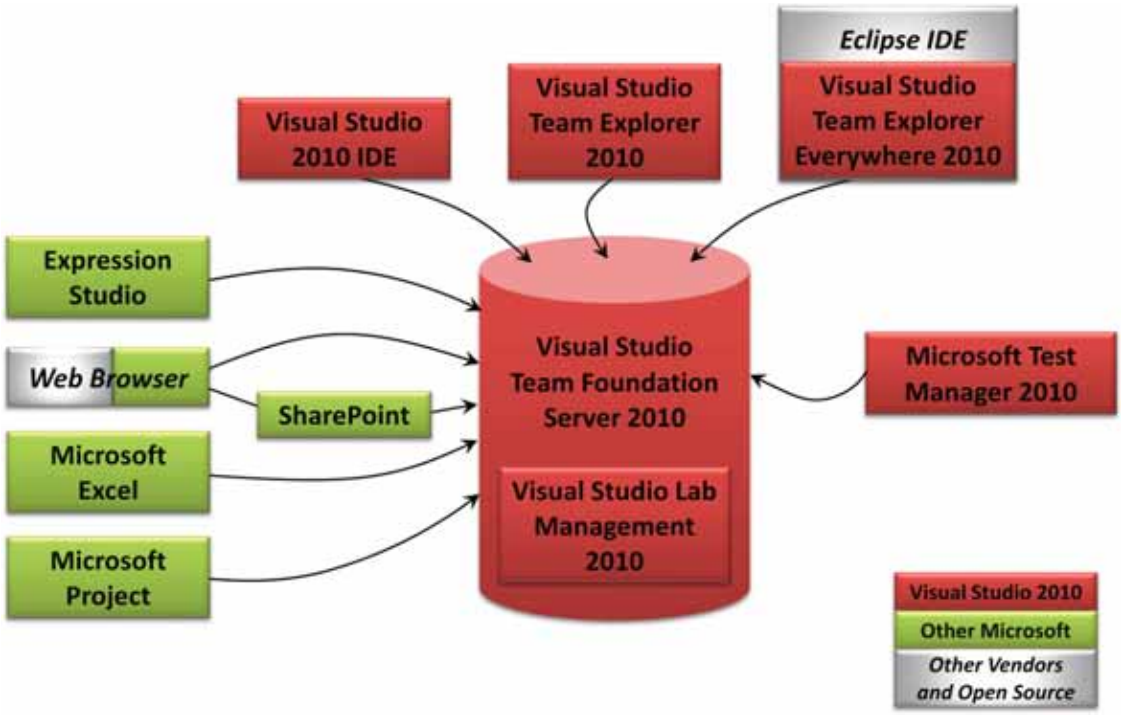


Figure 1: The components of Visual Studio 2010 rely on Team Foundation Server to connect with each other and with technologies from Microsoft, other vendors, and the open source world.

The main components of Visual Studio 2010 (shown in red) are the following:

- *Visual Studio Team Foundation Server (TFS) 2010:* As Figure 1 suggests, TFS is at the center of the Visual Studio story today. Every other part of the product connects to TFS, which acts as a central hub for information throughout the development process. It provides a place to store and manage requirements and other information, version control for source code, build management, bug tracking, test case management, reports based on this information, and more. TFS also provides an application programming interface (API) that lets other software access its services.

- *The Visual Studio 2010 IDE:* Millions of developers use this integrated development environment (IDE) today, either on its own or with earlier versions of TFS. The tool allows creating, compiling, and running code, along with support for testing and other development functions.

- *Visual Studio Team Explorer 2010:* This tool focuses on letting its users access information in TFS, such as reports and build status. It can run on its own, as Figure 1 shows, or inside the Visual Studio IDE.

- *Visual Studio Team Explorer Everywhere 2010:* While some development teams work solely in a Visual Studio world, many don't. Plenty of organizations build applications using both .NET and Java, for example. To help developers using Eclipse-based IDEs work effectively with Visual Studio 2010, the product includes an Eclipse plug-in that allows connecting with TFS.

- *Microsoft Test Manager (MTM) 2010:* Testing is a critical part of software development. While the Visual Studio IDE includes a range of testing support, it's focused on testing done by people with development skills. MTM focuses more on manual testing, providing a tool for generalist testers rather than developers.

- *Visual Studio Lab Management 2010:* This component allows creating and managing virtual machines (VMs) for a test lab. While it's certainly possible to do testing on real physical machines, using VMs can be simpler, faster, and cheaper. Lab Management is accessed through Microsoft Test Manager 2010, and as Figure 1 shows, it's implemented as part of TFS.

Other Microsoft technologies (shown in green in Figure 1) are also commonly used with Visual Studio 2010. They include the following:

- Expression Studio: This product suite includes tools such as Expression Blend to help designers create user interfaces (UIs) for Silverlight and Windows Presentation Foundation (WPF) applications. These interface definitions can be stored in TFS, then used by the Visual Studio IDE.

- Internet Explorer (or another Web browser): Using an aspect of TFS called *Team Web Access*, people can access TFS through a Web browser. This allows making TFS information accessible to a broad range of participants in the development process, including customers.

- SharePoint: Visual Studio 2010 lets a team create a SharePoint-based team project portal for accessing information such as documentation, team calendars, and dashboards with TFS information. As with Team Web Access, this lets people access information about the development process without requiring them to install any component of Visual Studio 2010 itself.

- Microsoft Excel: Many project managers (and even some developers) use Excel to keep track of progress. To help them do this, Visual Studio 2010 lets Excel directly access TFS. The product also provides a range of Excel workbooks and reports for project managers and others to use in the development process.

- Microsoft Project: Along with (or instead of) Excel, many project managers use Project. This tool can also access TFS, allowing things such as creating Gantt charts based on requirements stored in TFS.

A CLOSER LOOK AT TEAM FOUNDATION SERVER

TFS sits at the center of Visual Studio 2010, which makes it the place to start in understanding this product family. In a fundamental sense, TFS is a database. In fact, TFS is built on SQL Server, and it can be clustered for high availability. Yet while TFS does store lots of information, it also provides a variety of useful functions using that data. Figure 2 shows a summary of what TFS contains.

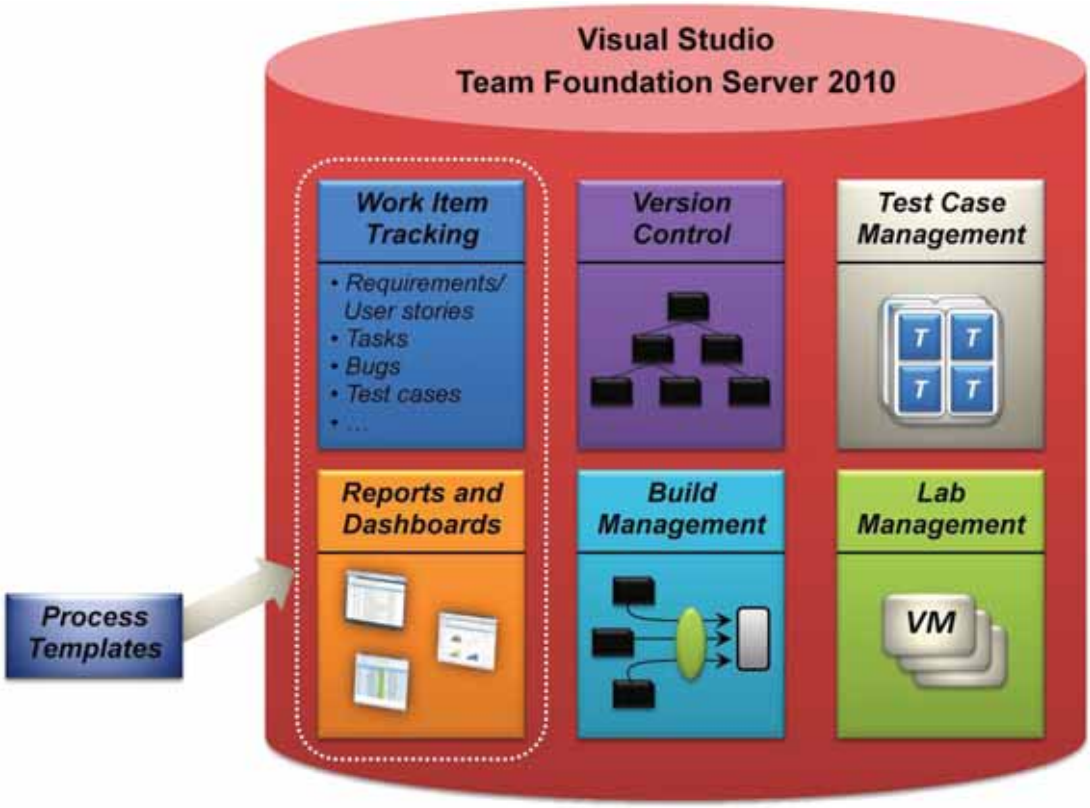


Figure 2: Team Foundation Server provides work item tracking, version control, test case management, lab management, build management, and the ability to create reports and dashboards.

As the figure shows, TFS has several parts:

- *Work item tracking*: The development process uses many different kinds of information: requirements, tasks to be done, bugs, test cases, and more. TFS holds all of this information in *work items*. Each work item is actually a record in the TFS database, and like database records, work items can be related to one another in various ways. They can also be customized, letting organizations create work items that fit their development process.

- *Version control*: A fundamental challenge in software development is keeping track of changes. TFS version control manages updates to source code and other information used in the development process.

- *Test case management*: Testing is a hugely important part of the development process. The goal of this TFS component is to help developers and testers create, use, and manage their test cases and test results. The section on testing later in this paper takes a look at what this aspect of TFS provides.

- *Lab management*: As was shown in Figure 1, Visual Studio Lab Management 2010 is implemented as part of TFS. This component is also described in more detail in the section on testing later in this paper. (Lab Management was called out separately in Figure 1 because it's purchased independently from the rest of TFS, as described later.)

- *Build management*: Creating a runnable application from source code and the other artifacts a development team creates can be remarkably complicated. And once an application has been built from these various parts, it's useful to run tests to verify the quality of the build. TFS build management provides an automated mechanism for doing these things and more.

- *Reports and dashboards*: Whether a team is small or large, the development process creates a mountain of information. The reports and SharePoint dashboards that TFS provides help the people involved in that process understand and make sense out of this mass of data.

Figure 2 also shows one more component: *process templates*. To understand their role, think about how teams work today. Many different development processes are in use, each with its defenders and detractors. Some approaches, such as waterfall processes, place people in quite distinct roles: architect, developer, tester, project manager, and more. Others, such as Scrum and other agile processes, define multidisciplinary teams with each team member performing a range of functions. Visual Studio 2010 isn't expressly designed to support Scrum or waterfall development or any other particular process. Instead, this tool family is meant to be useful with whatever development process a team is using.

Yet the set of work items, reports, and dashboards that a team would find useful varies across different processes. TFS addresses this reality with process templates. Each process template defines things such as specific work item types, reports, dashboards, and Excel workbooks. Organizations are free to create process templates that match their own processes, and others are available from third parties. To make life simpler, however, Visual Studio 2010 itself includes two process templates. They are:

- MSF for Agile Software Development, defining work items, reports, and more for an agile development process.

- MSF for CMMI Process Improvement, defining work items, reports, and more intended to be used in projects that need to record more information about the development process.

And don't be confused: Even though both template names begin with "MSF", neither one requires adopting the Microsoft Solutions Framework. Along with these two, Microsoft also makes available a downloadable process template for Scrum, today's most popular agile process.

Having a broad sense of TFS is important. But understanding what TFS actually does requires more. The next sections take a closer look at some of its components.

Work Item Tracking

Creating software requires keeping track of lots of information. What requirements must the software meet? What tasks are remaining for the project, what's the status of each one, and what are their relative priorities? What test cases have been created, and what bugs have those tests found? And perhaps most important, how are all of these individual pieces of information related to each other?

In Visual Studio 2010, all of this and more is done with work items. The exact set of work items available for a particular project depends on which process template is used. Both the MSF for Agile Software Development template and the MSF for CMMI Process Improvement template contain work items for requirements (called "user stories" in the Agile template), tasks, test cases, and bugs, for example, but

each also adds a few more. For instance, the CMMI template adds work items for risks, reviews, and change requests, all things that make sense for the development style this template supports.

Whatever work items are used, TFS allows linking them together in useful ways. For example, a requirement work item can be linked to the task work items that spell out what must be done to implement that requirement. It might also be linked to one or more test case work items that verify the correctness of the requirement's implementation. Each test case work item, in turn, can be linked to one or more bug work item that this test has uncovered. Figure 3 illustrates this idea.

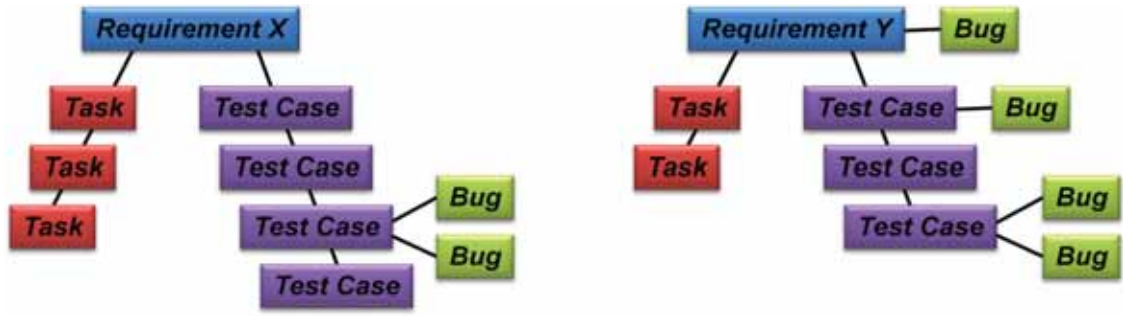


Figure 3: Work items can be linked together, letting team members and other project stakeholders trace the connections among them.

To see why this is important, think about what happens when information about the development process is not grouped together in a single place. Without this, answering important questions such as how well-tested the software is gets hard. There's no simple way to work out what tests have been run against each requirement and what bugs are outstanding for each of those tests. With a unified approach, answering these questions is straightforward: Each requirement is connected directly to its test cases, their bugs, and more, as Figure 3 shows. Providing this kind of traceability is a fundamental reason why unifying software development tools makes sense.

Work items can be created, examined, and modified using various tools. A project manager might choose Excel, while a developer accesses work items from within the Visual Studio IDE. If the team decides to allow it, even customers can access work items from a Web browser. This allows the people who are paying for a project to check progress, file bugs, and otherwise participate in the development process.

Version Control

When multiple developers are working on the same code base, storing that code in a version control system makes life significantly easier. Each developer can now check out the code she needs, work on it, then check it back in. Conflicts with changes made by other developers are handled by the version control system. This functionality is referred to by various names, including source code control and software configuration management. Whatever it's called, it's an essential part of software development today.

Visual Studio 2010 provides this with TFS version control, a shared repository for source code and other artifacts used in the development process. For example, both a developer writing source code with the Visual Studio IDE and a designer using Expression Blend to create a XAML user interface file can store their work in TFS version control. TFS also supports versioning of database schema, something that helps keep code and the data it uses in sync with each other.

TFS version control allows branching and merging, letting different groups work on the same code, then later combine their changes. And to make life easier for geographically distributed teams, developers in a remote location with a slow network link can use a local TFS proxy server. This proxy server provides a local cache for version-controlled information that gets regularly synced with the master copy in a central TFS server.

Like most things in TFS, the information in version control can be accessed via reports and dashboards. For example, a project manager can use these to see what has been checked in to version control by each developer. This is more efficient for the project manager, and it also helps reduce the time developers need to spend writing progress reports.

Build Management: Team Foundation Build

Throughout the life of a development project, a team will use the source code, database schemas, and other information in TFS version control to build executable versions of the application. TFS supports this using a technology called *Team Foundation Build*. Even though it's considered part of TFS, Team Foundation Build is typically installed on a separate machine from the other TFS components. Compiling software, a key part of many build processes, is a resource-intensive task, and so moving it off of the main TFS machine avoids slowing down everything else that this server does.

Build processes can be quite complex, with many steps. Once a build is complete, for example, Team Foundation Build can run *build verification tests (BVTs)* to check the build's quality. It can also deploy the newly built software to specific machines, automatically making the code available to testers. To help define these processes, Team Foundation Build allows specifying the steps in a build as a workflow implemented using Windows Workflow Foundation. The product includes a wizard for visually laying out the steps in a build process, a tool that's meant to make it easier to create and modify complex builds. And as usual with TFS, data about build processes and their results can be accessed via reports and dashboards.

Team Foundation Build supports several approaches: manual builds on demand, scheduled builds (such as nightly builds), and building whenever a change is checked into TFS version control. While frequent builds are useful—running a build for each checked-in code change is the cornerstone of continuous integration—what happens if a change breaks the build? If Team Foundation Build can't complete the build, no other developer or tester on this project can get a working executable until the error is corrected. This can be a significant problem, since one developer making a single mistake can stop all progress. To avoid this, Visual Studio 2010 can require *gated check-in* for some or all of a team's developers. With this option, changes submitted to version control undergo their own separate build (including BVTs) to make sure they're okay before the changes are committed. This significantly reduces the chance that errors in these newly checked-in changes will break the main build.

Reporting and Dashboards

The three aspects of TFS described so far—work item tracking, version control, and build management—create lots of data. The fourth aspect, reporting and dashboards, focuses on making this data available in useful ways. Reports can be created using either Excel or SQL Server Reporting Services, and they can present current or historical views of a development project. Dashboards, created using SharePoint,

provide a direct way to monitor a project's status in various areas. For both, having all of the project's information available in one place allows a broad view, correlating data from different areas.

As with work items, exactly which reports and dashboards are available depends on the process template a team uses. For example, the Agile template provides more than 30 reports, including Stories Overview, Builds Status, Test Plan Progress, Bugs by Priority, and Bug Trends, all of which can help team members see where the project stands. Figure 4 shows an example of the Stories Overview report viewed in the Visual Studio 2010 IDE.

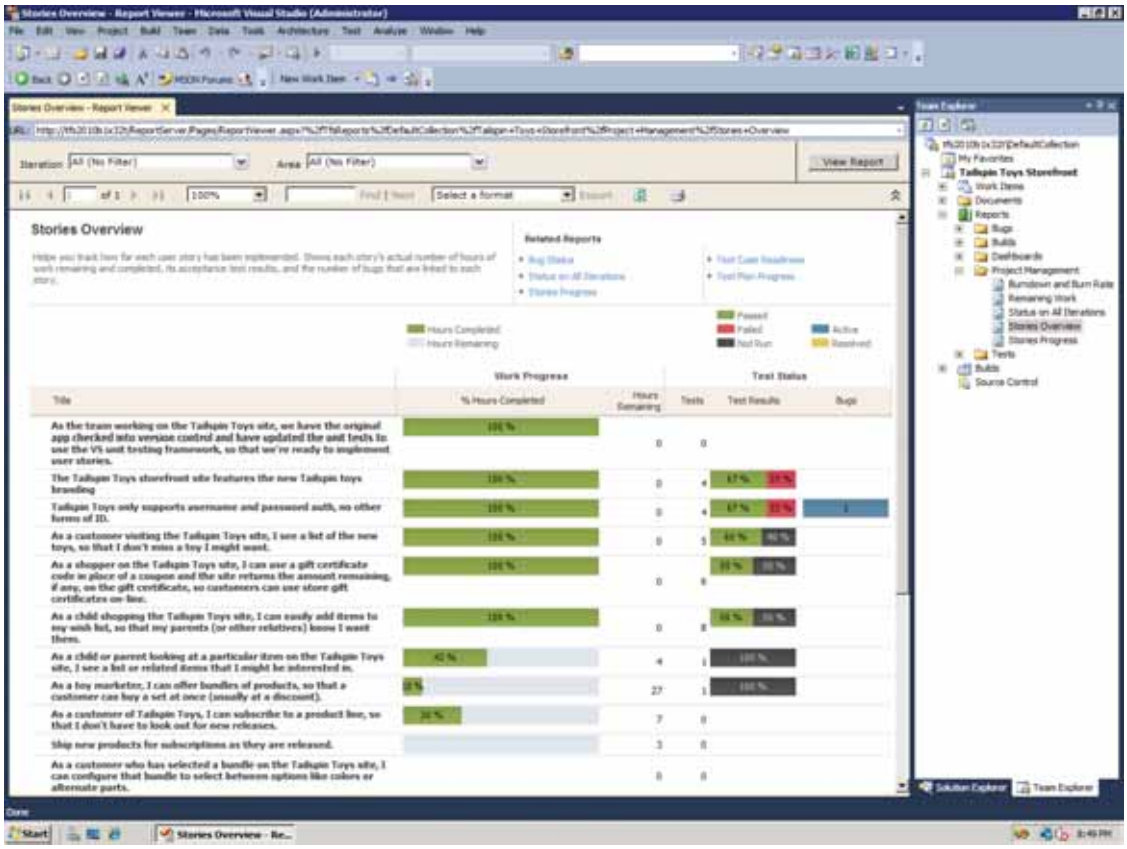


Figure 4: The Stories Overview report summarizes an agile team's progress in addressing the project's user stories.

On the left in this report is a list of the project's user stories (i.e., its requirements). For each one, the report shows the work remaining to complete it, a summary of test results, and the number of outstanding bugs. Notice how this report makes use of a range of information stored in TFS, including work items containing requirements, the bugs associated with each one, test results, and more. The pane on the right lists other available reports. It's possible to customize these reports or create entirely new reports using Excel or the Report Designer in SQL Server Reporting Services.

Reports can provide historical context, which makes them quite useful in understanding a project's status. If 90% of the project's tests passed today, is that good? It's impossible to know unless you can also see what percentage of tests passed two weeks ago or two months ago. Because TFS maintains historical data about a development project, then makes it accessible via reports, the people in charge of a project can

have a broader, more accurate view of its progress. And because TFS collects much of this data automatically, keeping track of code check-ins, builds, and more, project managers need no longer harass team members for status reports, then cut and paste the results.

Dashboards are created with SharePoint, and so they're accessed through the team project portal. As usual with SharePoint, a dashboard can be customized, letting team members display information from outside the TFS world such as important upcoming events. Both the Agile and CMMI process templates provide predefined dashboards, including Bugs, Build, Quality, Test, and Project. The templates also include an option called My Dashboard, which gives each team member a customized view showing just his tasks, his bugs, and other information that's relevant solely to him. Figure 5 shows an example of the Project dashboard for an agile development project.

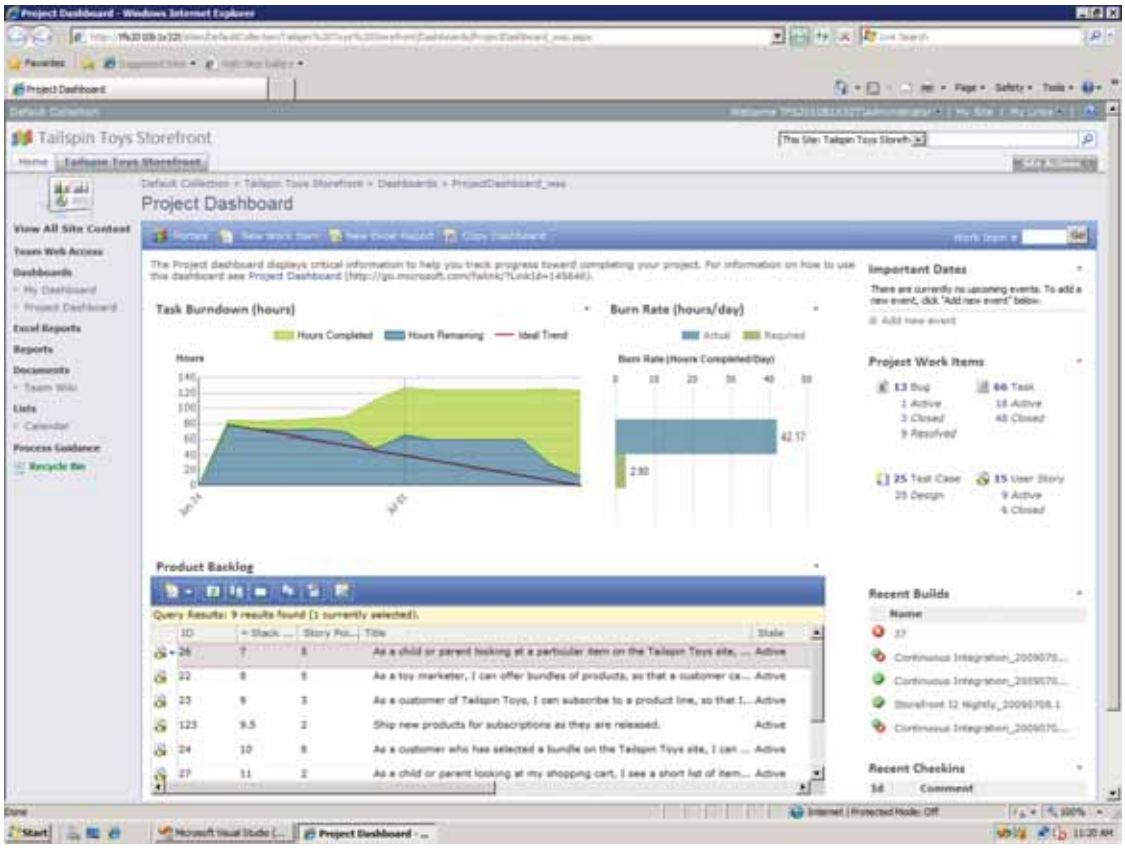


Figure 5: The Project dashboard displays the task burndown, product backlog, recent check-ins and builds, and other information about the current iteration of a development project.

This dashboard lets its user see several important metrics for the current iteration of an agile project, including a burndown chart showing the team's progress, the product backlog, the status of relevant work items, and recent code check-ins and builds. Most of the information in this dashboard is drawn from TFS. Much of it, including the status of project work items and recent check-ins/builds, is created automatically. Because all of these aspects of the development process are handled in a unified server, creating accurate, timely status displays is straightforward. This is one of the primary benefits of a unified tool set like Visual Studio 2010.

USING VISUAL STUDIO 2010

Team Foundation Server and the other components of Visual Studio 2010 provide a great deal of functionality. To understand what these tools offer, it's useful to walk through each of the activities in a typical development process: managing requirements; architecting a solution; developing code; testing code; and managing and tracking the project. It's also important to think about maintenance, which often consumes more money than a project's original development. This section looks at each of these areas, highlighting some of the most important aspects of this product family.

MANAGING REQUIREMENTS

Requirements are the backbone of a software development project. They drive the design and development, they determine what tests are done, and they're fundamental to deciding when the software is ready to ship. Given this central role, managing requirements effectively is important.

In Visual Studio 2010, requirements are stored as work items in TFS. The product doesn't specify how requirements should be gathered, however. One common solution is to record requirements using Microsoft Word or another tool. Third party products, such as TeamSpec from TeamSolutions, provide add-ins that allow requirements gathered in Word to be automatically synchronized with requirement work items in TFS. Another option is to use SketchFlow, a tool included with Microsoft's Expression Blend, to create quick sketches of user interfaces. Because these interface prototypes let people see what an application will look like, they can help in understanding a project's requirements.

However they're gathered, requirements stored in TFS can be used in several different ways. A primary goal of Visual Studio 2010 is to provide requirements traceability, connecting requirements with other aspects of development throughout a project's life. As mentioned earlier, requirements can be connected with other work items such as tasks and test cases to make this possible. These connections let team members do things like determine which requirements don't yet have test cases, figure out who's responsible for the tasks necessary to meet a given requirement, or decide what tests to work on today.

To work with requirements and the work items they're connected to, the people on a development team have a number of options. A developer might choose to use the Visual Studio IDE to see which bugs are associated with a specific requirement. A tester can use Microsoft Test Manager to display the tests for that requirement. A business analyst might prefer to see requirements via Excel, letting her list the tasks associated with each one, as Figure 6 shows.

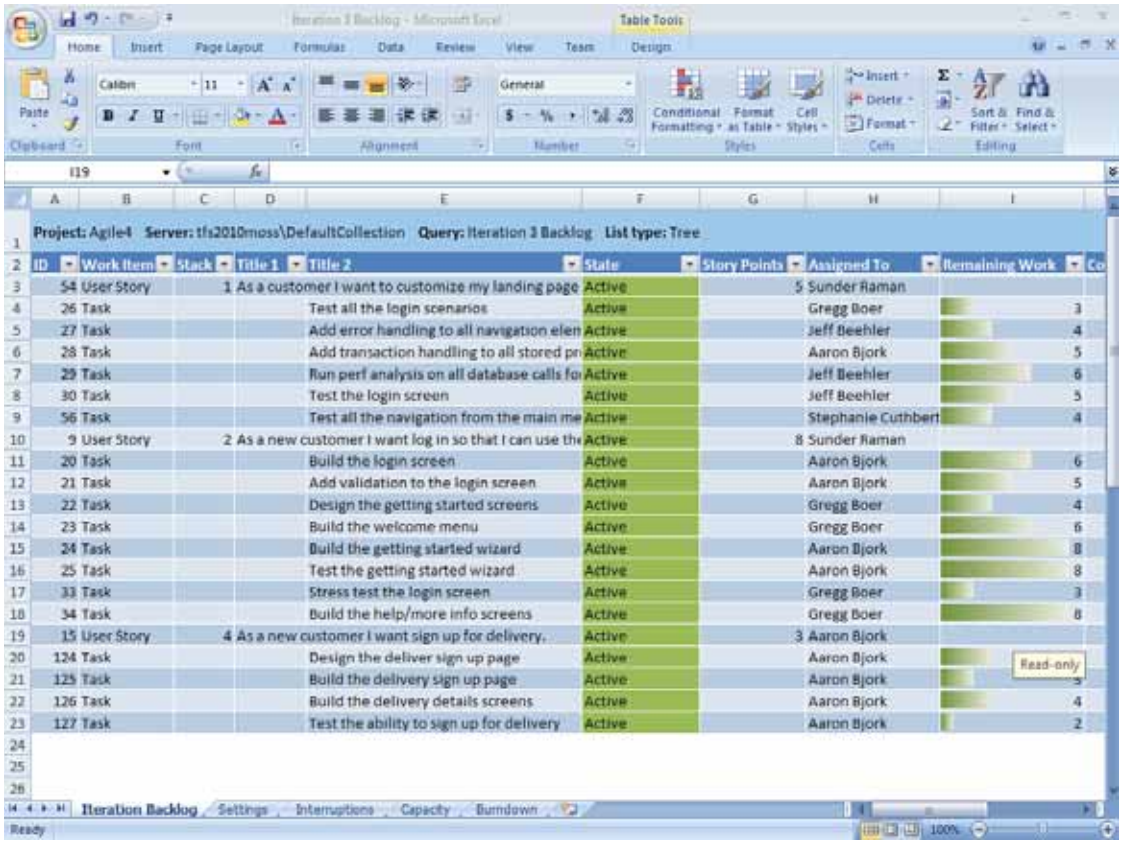


Figure 6: A business analyst might use Excel to work with requirements (user stories) and associated tasks stored in TFS.

This example shows an Excel spreadsheet containing several requirements (user stories), their associated tasks, owners, and other information. The analyst can modify these or create new ones directly in Excel, then store her changes back to TFS. The point is that different people want to work with requirements through different tools, and so Visual Studio 2010 allows them to be accessed in several different ways.

The overarching goal is to allow requirements traceability throughout the development lifecycle. By associating requirements with tasks, testing, and other parts of the process, and by making this information accessible through a variety of tools, Visual Studio 2010 aims at making it easier for teams to build the right software.

ARCHITECTING A SOLUTION

Before writing any code, the people responsible for building a new application usually start by thinking about its structure. What parts should the application have? What should each one do? And how should those parts fit together? Once code actually exists, they ask more questions. What does this class look like? What other classes is it related to? What's the sequence of calls from this method?

All of these questions lend themselves to visual answers. In every case, creating diagrams that show what's going on can be the clearest path to understanding. Accordingly, Visual Studio 2010 contains tools for creating and working with diagrams that address all of these questions.

These tools and the questions they address are sometimes lumped under the heading of “architecture”. It’s equally useful to see them as ways to visualize solutions, leading to better understanding and stronger designs. Whatever view one takes, visual tools can be useful. This section takes a look at what Visual Studio 2010 provides in this area.

Designing Code: UML Modeling

One common way to think and talk about an application’s behavior is by using the Unified Modeling Language (UML). To allow this, Visual Studio 2010 provides tools for application modeling with this popular language. These tools support five of the most common UML diagrams: Class, Sequence, Use Case, Activity, and Component.

Once they’re created, UML elements can be linked to work items in TFS. For example, an application architect might create a use case diagram, then link that use case with a work item containing the specific requirement this use case applies to. As usual, the requirement work item can then be linked with test cases, tasks, and other TFS work items. This can make it easier for the architect and other members of the development team to navigate more intelligently through the sea of information about this application and the process used to create it.

Controlling Code: Layer Diagrams

Grouping related responsibilities into clearly defined parts of the code makes sense. One obvious example of this is the division between user interface, business logic, and data in a multi-tier application. But these sharply defined boundaries aren’t useful solely for design; enforcing them also makes code more maintainable. Knowing that a change in, say, the user interface tier won’t affect the data tier eliminates one more risk in making that change.

To help define and enforce these boundaries, Visual Studio 2010 provides *layer diagrams*. An architect or developer can create a layer diagram, then associate different parts of the application with each layer by dragging and dropping a project or class file into it. Figure 7 shows a simple example.

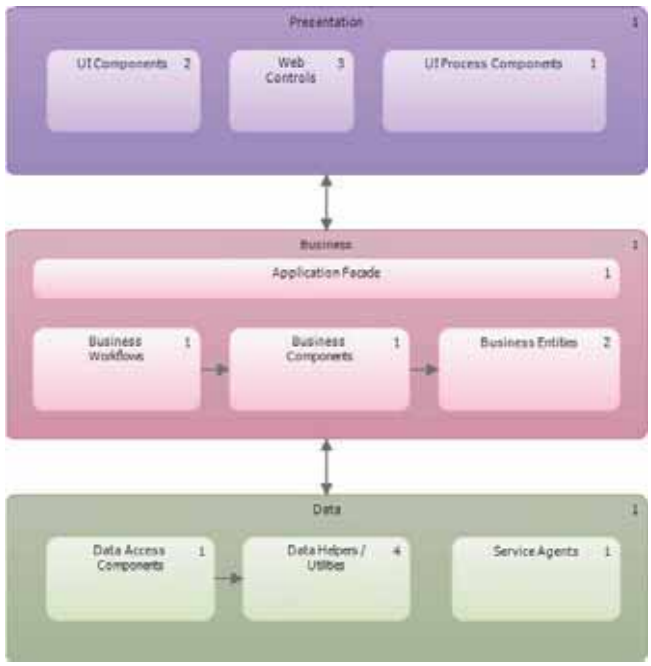


Figure 7: A layer diagram allows grouping code into layers, then enforcing rules about how the code in those layers interacts.

This diagram defines three layers: Presentation, Business, and Data. Each layer contains specific chunks of code, such as namespaces, classes, and methods. Dependencies across layer boundaries are constrained, as the figure suggests. And just to be clear: The Presentation/Business/Data split shown here is a good example, but the creator of a layer diagram is free to define any layering structure he likes. The tool isn't limited to this simple three-tier model.

These diagrams aren't just pretty pictures. A layer diagram can be used to validate the rules it contains, making sure that code conforms to the diagram's constraints. It's even possible to make this validation automatic, such as by using gated check-in to prevent checking in code that violates the layer constraints. The goal is to help maintain the application's original design, making maintenance easier and cheaper by preventing the architectural drift that can occur over time.

Understanding Code: Sequence Diagrams and More

Along with the tools just described, Visual Studio 2010 contains a few more options for working with code visually. For example, a developer can generate a UML Sequence diagram for any C# or Visual Basic (VB) method directly from source code in the Visual Studio IDE. This diagram shows what methods are called, what methods each of those methods calls, and so on, to a depth specified by the developer. These diagrams can be helpful for understanding what the code is doing and for finding ways to optimize that code.

Visual Studio 2010 also contains a Class Designer that can create a UML Class diagram directly from code, and vice-versa. Changes to the class diagram are reflected in the code, while changes to the code are reflected back to the diagram. The goal is to help developers more easily work with and understand an application's class structure.

Finally, Visual Studio 2010 contains the Architecture Explorer. As its name suggests, this tool lets a developer explore an existing .NET code base visually. While it's useful in various scenarios—imagine getting a new team member up to speed, for example—it's probably most useful during maintenance, and so it's described later in this paper.

Creating useful tools for architecture, visualization, and design isn't easy—different people want different things. (In fact, the graphical designers for modeling applications, systems, and deployment that were in Visual Studio Team System 2008 Architecture Edition have been removed in Visual Studio 2010.) The tools described here are meant to be practically useful, helping everybody on a development team create and understand their code more easily.

WRITING CODE

Once requirements have been gathered and at least some design has been done, it's time to start writing code. Visual Studio was originally created more than a decade ago to support this part of the development process, and it's still a critical aspect of what the tool family provides.

Like every IDE, Visual Studio 2010 provides a graphical interface for developers. Figure 8 shows a simple example.

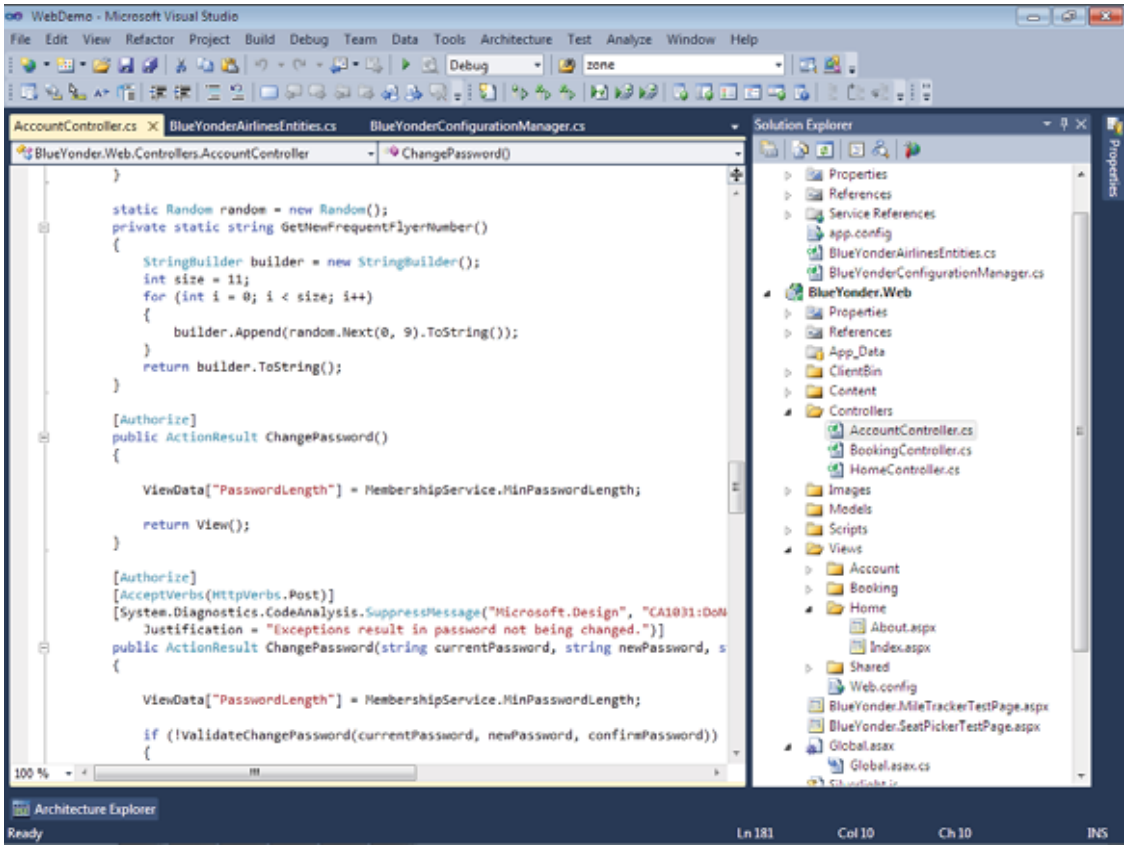


Figure 8: The Visual Studio 2010 IDE lets developers write, compile, execute, and test code.

As the figure suggests, the tool provides what a modern developer expects from an IDE, including a straightforward mechanism for managing code and configuration files, along with the ability to show different parts of the code in different colors. The Visual Studio 2010 IDE also supports using multiple monitors, with different parts of the interface shown on different screens.

This same user interface can be used to write code in any of the languages provided with Visual Studio 2010, including:

- C# and Visual Basic, languages with different syntaxes but very similar functionality. Both produce managed code, i.e., code based on the .NET Framework's Common Language Runtime (CLR).
- F#, a Microsoft-created language supporting functional and object-oriented programming. It also produces managed code.
- C++, which can produce both managed and unmanaged (i.e., native) code. Many projects in many organizations use C++, and Visual Studio 2010 contains various enhancements aimed expressly at C++ developers.
- JScript.NET, Microsoft's extended version of ECMA-262 (commonly known as JavaScript). This language produces managed code, but Visual Studio 2010 also supports creating standard JavaScript that runs in any Web browser.

Developers can also add other languages to the Visual Studio IDE. Microsoft provides IronPython and IronRuby, for example, CLR-based versions of Python and Ruby. Other vendors offer languages as well, such as COBOL from Micro Focus.

Whatever language a developer chooses, he creates a *project* to contain his work. Visual Studio 2010 provides a range of built-in project types, including projects for creating a WPF application, a Windows Service, a Class Library, various kinds of ASP.NET applications, test projects, modeling projects, and many more. One or more projects of different types can be combined to create a *solution*. For example, a solution might contain a project for an ASP.NET application containing code, a modeling project with UML diagrams describing that application, and a test project with tests for that application.

Supporting Software Development

Visual Studio 2010 allows developers to create applications for various versions of the .NET Framework, an ability that Microsoft calls *multi-targeting*. It also provides toolkits (with their own project types and more) for building software on various Windows-based platforms. They include the following:

- Microsoft Visual Studio 2010 SharePoint Developer Tools for customizing SharePoint sites or creating wholly new applications built on SharePoint.
- Office Developer Tools for creating applications that use or extend Excel, Word, PowerPoint, InfoPath, Outlook, Project, or Visio.
- Windows Azure Tools for Visual Studio, helping developers create cloud-based applications for the Windows Azure platform.

Developers can also download other software that runs inside the Visual Studio 2010 IDE, such as development tools for Windows Phone 7. Along with these toolkits, Visual Studio 2010 includes a number of specialized designers for working with various technologies, regardless of the application type that's being created. Here are a few examples:

- The WPF and Silverlight Designer, a graphical tool for creating and working with XAML user interfaces.
- The Workflow Designer for creating applications that use Windows Workflow Foundation.
- The XML Schema Designer, a graphical tool for working with XML schemas defined in XSD.
- The Object Relational (O/R) Designer, a graphical tool for creating a LINQ to SQL object model mapping for SQL Server data.
- The HTML Designer, a graphical tool for creating and working with HTML documents.

Visual Studio 2010 supports other aspects of modern software development as well. For example, as multi-core machines become ubiquitous, writing code that exploits this processing power becomes essential. To help developers do this, the Visual Studio 2010 IDE provides parallel computing libraries and debugging options for creating parallel applications.

Writing code, the most basic aspect of software development, can be hard. Writing good code is even harder, and so Visual Studio 2010 provides a number of additional features to help developers improve their code. Among them are these:

- Support for *refactoring*, which allows improving the structure, readability, and quality of code without changing what that code does.
- *Static code analysis*, which examines code for compliance with the Microsoft .NET Framework Design Guidelines or other custom-defined rules. For example, this analysis can help warn developers when they've left security holes that allow SQL injection attacks and other threats.
- *Dynamic code analysis*, including *performance profiling* and *code coverage*. Performance profiling lets a developer see how a running application (including parallel applications) divides its time across its methods, track the application's memory usage, and more. Code coverage shows what parts of an application's code were executed by a specific test, allowing the developer to see what's not being tested.
- *Code metrics*, a set of measurements calculated by Visual Studio 2010. They include simple things like the number of lines of code, along with more complex metrics such as cyclomatic complexity and a maintainability index. Since complex code is both harder to maintain and more likely to contain errors, having an objective measure of complexity is useful.

Supporting Database Development

Software development is more than just writing application code. Working with a database is also an important part of most projects. As always, tools can make this work easier.

The Visual Studio 2010 IDE allows creating database projects for working with SQL Server, much like the projects used for Windows applications. The tool also includes project types for creating applications that use the SQL CLR. And to help developers create databases and the code that uses them, the Visual Studio IDE includes visual designers for tables, queries, views, and other aspects of database development.

Like any other projects, these database projects can use the source code control and build management functions provided by TFS. For instance, a database project can help manage changes to a database's schema by putting it under version control. Visual Studio 2010 also provides support for tracking dependencies between database objects, refactoring database objects, database deployment, and other aspects of database change management.

Supporting Developer Testing and Debugging

Along with writing code, every developer does testing and debugging. It's common today, for example, for a developer to create *unit tests* for the code she writes. Each unit test runs against a specific component, such as a method, verifying one or more assumptions about the behavior of that component. Unit tests are commonly automated, which lets a group of unit tests be run easily whenever changes are made. (In fact, the build verification tests run by Team Foundation Build are most often unit tests.) To support this, Visual Studio 2010 provides a unit testing framework. It's also possible to use other unit testing frameworks with Visual Studio 2010, such as NUnit, although much of the integration with the rest of this product family is lost.

As just described, many applications also rely on logic embedded in a database, such as triggers and stored procedures. Like code running outside the database, this logic can benefit from unit tests. To help do this, Visual Studio 2010 provides support for creating *database unit tests*. The product also includes data generators to help create realistic test data for a database or to sanitize existing data by removing sensitive or private information.

Visual Studio 2010 provides other support for testing by developers as well. For example, it's increasingly common to write tests before all the code those tests will exercise has been written. Doing this typically requires generating stubs, something that Visual Studio 2010 can do automatically with an option called *Generate from Usage*. A developer can ask the tool to generate a stub for a class, method, field, or other identifier, making it easier to use a test-first development style.

An essential part of creating good code is having effective tools for debugging. The Visual Studio 2010 IDE includes a debugger, of course, but the product also provides a technology called *IntelliTrace*. Using this option, it's possible to create a trace of an application either during testing (as described later) or by running the application directly from the Visual Studio IDE. This trace provides a series of very detailed snapshots of the running application that the developer can load into the debugger, then use to replay its execution. It's even possible to move back and forth in the running application, getting a detailed look at exactly what's happening throughout. The goal is to provide a clear view into the code as it runs, making it easier to identify what's wrong.

A developer can also get other information about his code while it's being tested. For example, he can determine which tests need to be run to verify changes he's made, something known as *test impact analysis*. Rather than running a complete test suite after every change, this knowledge lets the developer run just those tests that cover the code he's modified.

The information TFS holds about a project—requirements, tasks, bugs, and everything else—needs to be accessible by every member of the development team, regardless of the tools they use. To let developers work with this information in a natural way, Team Explorer can run inside the Visual Studio 2010 IDE. Figure 9 shows an example of how this looks.

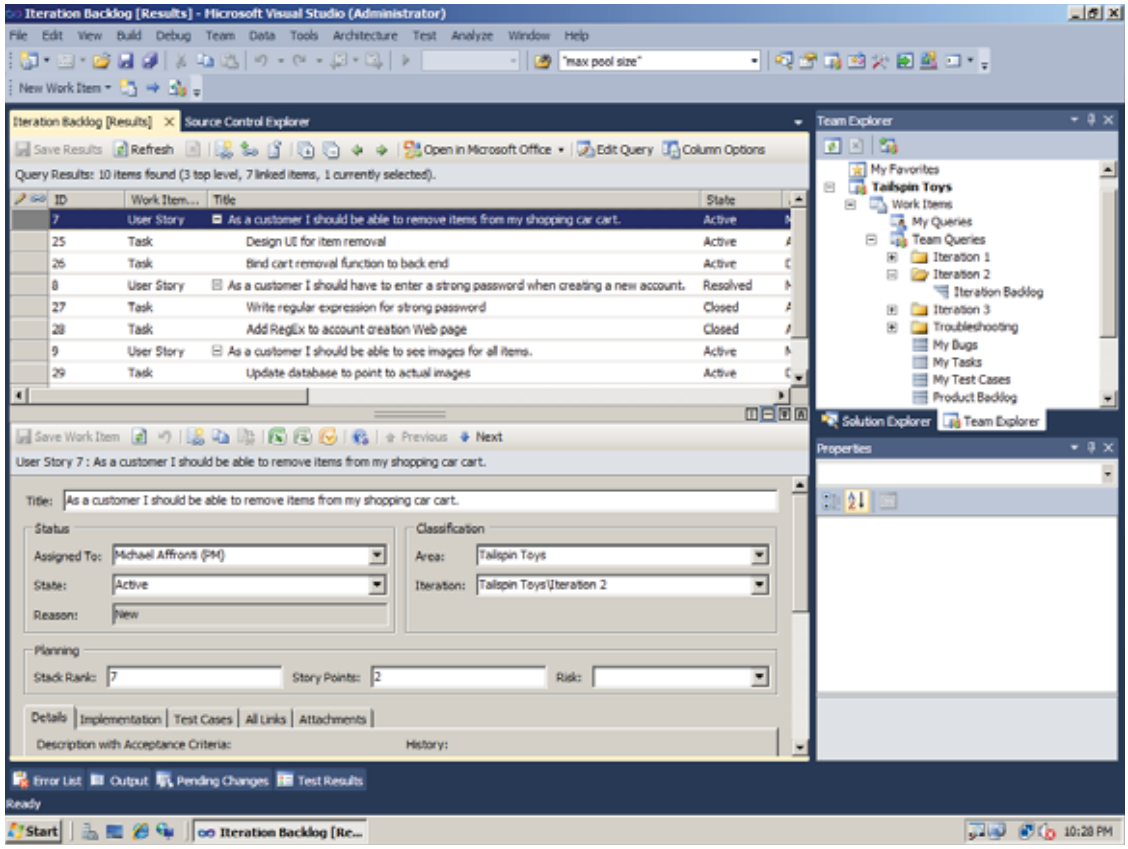


Figure 9: Using Team Explorer from inside the Visual Studio 2010 IDE lets developers work with requirements, tasks, bugs, and other TFS information.

In this example, current requirements (user stories) are shown at the top, along with the tasks associated with each one. User story 7 has been selected, and so its details are shown below: who it's assigned to, the state it's in, and other information. The pane in the upper right shows other TFS data that the developer can access under headings such as "My Bugs", "My Tasks", and "My Test Cases". The goal is to provide a useful window into the diverse information available in TFS about this development project.

TESTING CODE

Testing is an essential part of the development process. There's lots to do and plenty of information to keep track of. Some tests, such as unit tests, are typically created by developers. Yet much of testing is done by dedicated testers. Testing is a discipline in its own right, and testers play a critical role in most development teams.

Some testers are essentially developers who focus on testing. In many cases, the tests these developers create are automated, making it easy to run them over and over as code changes. Yet while automated testing is important, it's not always the right solution. Writing test code costs money, so a test that's run infrequently might not be worth automating. (In fact, seeing a positive return on the investment in creating an automated test suite might take years.) And what if the application is changing frequently? Constantly rewriting automated tests to match volatile code might not be worthwhile.

Given these realities, manual testing—exercising an application through its user interface—will always remain an important part of the story. Besides, really testing the user experience is hard to do with automated tests. Manual testing is often more effective at finding the bugs that bother users the most.

Providing solid support for both automated and manual testing is a primary goal of Visual Studio 2010. In both cases, effective test tools can make the job easier and more efficient. In Visual Studio 2010, these tools include the following:

- Ways to gather the results of running tests, including diagnostic data to help determine the cause of failures.
- A tool for creating and managing test plans.
- Software for creating, configuring, and deploying virtual machines for use in a test environment.
- Support for manual testing, including things such as the ability to record and automatically play back a manual test.
- Support for automated testing, including load tests.

All of this technology can be used together, or pieces can be used on their own. Whatever approach an organization chooses, the place to begin understanding this world is with the basics: how test results and diagnostic data are gathered.

Gathering Test Results and Diagnostic Data

The most fundamental aspect of testing is running a test, then getting the results. This can seem mundane, but it's not. Determining when a test has failed might be straightforward, but what information should the tester provide to the developer to help fix the bug this test has discovered? Just indicating that the test failed isn't enough—the developer might not even believe the tester, especially if the developer can't reproduce the bug on his own machine. What's needed is a way for the tester to supply enough information for the developer to understand the bug, then figure out and fix the underlying problem.

To do this, the test environment needs to provide mechanisms for gathering a range of information about the application under test. Visual Studio 2010 does this with *diagnostic data adapters (DDAs)*. Figure 10 shows how they fit into the test environment.

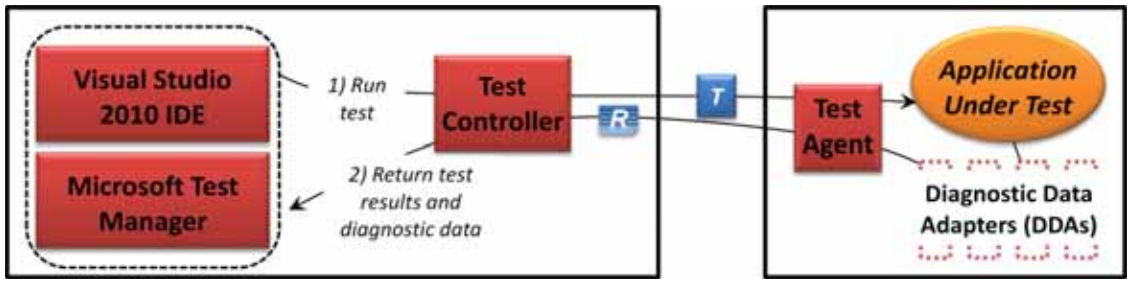


Figure 10: When running a test from the Visual Studio IDE or Microsoft Test Manager, a tester can rely on one or more diagnostic data adapters to collect data about that test.

As the figure shows, a test might be run either from the Visual Studio 2010 IDE or from Microsoft Test Manager (step 1). Here, the test is sent to a *test controller* which then sends it to a *test agent* running on the same machine as the application under test. As Figure 10 shows, this application is monitored by one or more DDAs while it's being tested. The diagnostic data those DDAs produce and the test results are returned to the tester (step 2).

Each DDA collects a specific kind of information, and the tester can select which DDAs are used for a particular test. Some examples of DDAs provided with Visual Studio 2010 are the following:

- Action Recording: Produces a recording containing all of the steps executed in a manual test. This log can be played back to re-run the test or used in other ways.
- ASP.NET Profiler: Provides profiling data for ASP.NET applications, such as a count of how much time was spent in each method.
- Event Log: Collects information written to event logs during the test. It can be configured to collect only specific event logs and event types.
- IntelliTrace: Creates a detailed trace of the application's execution. As described earlier, a developer can use this trace to replay the execution of a test in the Visual Studio 2010 IDE's debugger, providing a window into exactly what was happening when a bug appeared.
- Test Impact: Keeps track of which methods in the application were invoked by a test case. This DDA provides the raw material for the test impact analysis described earlier
- Code Coverage: Provides statistics about what percentage of the code was covered by one or more tests. As its name suggests, it provides the base information used by the code coverage option described earlier.
- System Information: Provides a description of the machine on which the test is run, including its CPU, memory, installed patches, and more.
- Video Recorder: Records a video of a computer's desktop on which a test is being run.

Different DDAs make sense for testing different parts of an application. For example, the Video Recording DDA makes sense when testing an application's user interface, while the ASP.NET Profiler (obviously) is useful while testing ASP.NET code on a server. Other DDAs, such as IntelliTrace, Code Coverage, and Test Impact, make sense for testing business logic in general, and System Information should probably be

turned on for every test. If the options supplied with Visual Studio 2010 aren't sufficient, it's also possible to create custom DDAs. In fact, a variety of DDAs are available from third parties, including options for non-Windows environments.

Whatever DDAs are used, the goal is the same: providing enough data about a bug so that a developer can resolve the issue. Rather than having testers find bugs in the test environment, then have developers be unable to reproduce those bugs, the detailed information collected by test agents and DDAs can be filed with the bug report in TFS. Providing this fine-grained data significantly increases the likelihood of a developer recreating and fixing a bug the first time it's found by a tester.

A Tool for Testers: Microsoft Test Manager 2010

Running tests from the Visual Studio IDE is fine for development-oriented testers. For many testers, however, a developer tool isn't the best option. To support these people, Visual Studio 2010 includes Microsoft Test Manager (MTM). This tool is designed explicitly for testers, especially those who don't need to edit code. One obvious indication of this is the MTM user interface: It's not based on the Visual Studio IDE. Instead, the tool provides its own interface focused explicitly on the tasks it supports. Figure 11 shows an example.

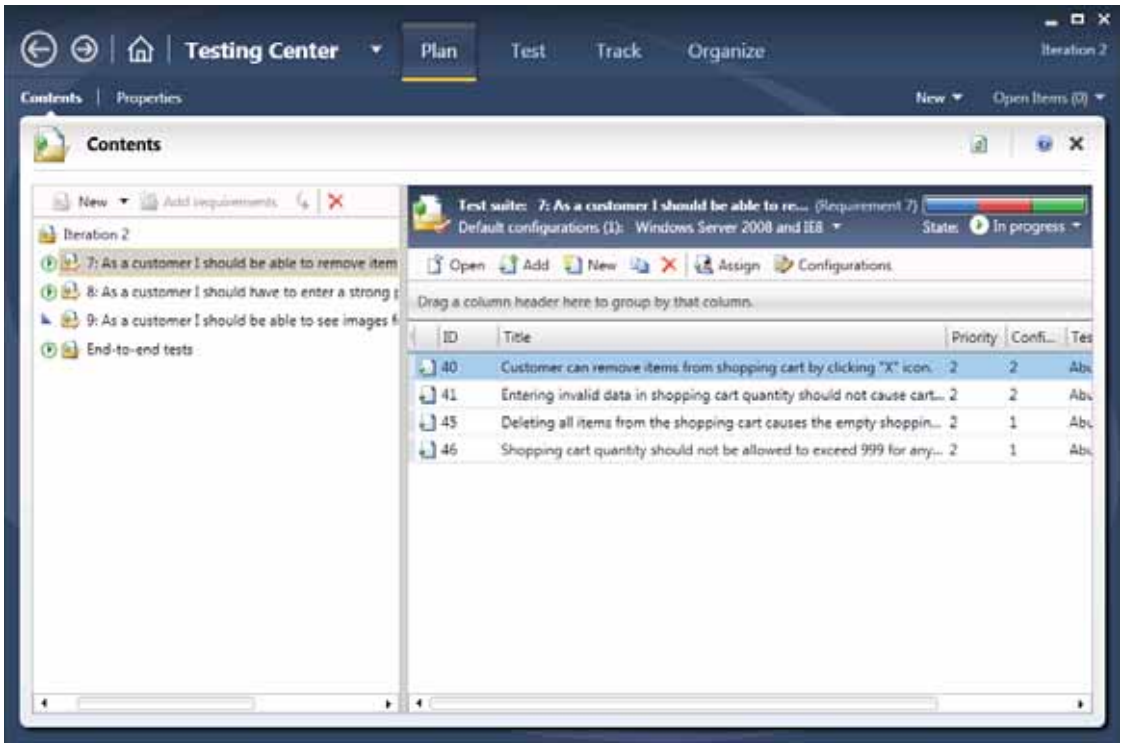


Figure 11: Microsoft Test Manager 2010 is designed expressly for testing—it's not a development tool.

MTM supports two distinct activities. One is acting as the client for Visual Studio Lab Management 2010, described in the next section. The other, unsurprisingly, is managing and running tests. Toward this end, MTM lets a tester define and work with *test plans*. A test plan consists of one or more *test suites*, each of which contains some number of automated and/or manual test cases. A test plan can also specify the

exact configuration that should be used for the tests it contains, such as specific versions of Windows and SQL Server. All of this information is stored in TFS using the Test Case Management functions mentioned earlier.

Test suites can be used in various ways. For example, each of an application's requirements (or user stories) might have its own test suite aimed at testing just code that addresses that requirement. In Figure 11, for instance, the highlighted requirement is associated with the test suite shown on the right.

MTM also lets a tester examine the output created by running tests. These results are associated with a specific test plan, allowing the output from all the test cases in that plan to be grouped together. And once again, this information is stored in TFS, making it available to the development team and other interested parties.

Managing Test Lab VMs: Visual Studio Lab Management 2010

Testing software requires machines on which to run that software. Those machines must replicate as closely as possible the eventual production environment. Traditionally, testers have done their best to replicate this environment with the physical machines on hand. While this approach can work, it's often simpler and cheaper to use virtual machines instead. VMs are easier to create and to configure into exactly what's required. Think about testing a multi-tier application, for example, where the user interface, business logic, and database all run on separate machines. Realistic tests for this application require three machines, something that's usually easier to do with VMs.

To help organizations create and manage VM-based test labs, Visual Studio 2010 includes Visual Studio Lab Management. This software is accessed via MTM's Lab Manager activity, as Figure 12 shows.

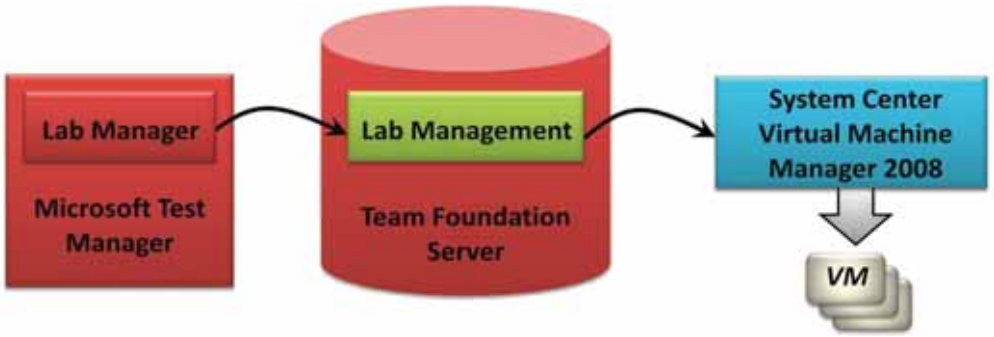


Figure 12: Visual Studio Lab Management 2010 allows creating and managing a VM-based test and development lab.

As the figure shows, Lab Management relies on System Center Virtual Machine Manager (VMM) 2008, which is included with this part of the product. VMM allows defining templates for virtual machines, then creating multiple Hyper-V-based VMs from a single template. Using Lab Management, a tester can create templates for each of the hardware and software configurations an application must run in, then create VMs from these templates as needed. A template might be pre-configured to contain the software a particular application depends on, such as SQL Server or Internet Information Services (IIS). It might also be configured to contain a test agent, making VMs created from it easier for a tester to use.

Putting the Pieces Together: A Testing Scenario

To see how all of these parts can be used together, it's useful to walk through a scenario. Figure 13 shows the first steps.

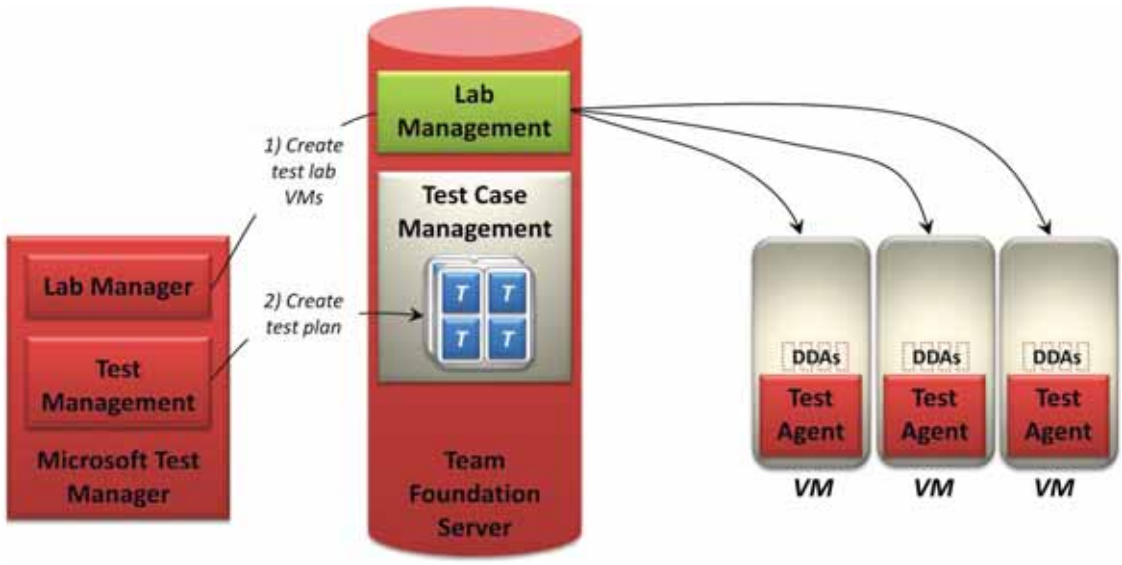


Figure 13: A tester can use Lab Management to create VMs for testing an application, then define a test plan for that application with Test Case Management.

This example begins with a tester using the Lab Manager activity in MTM to request new VMs from Visual Studio Lab Management (step 1). The templates used to create those VMs are pre-configured to contain a test agent, which means they're ready to be used for testing. The software being tested is a Web application, so three separate VMs are created, one for each tier of the application. Next, the tester uses MTM to create a test plan, storing it in TFS Test Case Management (step 2). Once the test plan is ready, testing can begin. Figure 14 illustrates this part of the process.

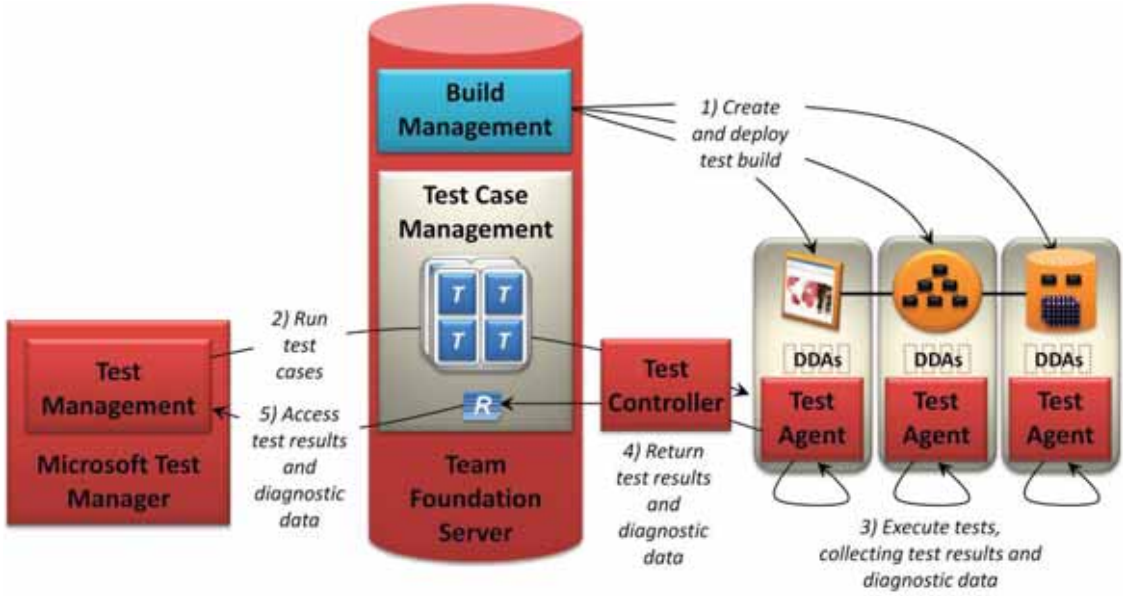


Figure 14: Once a new build is deployed, a tester can run test cases from MTM, then access the results and DDA-generated diagnostic data produced by those tests.

Since the goal is to test a specific build of the application, that build must first be created and deployed (step 1). As the figure suggests, Team Foundation Build can deploy a new build to the VM-based test environment as part of an automated build process. The tester can then use MTM to run test cases from the test plan (step 2) against this build. Each test is sent to a test controller, which distributes it to the test agents. These test agents run each test, using appropriate DDAs to gather diagnostic data (step 3). As described earlier, different DDAs are probably used in each of the three VMs, since each is testing a different part of the application. The Video Recording DDA might be used in the VM running the application's user interface, for example, the IntelliTrace, Code Coverage, and Test Impact DDAs might be used in the VM running the middle -tier business logic, and just the Event Log DDA might be used in the VM running the database. When the tests are completed, the test results and diagnostic data are sent back to TFS via the test controller (step 4). The tester can now use MTM to access and examine this information (step 5).

When a bug is found, the tester can submit a report directly from MTM. A description of the issue, together with whatever diagnostic data the tester chooses, is stored in TFS for a developer to use. When Lab Management is used, as in this example, it's even possible for a tester to submit a bug report that links to a snapshot of the VM in which the bug was detected. Because the bug report contains so much supporting information, the developer responsible for the code being tested is significantly more likely to believe that the bug is real. Just as important, that developer will have what she needs to find and fix the underlying problem.

In this scenario, the tests run in step 2 might be either manual or automated. Both testing styles are important, and so Visual Studio 2010 provides support for both. The next sections look at each approach, beginning with manual testing.

Supporting Manual Testing

Manual testing, where someone sits at a screen exercising an application, is an inescapable part of software quality assurance. This kind of testing might be done in an exploratory fashion, or it might require the tester to follow a rigidly specified set of test scripts. In either case, a large part of an application's tests are often manual.

A primary purpose of Microsoft Test Manager is to support manual testing. Along with the user interface shown earlier for test management, MTM also provides an interface for running manual tests. Sometimes referred to as *Test Runner*, an example of this interface is shown in Figure 15.

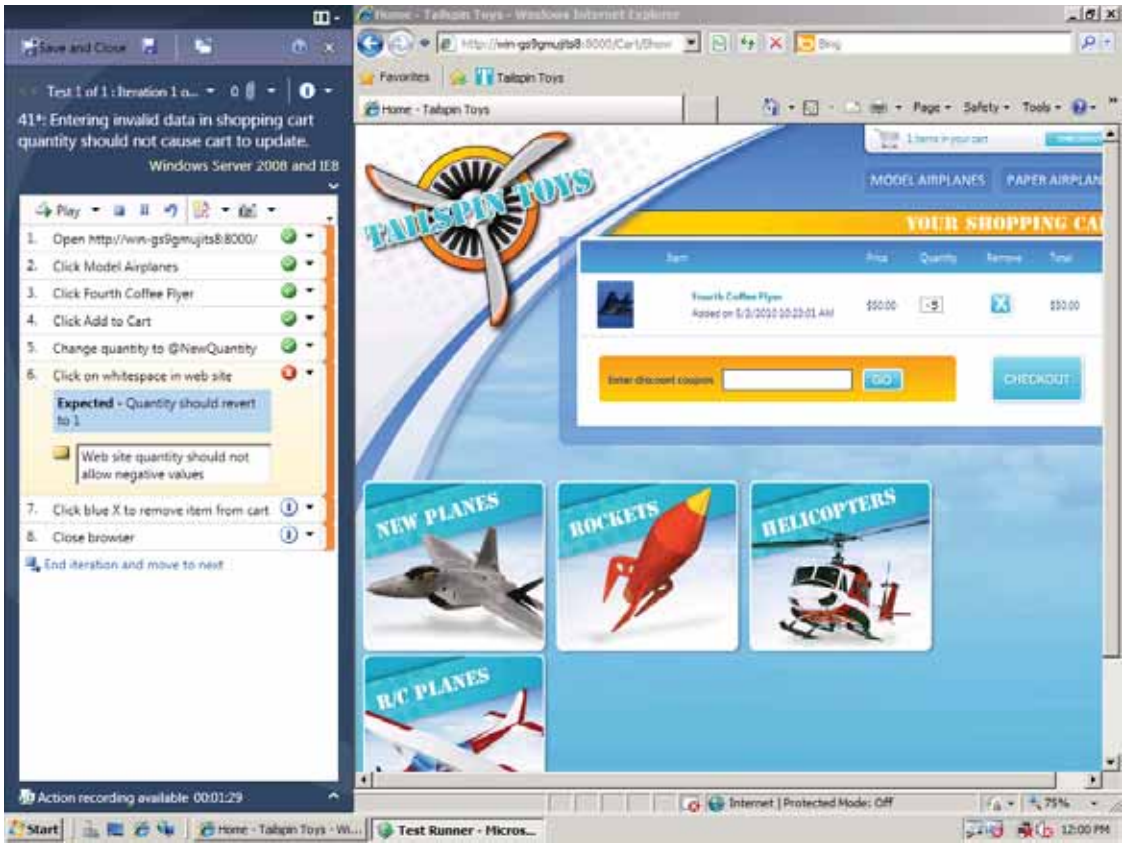


Figure 15: Microsoft Test Manager, shown docked to the left of the application under test, provides a user interface for running manual tests.

The UI of the application under test is shown on the right, allowing the tester to interact with it. The pane on the left lists the steps in the manual test currently being performed. The green circles in this pane mean that the tester has marked this step as successful. A red circle appears at step 6, however, indicating that the tester believes this step has failed. The tester can now file a bug report directly from Test Runner. And because manual tests use the Visual Studio 2010 testing infrastructure described earlier, any of the diagnostic data provided by the DDAs, including IntelliTrace, can be included with this bug report.

Some DDAs are especially useful with manual tests. The Action Recording DDA, for instance, creates a log of the actions performed in a test. This action log contains everything the tester actually did, including details such as mouse hovers. This fine-grained information lets a developer see exactly what steps the tester went through and so can be quite useful in replicating the bug. The action log can be included as part of a bug report, perhaps accompanied by a video of the tester's screen.

An action log can also be used sometime later by the tester to automatically step through just part of the test. This option, referred to as *fast forward for manual testing*, lets a tester (or a developer) move quickly to a later part of the test rather than laboriously working through every step to get where she needs to be. For example, a tester who wishes to test an application's behavior deep into a series of screens can rely on this facility to zip quickly through the navigation required to get there.

Supporting Automated Testing

While manual tests are important, automated testing—software that tests other software—is also useful. Automated tests can only be created with the Visual Studio IDE—they can't be created using MTM. They can be run in various ways, however: from the IDE, from MTM, or by Team Foundation Build as part of the build process.

Because there are a variety of things to be tested, Visual Studio 2010 has built-in support for several types of automated tests. Figure 16 illustrates some of the most important options.

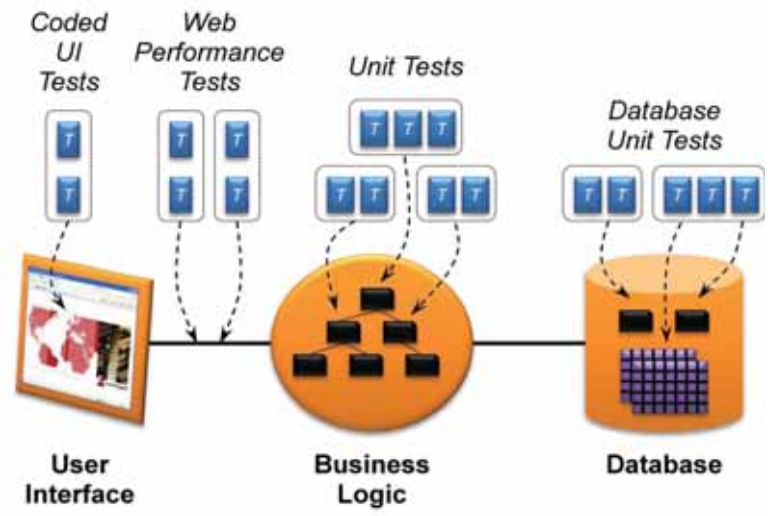


Figure 16: Visual Studio 2010 supports several kinds of automated tests.

The kinds of automated tests supported by Visual Studio 2010 include the following:

- *Unit tests*: As described earlier, a unit test verifies specific behavior in a particular component of an application. Unit tests are typically created by the developer who writes the code being tested.
- *Database unit tests*: These are unit tests aimed specifically at aspects of an application's database, such as a stored procedure. They're also typically created by developers.

- *Web performance tests*: An effective way to test the functionality of a Web application is to send HTTP requests directly to the application's business logic. This is exactly what Web performance tests do. Typically built by testers, these tests can be defined manually using Visual Studio 2010's *Web Test Editor*, or (more likely) created automatically by using the product's *Web Test Recorder* to record HTTP requests made from a browser.

- *Coded user interface (UI) tests*: Sometimes referred to as *UI test automation*, a coded UI test is software that executes actions directly against an application's user interface, then verifies that the correct behavior occurs. Coded UI tests are typically created by a tester recording the manual actions she takes against an application's UI, then letting Visual Studio 2010 generate code to replicate those actions. The product also provides support for validating a test's result, such as checking that a particular text box in the UI should contain a specific value.

There's one more important category of automated testing to describe: load tests. A load test is intended to simulate how an application performs when it has many users, and it's typically composed of a group of Web performance tests. Figure 17 illustrates how Visual Studio 2010 provides load testing.

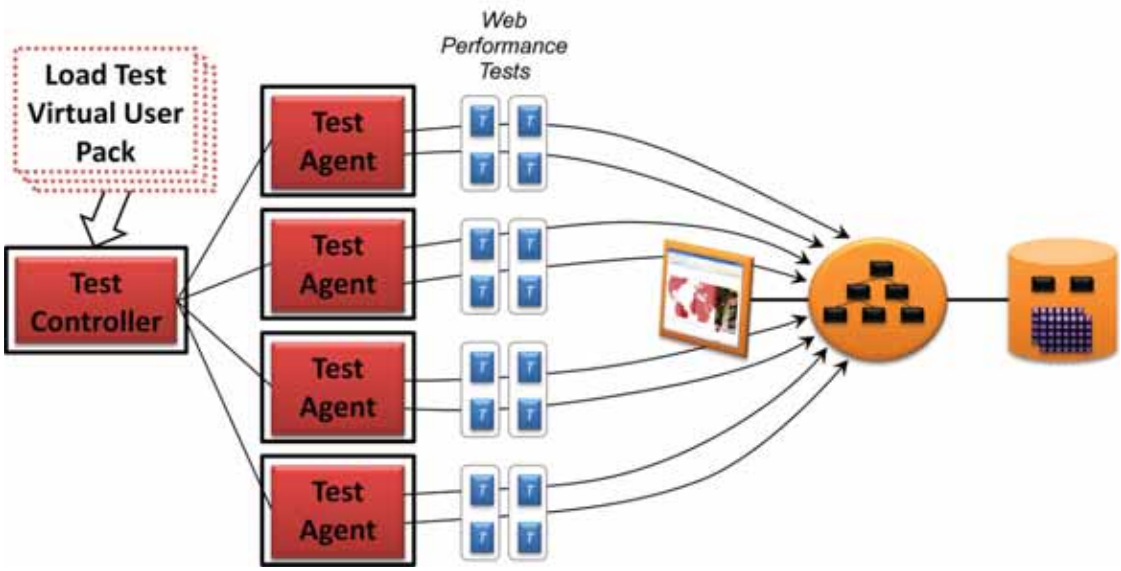


Figure 17: In load testing, multiple test agents submit Web performance tests against an application's business logic, simulating the behavior of many simultaneous users.

As the figure shows, load testing relies on a test controller and one or more test agents, each of which runs on its own physical machine. The controller parcels out Web performance tests among the test agents. Every test agent then submits its tests to the application. The result helps the tester understand how this application will behave under the load created by many users.

To simulate large numbers of users, an organization can buy one or more copies of the Visual Studio Load Test Virtual User Pack 2010. Installed on the test controller, as Figure 17 shows, each pack represents 1,000 virtual users. If a load test needs to see how an application behaves with 3,000 users, for example, the tester needs to install three Load Test Virtual User Packs.

Whether it's manual or automated, testing is a critically important part of creating applications. By providing a common foundation for both that provides lots of information about bugs, Visual Studio 2010 aims at improving the quality of applications. Not incidentally, it also aims at improving the lives of the people who test those applications.

MANAGING AND TRACKING A PROJECT

Whatever stage a project is in, the people involved need to know what's going on. Perhaps a project manager is concerned with keeping the work on schedule, or maybe the project is using Scrum, with the ScrumMaster playing a coordinating role on a self-organizing team. Whatever the situation, the people involved must manage and track the work. As already described, Visual Studio 2010 provides reports, dashboards, and more aimed at doing this. It's worth taking a closer look at two specific areas, however: the beginning of a project and its end.

Pretty much every project (and every iteration) starts with planning. Plans are typically based on requirements, which are stored in TFS. A project manager might use Microsoft Project to read those requirements directly from work items. She can then use this tool to plan a schedule for the effort, as Figure 18 shows.

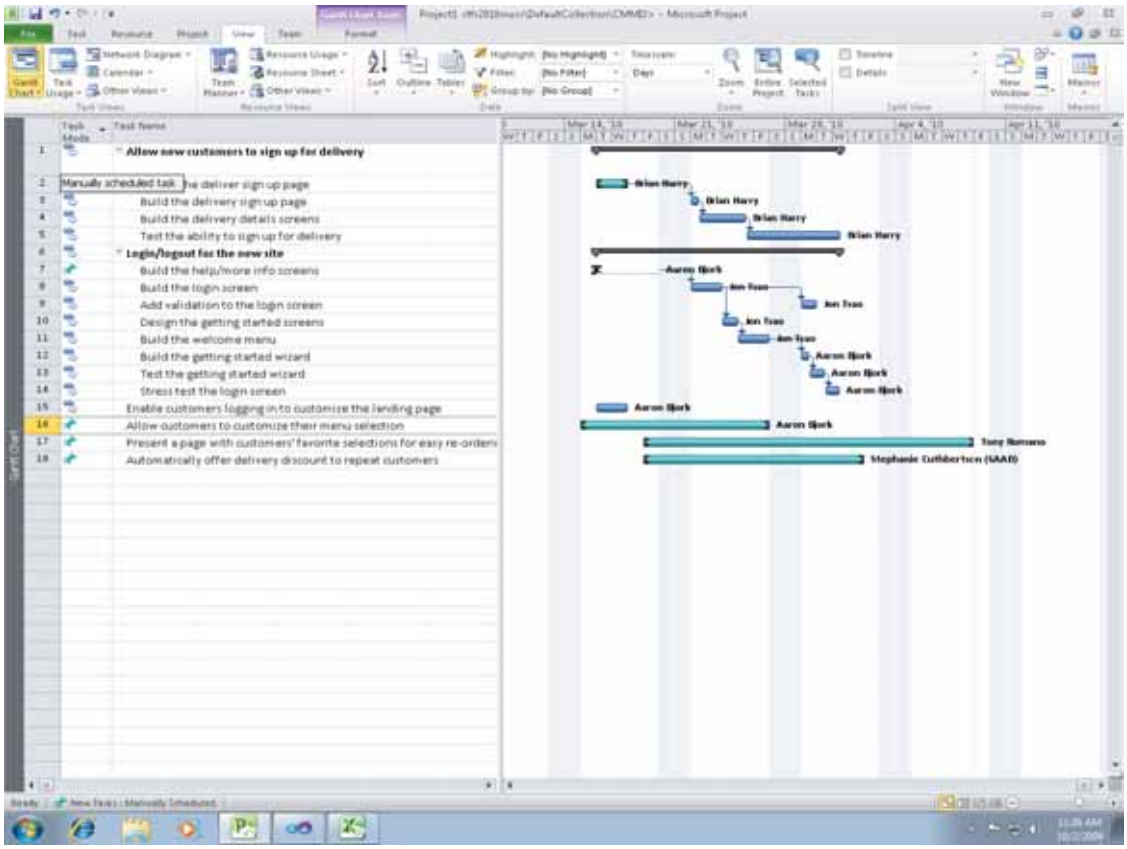


Figure 18: Microsoft Project can read work items such as requirements and tasks directly from TFS, then let a project manager construct a schedule.

The project manager can use familiar techniques, such as the Gantt chart shown here, to plan the project's schedule, with the requirements and tasks drawn directly from TFS. For project managers who prefer Excel, Visual Studio 2010 provides other options. For example, the Agile process template includes an Excel workbook designed expressly for planning an agile project and the iterations (sprints) within that project. As always, the information this workbook uses is synchronized with TFS.

Projects start, and projects eventually end. Assuming a project ends successfully, the software must be handed over to the customer. But when is it ready to ship? How can a team know when it's time to release their work? One obvious metric is fulfilling enough of the project's requirements, but another is quality, something that's harder to judge accurately. To help with this, both the Agile and CMMI process templates provide a Quality dashboard that presents relevant information from TFS. Figure 19 shows an example.

Figure 19: The Quality dashboard gives a view into several quality-related metrics for a project.

This dashboard provides a window into important metrics for project quality over a specific period, such as the previous month. Those metrics include the following:

- Test Plan Progress: Shows the team's progress in running test cases defined in test plans created with Microsoft Test Manager.

- Build Status: Shows the number of builds that succeeded or failed.
- Bug Progress: Shows the number of bugs, broken down by status: Active, Resolved, or Closed. Shippable code quality needn't mean zero bugs, but a continually rising count of active bugs doesn't bode well for quality.
- Bug Reactivations: Shows how many bugs that were previously marked as Resolved or Closed have been reactivated. If this number is rising, the quality of the team's bug fixes probably isn't very high.
- Code Coverage: Shows the percentage of code tested by build verification tests and others.
- Code Churn: Illustrates how many lines of code were added, deleted, and changed in check-ins to TFS version control. This is another useful quality measure, since the total should be heading down as a project nears its end.

Using this dashboard, project managers, ScrumMasters, and others—even the customer—can better understand the team's progress. Rather than seeing individual bits of data in isolation, they can see the totality of a project's state and thus make better decisions. This information can also help in determining when the product they're building is ready to ship. Different projects have different standards: A team building a community Web portal probably has a lower quality bar than, say, one building an online ordering system for a major retailer. Yet whatever the group's risk tolerance, they need data to make a good decision. The Quality dashboard is one example of how Visual Studio 2010 provides this information.

MAINTAINING CODE

Releasing the results of a development project doesn't mean that there's no work left to be done. If the software that's been created has any value, it will stick around for a while, and it's bound to require changes. Maybe the people who use it have ideas for improvement, or perhaps the business process it's part of needs to change in some way. Whatever the reason, applications require maintenance.

To a great degree, the tools used for development are also appropriate for maintenance. Still, there are unique challenges that show up in this part of an application's lifecycle. For example, the people who do maintenance work often aren't the original developers. A fundamental challenge for these people is to understand the application's code base. The conventional way to do this is by brute force: just sit down and read the code. Yet understanding the complicated interactions and dependencies in even a moderately sized code base can be very difficult. Besides, we are visual creatures—why not use a graphical tool to help?

Toward this end, Visual Studio 2010 includes the Architecture Explorer. This tool gives its user a window into the structure of existing C# and Visual Basic code. By generating *dependency graphs* that show how various parts of an application work with each other, the Architecture Explorer can help developers and architects understand what the code is doing. Figure 20 shows an example.

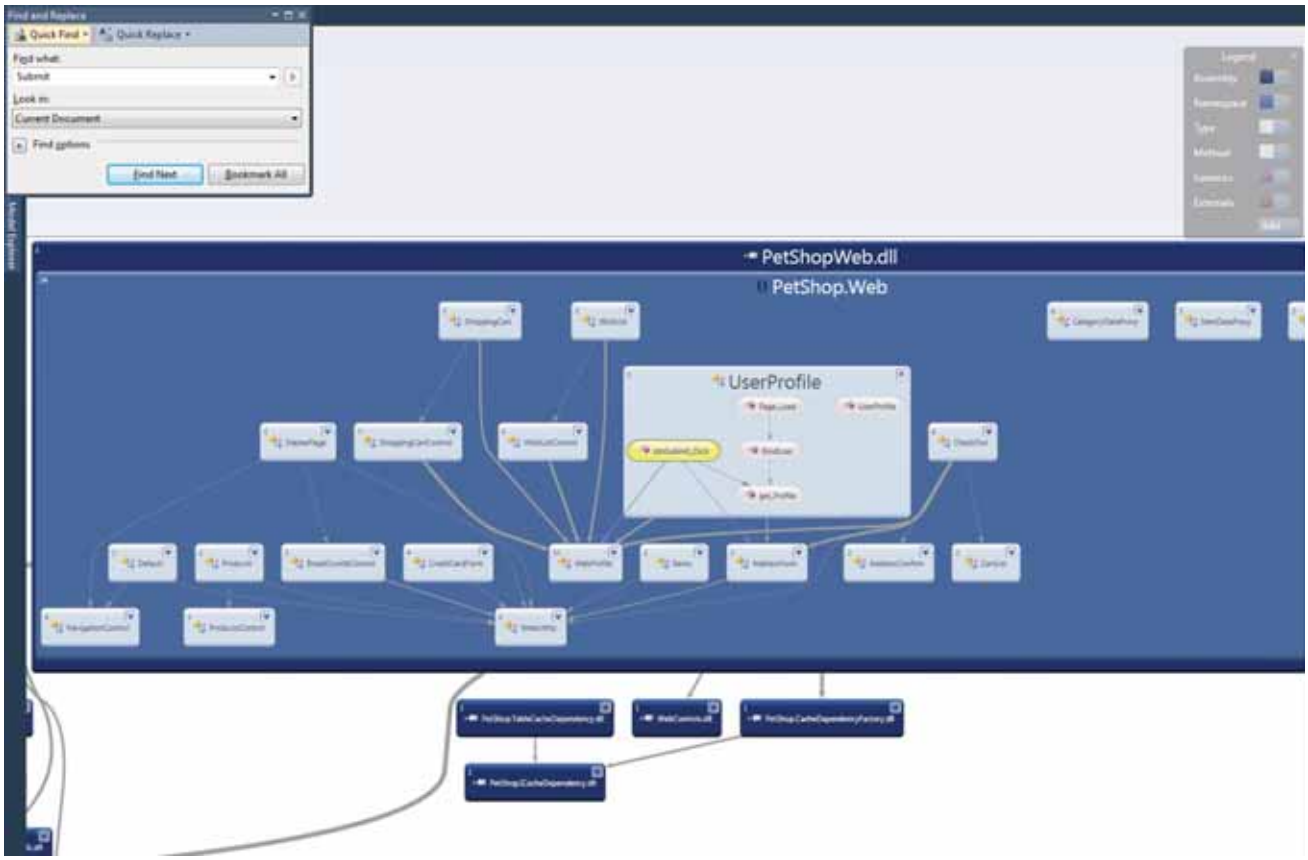


Figure 20: A dependency graph illustrates the relationships among different parts of an application's code.

This example shows a dependency graph focused on an assembly called PetShopWeb.dll. This assembly contains the namespace PetShop.Web, which itself contains a number of other types. The connections between these types are shown as gray lines, with the thickness of each line representing the number of interactions between each pair. As the dialog box in the upper left shows, the user has searched for methods whose name contains the word “Submit”, and the Architecture Explorer is showing one match: a method in the UserProfile type.

As the diagram suggests, the tool is interactive. A user can zoom in and out of the dependency graph, choosing different granularities. For example, a developer new to this application can get a broad view by starting at the assembly level, then zoom in to examine a particular assembly in detail (which is the situation shown in Figure 20). She might then zoom further in to look at a particular class and the other classes it depends on. Having this understanding of dependencies can help mitigate the risk of making a change to the code, since it’s easier to trace the impact a change might have.

While it’s always possible to acquire the same knowledge by reading the source code, starting with a visual approach is likely to speed up the process. And given the importance—and the cost—of maintaining software, providing tools to help makes sense.

ADOPTING VISUAL STUDIO 2010

Visual Studio 2010 has many pieces, and different development teams will choose different configurations. Adopting this tool set requires understanding at least the basics of how Microsoft packages the product. It also requires thinking about how to migrate to this new world from what's already in place. This section looks at both of these issues.

DECIDING WHAT TO BUY

As is probably obvious by now, Visual Studio 2010 contains lots of functionality. This functionality is divided across several different software components, as was shown back in Figure 1. Microsoft groups these components into a variety of different products, commonly referred to as *stock-keeping units* (*SKUs*). Figure 21 lists the primary SKUs for Visual Studio 2010, showing the components in each one.

	Visual Studio Team Foundation Server	Visual Studio IDE	Visual Studio Team Explorer	Microsoft Test Manager	Visual Studio Lab Management	Visual Studio Team Explorer Everywhere
Visual Studio 2010 Ultimate*	X	X	X	X		X
Visual Studio 2010 Premium*	X	X	X			
Visual Studio 2010 Professional*	X	X	X			
Visual Studio Test Professional 2010*	X		X	X		
Visual Studio Lab Management 2010					X	
Visual Studio Team Explorer Everywhere 2010						X

*With MSDN subscription

Figure 21: Each SKU for Visual Studio 2010 includes a specific set of components.

Some comments on this diagram:

- While TFS is shown in the figure as part of three Visual Studio 2010 IDE SKUs (Professional, Premium, and Ultimate) and Test Professional, it's actually provided in the MSDN subscription. It's possible to buy the IDEs without this subscription (and thus without TFS) or to buy TFS on its own.
- Recall that Lab Management is implemented as part of TFS and that it requires the Lab Management client contained in Microsoft Test Manager. Because of this, the Lab Management 2010 SKU must be used together with both TFS and a SKU that includes MTM. Since publishing this white paper Microsoft has altered the licensing terms for Lab Management. The new licensing gives Lab Management to all users of Visual Studio 2010 Ultimate with MSDN or Visual Studio Test Professional 2010 with MSDN

- Visual Studio 2010 has a few SKUs that aren't shown in this figure. For example, while Visual Studio 2010 Ultimate includes 250 virtual users for load testing, an organization that needs more than this will need to purchase one or more SKUs for Visual Studio Load Test Virtual User Pack 2010.

- The Project dashboard shown in Figure 5 is available with Windows SharePoint Services 3.0 or its successor, Microsoft SharePoint Foundation 2010. Other dashboards, including the Quality dashboard shown in Figure 19, require a license to Microsoft Office SharePoint Server 2007 or its successor, Microsoft SharePoint Server 2010.

Notice that the first three SKUs shown in the table—Visual Studio 2010 Ultimate, Visual Studio 2010 Premium, and Visual Studio 2010 Professional—all include TFS, the Visual Studio 2010 IDE, and Team Explorer. Nonetheless, they are different products, each with its own set of functions. Figure 22 shows the main capabilities of these three SKUs and how they're related to one another.

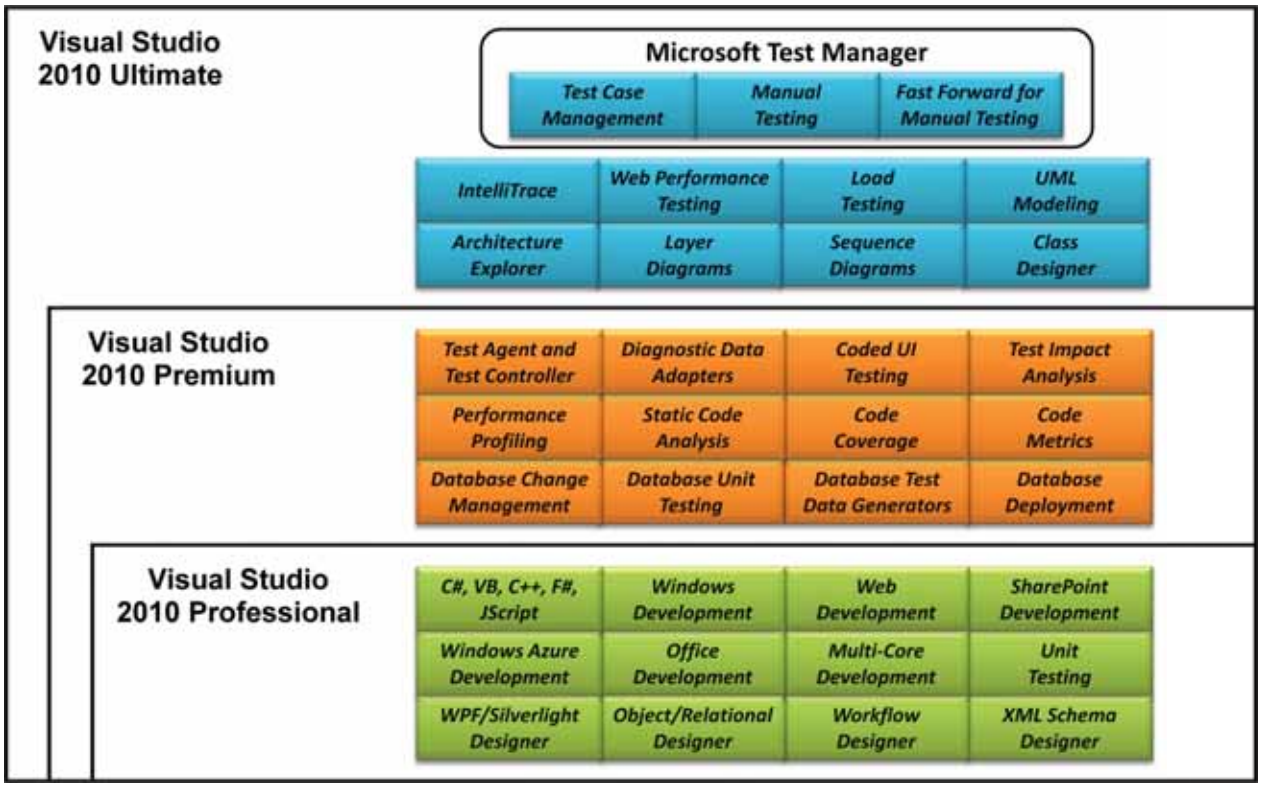


Figure 22: The feature sets in Visual Studio 2010 Professional, Premium, and Ultimate nest like a Russian doll.

As the figure shows, Visual Studio 2010 Professional is a subset of Visual Studio 2010 Premium, which is itself a subset of Visual Studio 2010 Ultimate. In previous releases of this product, then called Visual Studio Team System, Microsoft offered different SKUs aimed at different roles: architects, developers, testers, and database developers. Yet many development teams don't have a sharp separation between these roles—everybody does at least a little of everything. Recognizing this, Microsoft chose to adopt this nested structure for Visual Studio 2010. Which version an organization (or a single developer) buys depends on what functionality they require.

MIGRATION: GETTING THERE FROM HERE

Every development team today already uses some tools. They might use the Visual Studio IDE, for instance, another tool for version control, a third for bug tracking, and an open source unit testing framework. If this team (or the entire organization) decides to adopt Visual Studio 2010, moving from where they currently are to full use of this integrated tool set requires some thought.

One common approach is to adopt Visual Studio 2010 for a new project. This project uses these new tools exclusively, letting developers become proficient with the integrated environment. Assuming all goes well, another project can then use Visual Studio 2010, probably seeded with experienced people from the first project.

What about existing software that's maintained in some other version control system? One option is to migrate that code base to TFS. Both Microsoft partners and Microsoft itself offer tools to help with this, providing migration tools for Visual Source Safe, IBM Rational products, Perforce, Borland StarTeam, and other technologies. Alternatively, an organization might choose to leave existing source where it is, using Visual Studio 2010 only for new projects. One more possibility is to have the source code directory structure on a local machine (called a TFS *workspace*) mirror the structure used in another version control system. This lets the same source code be managed by both TFS version control and the other system, an approach that can help in migration.

Moving to Visual Studio 2010 from earlier versions of Visual Studio Team System also requires a bit of work. The 2010 release uses different formats for projects and build definitions, and so this new version includes tools to convert existing projects and build definitions into these formats. It's possible to access the 2010 version of TFS from older Visual Studio clients, however, which means that an organization needn't upgrade everything at once.

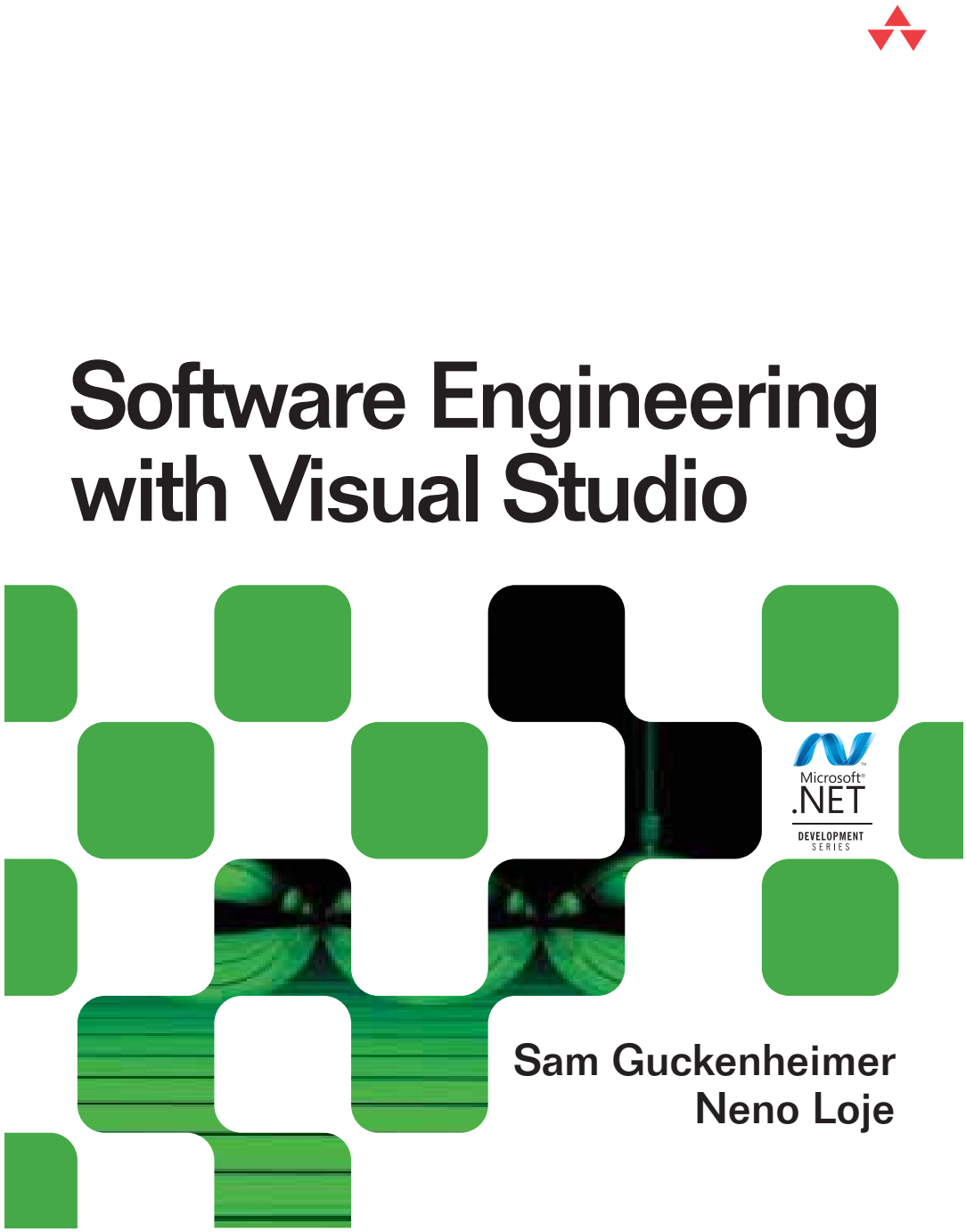
CONCLUSION

Most modern software development is done by teams. Whatever the project, the real goal is to optimize the team's performance over the entire development process, from requirements to release and beyond. An integrated tool set such as Visual Studio 2010 provides a strong foundation for doing this. Built around the hub of Team Foundation Server, the product targets every major aspect of application lifecycle management.

Both craftsmen and artists depend on their tools. Software developers are a little bit of both, and they're no less tool-dependent. While good tools alone aren't sufficient—good people are also required—using the right tools can make development teams significantly more effective. And when teams are functioning well, work gets done, projects get completed, and customers are satisfied. At the end of the day, isn't this the goal of every development project?

ABOUT THE AUTHOR

David Chappell is Principal of Chappell & Associates (www.davidchappell.com) in San Francisco, California. Through his speaking, writing, and consulting, he helps people around the world understand, use, and make better decisions about new technology.


Software Engineering
with Visual Studio
Microsoft
.NET
DEVELOPMENT
SERIES
Sam Guckenheimer
Neno Loje

DRAFT MANUSCRIPT

Books Available

December 2010

This sample chapter has been provided by Pearson Education at this early stage to create awareness for this upcoming book. **It has not yet been copyedited or proofread**; we trust that you will judge this chapter on its content, not grammatical and punctuation errors that will be fixed at a later stage.

Software Engineering with Microsoft Visual Studio

Sam Guckenheimer
Neno Loje

ISBN-13: 9780321685858 ISBN-10: 0321685857* Paperback * 380 pages

Pearson Education

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris •
Madrid Capetown • Sydney • Tokyo • Singapore • Mexico City

Preface to the Second Edition (excerpt)

Five years ago, we extended the world's leading product for individual developers, *Microsoft Visual Studio*, into *Visual Studio Team System*, and it quickly became the world's leading product for development teams. This addition of Application Lifecycle Management (ALM) to Visual Studio made life easier and more productive for hundreds of thousands of our users and tens of thousands of our Microsoft colleagues. Now in 2010, we've just shipped *Visual Studio 2010 Premium, Ultimate, Test Professional, Team Foundation Server* and *Lab Management*. (We've dropped the *Team System* name.)

We've learned a lot from our customers in the last five years. *Visual Studio 2010* is a huge release that enables a high-performance agile software team to release higher quality software more frequently. We set out to enable a broad set of scenarios for our customers. We systematically attacked major root causes of waste in the application lifecycle, elevated transparency for the broadly engaged team, and focused on flow of value for the end customer. We have eliminated unnecessary silos among roles, to focus on empowering a multi-disciplinary, self-managing team. Here are some examples.

No more no repro. One of the greatest sources of waste in software development is a developer's inability to reproduce a reported defect. We call this a "no repro" bug. A tester or user files a bug and later receives a response to the effect of "Cannot reproduce," or "It works on my machine," or "Please provide more information," or something of the sort. Usually this is the first volley in a long game of Bug Ping-Pong, in which no soft-

ware gets improved but huge frustration gets vented. Bug Ping-Pong is especially difficult for a geographically distributed team. As detailed in the Chapters 1 and 8, VS 2010 shortens or eliminates this no-win game.

No more waiting for build setup. Many development teams have mastered the practice of *Continuous Integration* in order to produce regular builds of their software many times a day, even for highly distributed web-based systems. Nonetheless, testers regularly wait for days to get a new build to test, because of the complexity of getting the build deployed into a production-realistic lab. By virtualizing the test lab and automating the deployment as part of the build, VS 2010 enables testers to take fresh builds daily or intraday with no interruptions. Chapter 7 describes how to work with Build and Lab automation.

No more UI regressions. The most effective UI testing is often exploratory, unscripted manual testing. However, when bugs are fixed, it is often hard to tell if they have actually been fixed or if they simply haven't been found again. VS 2010 removes the ambiguity by capturing the action log of the tester's exploration and allowing it to be converted into an automated test. Now fixes can be retested reliably and automation can focus on the actually observed bugs, not the conjectured ones. Chapter 8 covers both exploratory and automated testing.

No more performance regressions. Most teams know the quickest way to lose a customer is with a slow application or web site. Yet teams don't know how to quantify performance requirements and, accordingly, test for load capacity until right before release, when it's too late to fix the bugs that are found. VS 2010 enables teams to begin load testing early. Performance does not need to be quantified in advance, because the test can answer the simple question, "What has gotten slower?" And from the end-to-end result, VS profiles the hot paths in the code and points the developer directly to the trouble spots. Chapters 6 and 8 cover profiling and load testing.

No more missed requirements or changes. Software projects have many moving parts, and the more iterative they are, the more the parts move. It's easy for developers and testers to misunderstand requirements or overlook the impact of changes. To address this, *Visual Studio Test Professional* introduces Test Impact Analysis. This capability compares the changes between any two builds and recommends which tests to run, both

by looking at the work completed between the builds and by analyzing which tests cover the changed code based on prior coverage. Chapters 3 and 4 describe requirements and change management.

No more planning black box. In the past, teams have often had to guess at their historical velocity and future capacity. VS 2010 draws these directly from the Team Foundation Server database and builds an Excel worksheet that allows the team to see how heavily loaded every individual is in the sprint. The team can then transparently shift work as needed. Examples of planning are discussed in Chapters 2 and 4.

No more late surprises. Agile teams, working iteratively and incrementally, often use burndown charts to assess their progress. Not only does VS 2010 automate the burndowns, but project dashboards go beyond burndowns to provide a real-time view of quality and progress from many dimensions: requirements, tasks, tests, bugs, code churn, code coverage, build healtn and impediments. Chapter 4 introduces the "happy path" of running a project, while Chapter 9 looks at troubleshooting project "smells".

No more legacy fear. Very few software projects are truly "greenfield", developing brand new software on a new project. More frequently, teams extend or improve existing systems. Unfortunately, the people who worked on earlier versions are often no longer available to explain the assets they have left behind. VS 2010 makes it much easier to work with the existing code by introducing tools for architectural discovery. VS 2010 reveals the patterns in the software and allows you to automatically enforce rules that reduce or eliminate unwanted dependencies. These rules can become part of the check-in policies that ensure the team's definition of "done" to prevent inadvertent architectural drift. Architectural changes can also be tied to bugs or work, to maintain transparency. Chapter 5 covers the discovery of existing architecture and Chapter 7 shows you how to automate the definition of "done".

No more distributed development pain. Distributed development is a necessity for many reasons: geographic distribution, project complexity, release evolution. VS 2010 takes much of the pain out of distributed development processes both proactively and retrospectively. *Gated check-in* proactively forces a clean build with verification tests before accepting a

check-in. *Branch visualization* retrospectively lets you see where changes have been applied. The changes are visible both as code and work item updates (for example bug fixes) that describe the changes. You can visually spot where changes have been made and where they still need to be promoted. Chapters 6 and 7 show you how to work with source, branches, and backlogs across distributed teams.

No more technology silos. More and more software projects use multiple technologies. In the past, teams often have had to choose different tools based on their runtime targets. As a consequence, .NET and Java teams have not been able to share data across their silos. Visual Studio Team Foundation Server 2010 integrates the two by offering clients in both the Visual Studio and Eclipse IDEs, for .NET and Java respectively. This changes the *either-or* choice into *both-and,* so that everyone wins. Again, Chapters 6 and 7 include examples of working with your Java assets alongside .NET.

These scenarios are not an exhaustive list, but a sampling of the motivation for VS 2010. All of these illustrate our simple priorities: reduce waste, increase transparency, and accelerate the flow of value to the end customer. This book is written for software teams considering running a software project using VS 2010. This book is more about the *why* than the *how*.

This book is written for the team as a whole. It presents information in a style that will help all team members get a sense of each other's viewpoint. I've tried to keep the topics engaging to all team members. I'm fond of Einstein's dictum "As simple as possible, but *no* simpler," and I've tried to write that way. I hope you'll agree and recommend the book to your colleagues (and maybe your boss) when you're done.

Enough about Visual Studio 2010 to Get You Started

When I write about Visual Studio or "VS", I'm referring to the full product line. As shown in Figure P.1, the Visual Studio 2010 family is made up of two server components and a small selection of client-side tools, all available as VS Ultimate.

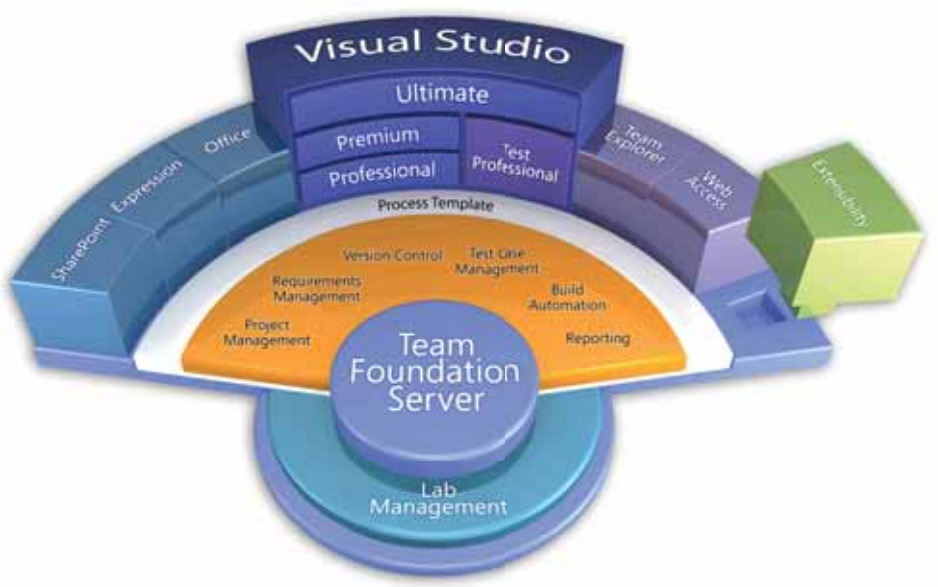


Figure P.1 Team Foundation Server and Lab Management are the server components of Visual Studio 2010. The client components are available in VS Ultimate.

Team Foundation Server (TFS) is the ALM backbone, providing source control management, build automation, work item tracking, test case management, reporting, and dashboards. Lab Management extends TFS to integrate physical and virtual test labs into the development process.

If you just have TFS, you get a client called Team Explorer that launches standalone or as a plug-in to either the Visual Studio Professional or Eclipse IDEs. You also get Team Web Access and plug-ins that let you connect from Excel or Project. SharePoint hosts the dashboards.

Visual Studio Premium adds the scenarios that are described in Chapter 6 around working with the code. Visual Studio Test Professional, although it bears the VS name, is a separate application outside the IDE, designed with the tester in mind. You can see lots of Test Professional examples in Chapter 8. Visual Studio Ultimate, which includes Test Professional, adds architectural modeling and discovery, discussed in Chapter 5.

Of course, all of the clients read and feed data into TFS and their trends surface on the dashboards, typically hosted on SharePoint. You can make your own dashboards with Excel too, but they are harder to scale. "Happy path" dashboard examples are the focus of Chapter 4; unhappy paths are in Chapter 9.

Unlike prior versions, VS 2010 does not have role-based editions. This follows our belief in multidisciplinary, self-managing teams. We want to smooth the transitions and focus on the end-to-end flow. Of course, there's plenty more to learn about VS at the Developer Center of http://msdn.microsoft.com.

About Me

When I wrote the first edition of this book, I had been at Microsoft less than three years. I described my history like this:

I joined Microsoft in 2003 to work on Visual Studio Team System (VSTS), the new product line that was just released at the end of 2005. As the group product planner, I have played chief customer advocate, a role that I have loved. I have been in the IT industry for twenty-some years, spending most of my career as a tester, project manager, analyst, and developer.

As a tester, I've always understood the theoretical value of advanced developer practices, such as unit testing, code coverage, static analysis, and memory and performance profiling. At the same time, I never understood how anyone had the patience to learn the obscure tools that you needed to follow the right practices.

As a project manager, I was always troubled that the only decent data we could get was about bugs. Driving a project from bug data alone is like driving a car with your eyes closed and only turning the wheel when you hit something. You really want to see the right indicators that you are on course, not just feel the bumps when you stray off it. Here too, I always understood the value of metrics, such as code coverage and project velocity, but I never understood how anyone could realistically collect all that stuff.

As an analyst, I fell in love with modeling. I think visually, and I found graphical models compelling ways to document and communicate. But the models always got out of date as soon as it came time to implement anything. And the models just didn't handle the key concerns of developers, testers, and operations.

In all these cases, I was frustrated by how hard it was to connect the dots for the whole team. I loved the idea in Scrum (one of the agile processes) of a "single product backlog"—one place where you could see all the work—but the tools people could actually use would fragment the work every which way. What do these requirements have to do with those tasks, and the model elements here, and the tests over there? And where's the source code in that mix?

From a historical perspective, I think IT turned the corner when it stopped trying to automate manual processes and instead asked the question, "With automation, how can we reengineer our core business processes?" That's when IT started to deliver real business value.

They say the cobbler's children go shoeless. That's true for IT, too. While we've been busy automating other business processes, we've largely neglected our own. Virtually all tools targeted for IT professionals and teams seem to still be automating the old manual processes. Those processes required high overhead before automation, and with automation, they still have high overhead. How many times have you gone to a one-hour project meeting where the first ninety minutes were an argument about whose numbers were right?

Now, with Visual Studio, we are seriously asking, "With automation, how can we reengineer our core IT processes? How can we remove the overhead from following good process? How can we make all these different roles individually more productive while integrating them as a high-performance team?"

Needless to say, that's all still true.

Sam Guckenheimer
Redmond, WA
April 2010

About the Authors

Sam Guckenheimer

Sam Guckenheimer is the Group Product Planner of Microsoft Visual Studio product line. He joined Microsoft in 2003 to lead the vision for Visual Studio Team System and has been responsible for its strategy since then. (In Scrum terms, he's the Product Owner.) Sam has been instrumental in driving the product line to its market-leading position today.

The first edition of Software Engineering with Microsoft Visual Studio Team System has been the standard introduction to the Visual Studio product line and was translated in six languages.

Sam has 25 years' experience as architect, developer, tester, product manager, project manager and general manager in the software industry in the US and Europe. He holds five patents on software lifecycle tools. A frequent speaker at industry conferences, Sam is a Phi Beta Kappa graduate of Harvard University. Sam lives in Kirkland, WA, with his wife and three of his four children, in an innovative green house that can be seen at in the April, 2009, issue of Metropolitan Home at http://www.elledecor.com/home-zremodeling/articles/how-live-prefab.

Neno Loje

Neno Loje has been an independent Application Lifecycle Management (ALM) consultant and Team Foundation Server (TFS) specialist for five years. During that time, he has helped many companies set up a team environment and software development process with VSTS. He is fascinated in

learning how the removal of unnecessary, manual activities makes developers and entire projects more productive and is continually surprised to see how many ways exist (both in process and tools) to achieve the same end goal: *delivering customer value though software*.

When he started in 2005 as a .NET developer, he didn't "get" what ALM was all about or how it could benefit software development, especially for a small team. Once he started using VSTS—because his company was looking for a modern, reliable source code management system—what he discovered went way beyond a pure source control system. By setting up an automated build, which ran pretty painlessly within minutes, the frequency of testing increased and the amount of features to test incrementally decreased. Further, his team was able to show off intermediate results to stakeholders, gather feedback sooner, and gradually automate more and more of previously manual and error-prone release processes.

Interestingly the team did not spend less time on the project. Instead, they reinvested time on more interesting stuff, including additional development and automated tests, which led to higher-quality interim releases and the ability to detect if core functionality was working in every build. (In contrast, they had previously told the testers to start working once the software compiled and installed, even if it didn't start properly.)

When the team looked back at how it worked before VSTS, they questioned how they could 'survive' without those tools. However, what had actually changed weren't just tools, but the *way* they developed software. They did not follow any formalized process or think too much about *how* they did things. Software simply got much easier to release, and testing was no longer deferred to the last week prior to release. The customers, especially, appreciated their new way of work, which was transparent to though the team's ability to deliver more frequent releases.

Says Neno, "*ALM helps teams focus on the important things; VS and TFS are a pragmatic approach to ALM—even for small, non-distributed teams. If you're still not convinced, try it out and judge for yourself.*"

1
The Convergent Evolution

A crisis is a terrible thing to waste.
attr. Paul Romer

The years 2008–10 were the most tumultuous period for the automobile industry in more than three decades. In 2008, Toyota—youngest of the world's major manufacturers—became the world market leader, as it predicted it would six years earlier.[1] Then in 2009, two of the three American manufacturers went through bankruptcy, while the third narrowly escaped. The emergence from this crisis underscored how much the Detroit manufacturers had to adapt to new market conditions. In 1990, Jim Womack and colleagues had coined the term *Lean* in their book *The Machine that Changed the World* to describe a new way of working that Toyota had invented.[2] By 2010, Lean had become a requirement of doing business. As the *New York Times* headline read, "G.M. and Ford Channel Toyota to Beat Toyota."[3]

Then in 2010, Toyota itself stumbled in a major recall. At the same time that its competitors were racing to show off their newly Lean methods, the press was starting to question whether Lean was living up to its reputation.[4] Notably, Toyota's definition of Lean had not included transparency with its customers and community, and the company was taking appropriate heat for this omission. The reality was that Lean had been a great and

necessary advance, but was insufficient without customer transparency too.

Three Forces to Reconcile

Software companies, of course, experienced their spate of bankruptcies in the years 2000-02 and IT organizations were newly challenged to justify their business value. In this period, many industry leaders asked how Lean could have a similarly major impact on software engineering.

Lean was one of several methods that became known as "Agile processes". On a weekend in 2001, seventeen software luminaries convened to discuss "lightweight methods." At the end of the weekend, they launched the Agile Alliance, initially charged around the *AGILE MANIFESTO*.[5] By now, "agility" is mainstream. In the words of Forrester Research:

> Agile adoption is a reality. Organizations across all industries are increasingly adopting Agile principles, and software engineers and other project team members are picking up Agile techniques.[6]

Every industry analyst advocates Agile, every business executive espouses it, and everyone tries to get more of it.

At the same time, two external economic factors came into play. One is global competition. The convergence of economic liberalization, increased communications bandwidth, and a highly skilled labor force in emerging markets made the outsourcing of software development to lower-wage countries (notably Brazil, Russia, India, and China) profitable. The offshore consultancies, in turn, needed to guarantee their quality to American and European customers. Many latched onto Capability Maturity Model Integration (CMMI) from the Software Engineering Institute at Carnegie Mellon University. CMMI epitomized the heavyweight processes against which the agilists rebelled, and it was considered too expensive to be practical outside of the defense industry. The offshorers, with their cost advantage, did not mind the expense and could turn the credential of a CMMI appraisal into a competitive advantage.

The second economic factor is increased attention to regulatory compliance after the lax business practices of the 1990s. In the United States, the Sarbanes-Oxley Act of 2002 (SOX) epitomizes this emphasis by holding business executives criminally liable for financial misrepresentations. This means that software and systems that process financial information are subject to a level of scrutiny and audit much greater than previously known.

These forces—shorter product cycles, outsourcing/offshoring, and compliance—cannot be resolved without a paradigm shift in the way we approach the software lifecycle. The modern economics require agility with accountability. Closing the gap requires a new approach, both to process itself and to its tooling.

What Software Is Worth Building?

To overcome the gap, you must recognize that software engineering is not like other engineering. When you build a bridge, road, or house, for example, you can safely study hundreds of very similar examples. Indeed, most of the time, economics dictate that you build the current one almost exactly like the last to take the risk out of the project.

With software, if someone has built a system just like you need, or close to what you need, then chances are you can license it commercially (or even find it as freeware). No sane business is going to spend money on building software that it can buy more economically. With thousands of software products available for commercial license, it is almost always cheaper to buy. Because the decision to build software must be based on sound return on investment and risk analysis, the software projects that get built will almost invariably be those that are *not* available commercially.

This business context has a profound effect on the nature of software projects. It means that software projects that are easy and low risk, because they've been done before, don't get funded. The only new software development projects undertaken are those that haven't been done before or those whose predecessors are not publicly available. This business reality, more than any other factor, is what makes software development so hard and risky, which makes attention to process so important.

Contrasting Paradigms

The inherent uncertainty in software projects makes it difficult to estimate tasks correctly, which creates a high variance in the accuracy of the estimates. A common misconception is that the variance is acceptable because the positive and negative variations average out. However, because software projects are long chains of dependent events, the variation itself accumulates in the form of downstream delays.[7]

Unfortunately, most accepted project management wisdom comes from the world of roads and bridges. In that world, design risks are low, design cost is small relative to build cost, and the opportunity to deliver incremental value is rare. (You can't drive across a half-finished bridge!) With this style of project management, you determine an engineering design early, carefully decompose the design into implementation tasks, schedule and resource the tasks according to their dependencies and resource availability, and monitor the project by checking off tasks as completed (or tracking percentages completed). For simplicity, I'll call this style of project management the *work-down* approach because it is easily envisioned as burning down a list of tasks.

The work-down approach succeeds for engineering projects with low risk, low variance, and well-understood design. Many IT projects, for example, are customizations of commercial-off-the-shelf software (COTS), such as enterprise resource planning systems. Often, the development is a small part of the project relative to the business analysis, project management, and testing. Typically, these projects have lower variability than new development projects, so the wisdom of roads and bridges works better for them than for new development.

Since 1992, there has been a growing challenge to the work-down wisdom about software process. Agile, Lean, Scrum,[7] Kanban,[8] Theory of Constraints,[9] System Thinking,[10] XP,[11] and Flow-Based Product Development[12] have all been part of the challenge. All of these overlap and are converging into a new paradigm of software engineering and Application Lifecycle Management. No single term has captured the emerging paradigm, but for simplicity, I'll call this the *VALUE-UP* approach. And as happens with new paradigms, the value-up view has appeared in fits and starts (see Figure 1.1).

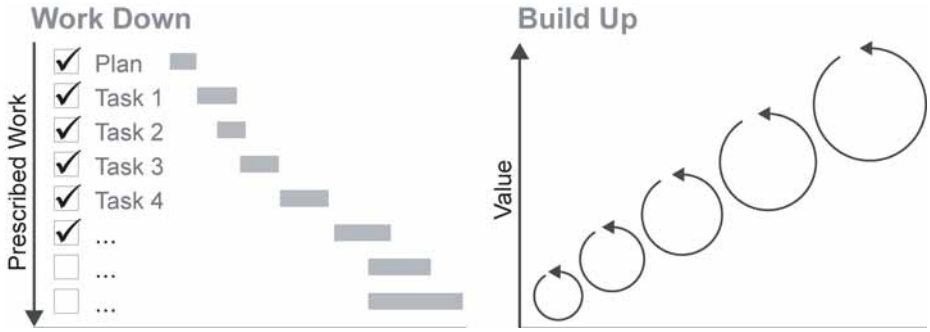


FIGURE 1.1 The attitudinal difference between work-down and value-up is in the primary measurement. Work-down treats the project as a fixed stock of tasks at some cost that need completion and measures the expenditure against those tasks. Value-up measures value delivered at each point in time and treats the inputs as variable flows rather than a fixed stock. Several processes that have emerged in the last decade or two converge on an adaptive, empirical approach to process control rather than the prescriptive, work-down style of waterfall predecessors.

The Value-Up approach stresses three fundamental principles which reinforce each other:

Flow of value, where value is defined by the customer.

Continual reduction of waste impeding the flow.

Transparency enabling team members to continually improve the above two.

These three principles reinforce each other as shown in Figure 1.2. Flow of value enables transparency, in that you can measure what's important to the customer, namely potentially shippable software. Transparency enables discovery of waste. Reducing waste, in turn, accelerates flow and enables greater transparency.

These three aspects need to be understood together like three legs of a stool. Visual Studio Team System 2005 was one of the first commercial products to support software teams applying these practices and Visual Studio 2010 has made a great leap forward to create transparency, improve flow, and reduce waste in software development. VS 2010 is also one of the first products to tackle end-to-end Value-Up engineering and project management practices. A key set of these practices come from Scrum.

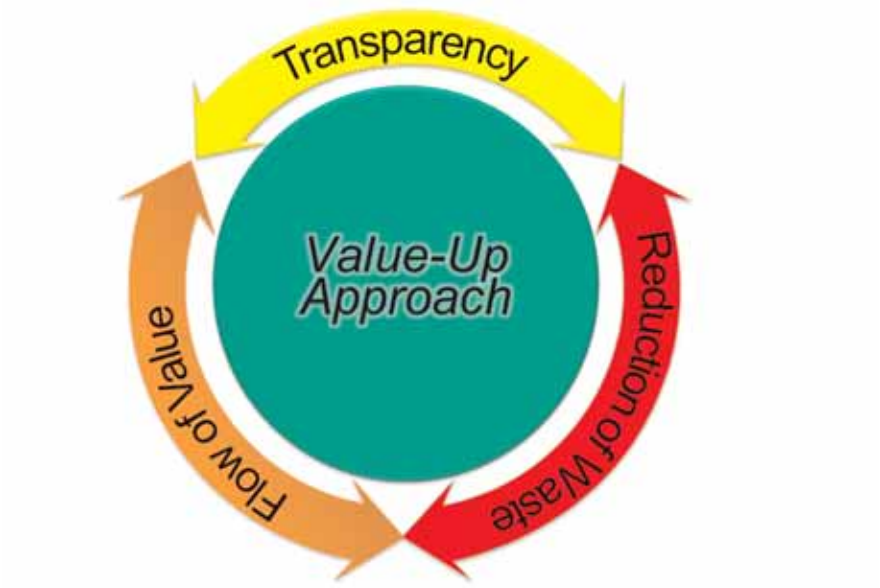


FIGURE 1.2 Flow of Value, Transparency, and Reduction of Waste form the basis of the Value-Up Paradigm.

Scrum

As Forrester Research recently discovered, "When it comes to selecting an Agile methodology, Scrum is the overwhelming favorite."[13] Scrum leads over the nearest contender by a factor of three. Scrum has won acceptance because it simplifies putting the principles of Flow of Value, Transparency, and Reduction of Waste into practice.

Scrum identifies three interlocking cadences: Release or Product Planning, Sprint (usually 2-4 weeks), and Day, and for each cadence it prescribes specific ceremonies and time boxes to keep the overhead low. To ensure flow, every Sprint produces a potentially shippable increment of software that delivers a subset of the product backlog in a working form. The cycles are shown in Figure 1.3.[14]

Scrum introduced the concept of the *product backlog*, "a prioritized list of everything that might be needed in the product."[15] The product backlog contains the definition of the intended customer value. Scrum enables

transparency by prescribing the frequent delivery of "potentially shippable increments" to enable stakeholders to assess the value and provide feedback to the team. Stakeholders need to see and try the interim increments in order to advise on what is desired next.

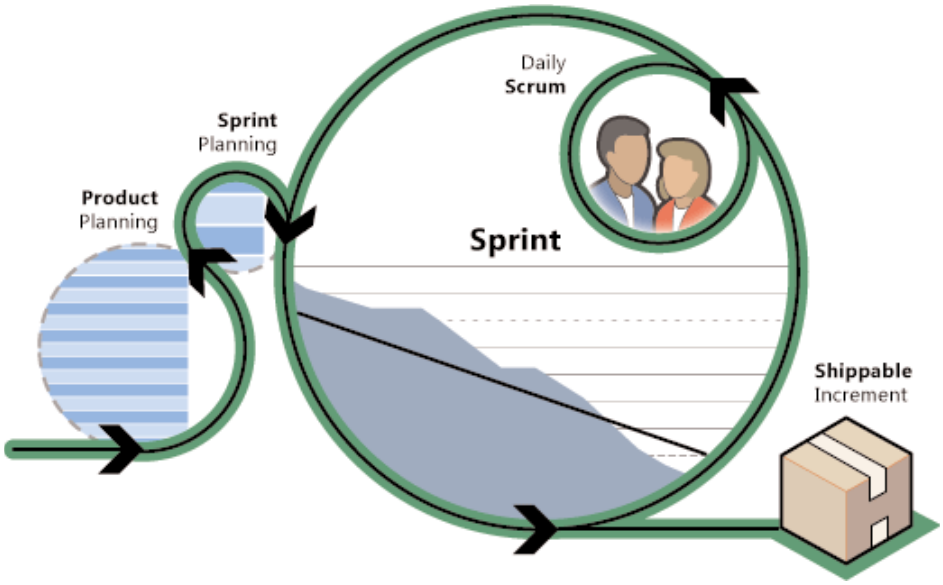


Figure 1.3 The central graphic of the Scrum methodology is a great illustration of flow in the management sense. Not surprisingly, Scrum pioneered the concept of a single product backlog as a management technique.

Core to Scrum is the concept of self-managing teams, who use transparently available metrics to control their own work in process and improve their own velocity of flow. Team members are encouraged to make improvements whenever necessary to reduce waste. The Sprint cadence formally ensures that a "retrospective" is used at least monthly to identify and prioritize actionable process improvements. Scrum characterizes this cycle as "inspect and adapt".

VS 2010 actively supports teams in practicing Scrum. Dashboards transparently keep the product backlog visible to the team and stakeholders and expose potential areas of waste. And many mechanisms discussed below help the team keep software potentially shippable.

Increasing the Flow of Value in Software

Central to Value-Up is an emphasis on *flow*. The flow of customer value is the primary measure of the system of delivery. David J. Anderson summarizes this view in *Agile Management for Software Engineering*:

> Flow means that there is a steady movement of value through the system. Client-valued functionality is moving regularly through the stages of transformation—and the steady arrival of throughput—with working code being delivered.[16]

In this paradigm, you do not measure planned tasks completed as the primary indicator of progress; you count units of value delivered.

However, unlike a manufacturing process, software development projects don't produce the same things over and over. In practice, most software projects are started only when there is no available alternative to reuse, buy or download. In other words, when the solution is unknown. This newness creates an inherent tension around the communication between stakeholders and the development team. When stakeholders say "Make me one of those," their desired result is typically not something for which complete manufacturing specs and processes exist.

Consistent with Scrum, VS 2010 offers an always visible product backlog to increase the communication around the flow of customer-valued deliverables. The product backlog is the current agreement between stakeholders and the development team regarding the next increments to build, and is kept in terms understandable to the stakeholders. The product backlog is visible on a common dashboard, showing both progress and impediments.

By measuring progress in the product backlog, Visual Studio keeps the focus on customer value. Further, by measuring and displaying progress (or lack of) across many dimensions of the potentially shippable software, VS keeps the focus on this flow.

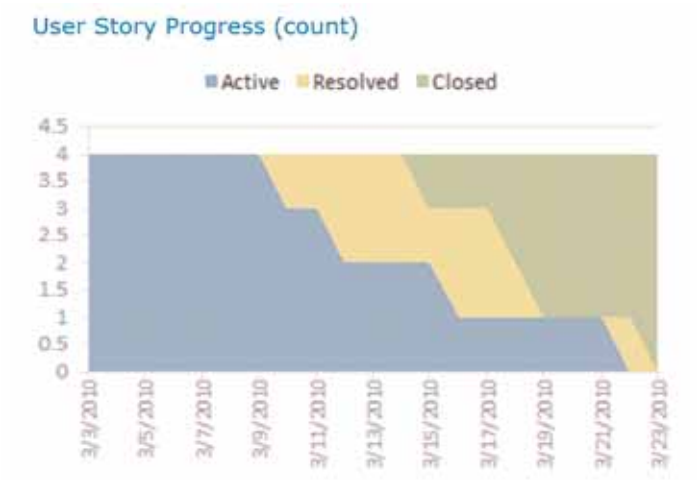


FIGURE 1.4 The project dashboard uses a burndown chart to show the state transitions of user stories as they move from active (committed in the Sprint) to resolved (ready for test) to closed (done and tested). This trend is a great way to assess flow.

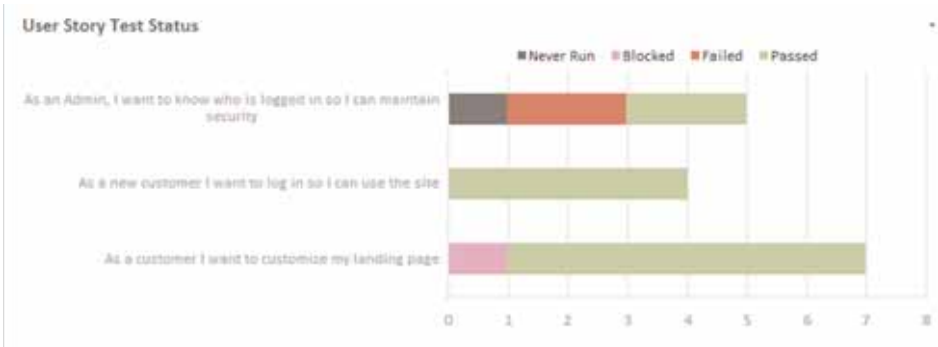


FIGURE 1.5 In addition to the aggregate trend shown in Figure 1.4, this graphic from the VS 2010 team dashboard breaks out the current test status of each product backlog item, i.e. how close each item is to *potentially shippable*.

Potentially Shippable

In 2008, the plight of the financial sector plunged the world economy into the steepest recession of the last seventy years. Economists broadly agree that the problem was a shadow banking system with undisclosed and

unmeasured financial debts, hidden by murky derivatives. Hopefully, this crisis will lead government regulators to remember Justice Brandeis' words, "Sunlight is said to be the best of disinfectants; electric light the most efficient policeman."[17]

For software teams, the equivalent of these unknown liabilities is *technical debt.* Technical debt refers to work that needs to be done in order to achieve the *potentially shippable* threshold, such as fixing bugs, unit testing, integration testing, performance improvement, security hardening, or refactoring for sustainability. Technical debt is an unfortunately common form of waste. Unanticipated technical debt can crush a software project, leading to unpredictable delays, costs, and late cancellation. And similar to the contingent financial liabilities, technical debt is often not disclosed or measured until it is too late.

Among the problems with technical debt, is the fact that it prevents the stakeholders from seeing what software is actually in a potentially shippable state. This obstacle is the reason Scrum prescribes that every product backlog item must be delivered according to a *Definition of Done* agreed by the team. While Scrum does not prescribe a universal *Definition of Done,* it does encourage the use of transparent automation such as described in the VS examples below. Think of the transparency like Louis Brandeis' electric light—it makes the policeman less necessary. Together the common *Definition of Done* and transparent view of progress prevent the accumulation of technical debt, and thereby enable the team and its stakeholders to assess the team's true velocity.

Reducing Waste in Software

The enemy of flow is waste. This opposition is so strong that reduction of waste is the most widely recognized aspect of Lean. Taiichi Ohno of Toyota, the father of Lean, developed the taxonomy of *muda* (Japanese for *waste*), *mura* (inconsistency) and *muri* (unreasonableness), such that these became common business terms.[18] Ohno categorized seven types of *muda* with an approach for reducing every one. Mary and Tom Poppendieck introduced the *muda* taxonomy to software in their first book.[19] An updated version of this taxonomy is shown in Table 1.1.

TABLE 1.1 Taiichi Ohno's taxonomy of waste provides a valuable perspective for thinking about impediments in the application lifecycle.

Muda	**In-Process Inventory**	Partially implemented user stories, bug debt, and incomplete work carried forward. Requires multiple handling, creates overhead & stress.
	Over-Production	Peanut butter. Teams create low-priority features and make them self-justifying. This work squeezes capacity from the high-priority work.
	Extra Processing	Bug debt, reactivations, triage, redundant testing, relearning of others' code, handling broken dependencies
	Transportation	Handoffs across roles, teams, divisions, etc.
	Motion	Managing enlistments, lab setup, parallel release work
	Waiting	Delays, blocking bugs, incomplete incoming components or dependencies
	Correction	Scrap and rework of code
Muda	**Unevenness**	Varying granularity of work, creating unpredictability in the flow
	Inconsistency	Different definitions of *Done*, process variations that make assessment of *potentially shippable* impossible
Muda	**Absurdity**	Stress due to excessive scope
	Unreasonableness	Expectations of heroics
	Overburden	Stress due to excessive overhead

Consistent with Ohno's taxonomy, *in-process inventory, transportation, motion,* and *waiting* often get overlooked in software development. Especially when many specialist roles are involved, waste appears in many subtle ways. As Kent Beck observed, "the greater the flow, the greater the need to support transitions between activities."[20] Some of the transitions take

seconds or minutes, such as the time a developer spends in the red-green-refactor cycle of coding and unit testing. Other transitions too often take days, weeks, or unfortunately months. All the little delays add up.

Consider the effort spent in making a new build available for testing. Or think about the handling cost of a bug that is reported fixed and then has to get reactivated. Or consider writing specs for requirements that ultimately get cut. All of these wastes are common to software projects.

Visual Studio 2010 has focused on reducing the key sources of waste in the software development process. Team Foundation Server build automation allows continuous or regularly scheduled builds, and with "gated check-in" can force builds before accepting changed code. Lab Management can automatically deploy those builds directly into virtualized test environments.

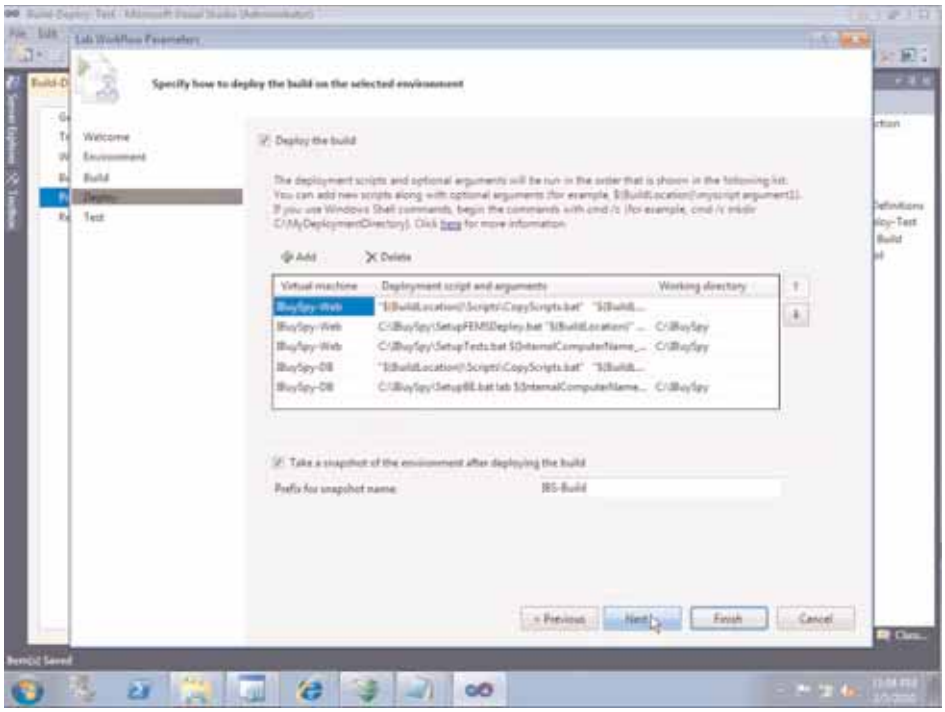


FIGURE 1.6 Team Build definitions define the workflow for both build and deployment. This allows the automation of deployment into a virtualized test lab.

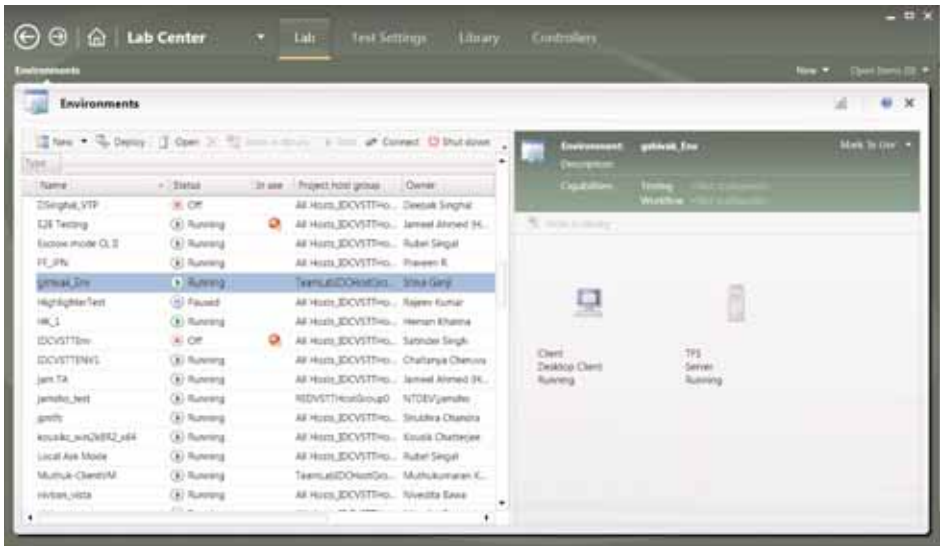


FIGURE 1.7 The Lab Center in Microsoft Test Manager (part of Visual Studio) manages Test Environments composed of virtual machines. The Team Build, shown in Figure 1.6, automatically deploys the software under test into these Test Environments.

An egregious example of waste is "Bug Ping-Pong". Every tester or product owner has countless stories of filing bugs with meticulous descriptions, only to receive the response from a programmer, "Cannot reproduce." There are many variants of this "No repro" categorization, such as "Need more information" or "Works on my machine." This usually leads to a repetitive cycle that involves every type of *muda* as the tester and programmer try to isolate the fault. And the cycle often leads to frustration, blame, and low morale.

Bug Ping-Pong happens not because testers and developers are incompetent or lazy, but because software bugs are often truly hard to isolate. Some bugs may demonstrate themselves only after thousands of asynchronous events occur, and the exact repro sequence cannot be recreated deterministically. Bugs like this are usually found by manual or exploratory testing, not by test automation. When a tester files a bug, VS 2010 automatically invokes up to six mechanisms to eliminate the guesswork from fault isolation.

All of the tester's interactions with the software under test are captured in an action log, grouped according to the prescribed test steps (if any).

A *full motion video* captures what the tester sees, time-indexed to the test steps.

Screenshots highlight anything the tester needs to point out during the sequence.

System Configurations are automatically captured for each machine involved in the test environment.

An *IntelliTrace log* records application events and the sequence of code executed on the server, to enable future debugging based on this actual execution history.

Virtual machine checkpoints record the state of all the machines in the test environment in their actual state at the time of failure.

Some examples from VS 2010 follow in figures 1.8 and 1.9.

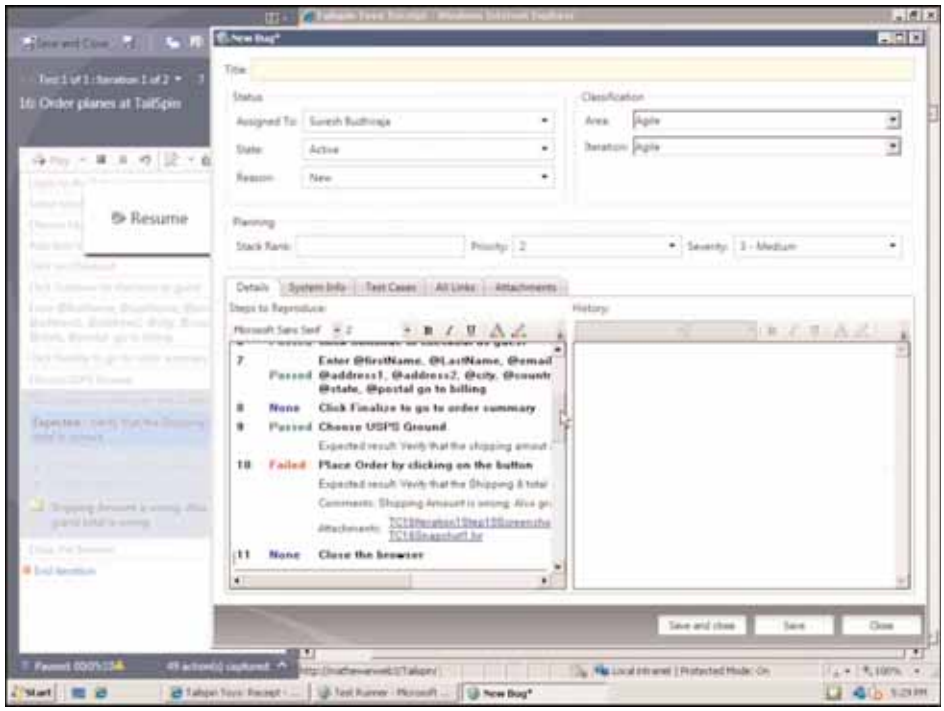


FIGURE 1.8 When a tester files a bug, full motion video is automatically indexed to the test steps and captured with the bug, so that a developer can see exactly what the tester saw at the time of the bug report.

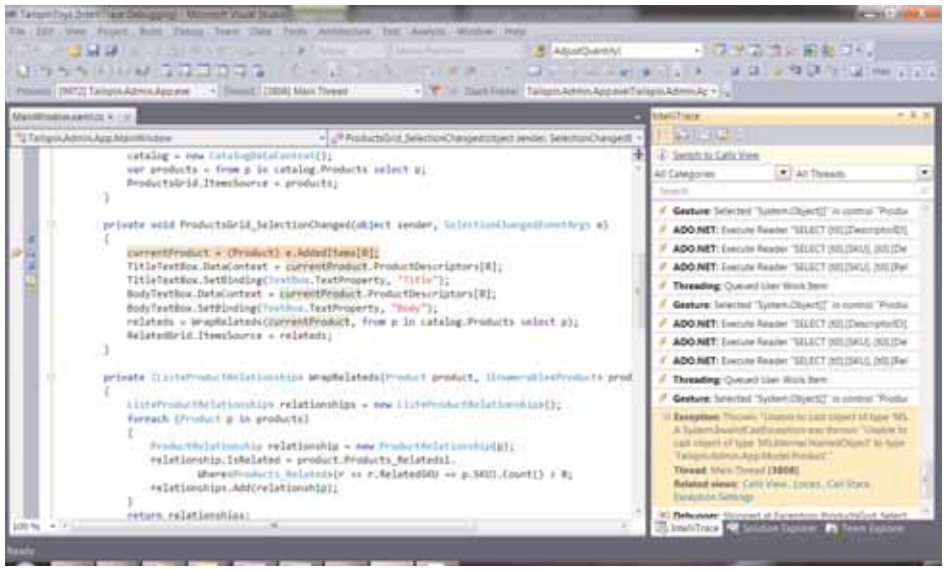


Figure 1.9 The IntelliTrace log captures the actual call sequence of the software under test, here on an ASP.NET server. You can navigate either from the Calls view on the right to the code editor on the left, or from the code file to the call sequence.

When a developer receives an actionable bug like this, he can move through the history of the fault both in video and in the IntelliTrace log. With IntelliTrace, it's possible to step or jump though the code forwards or backwards, as the code was actually executed. From the execution log, the developer can jump into the code editor to make a fix. After fixing the code in question, Test Impact Analysis will suggest the unit tests to run based both on the directly changed code and any dependencies whose behavior might be affected by the change.

To guard against re-introducing the bug, VS can turn the tester's action log into an automated regression test. Now, after the bug is fixed, automated tests can prevent its unnoticed recurrence. Note this workflow is different from the typical test automation path of today. Rather than conjecturing which regression tests would be useful, the tester or developer creates the test when the bug is discovered. These tests complement (rather than replace) automated unit tests, but because they are based on logs from actual bugs, their value is clear from the time of creation.

Work In Process Hides Waste

Very frequently, waste goes unnoticed because it is buried in the queues of work-in-process.[21] In software development, work-in-process consists of tasks not yet done, bugs not yet fixed, tests not yet run, builds not yet verified, and software not yet deployed. There is also a frequent problem of over-production, where teams throw extra work into the project that never makes it to completion and then has to be cut before release. Far too often, software teams accept these conditions as normal, and fail to see the technical debt accumulating.

Eliminating Bug Ping Pong is one of the clearest ways in which VS 2010 reduces work in process and allows quick turnaround and small batches in testing. Another is *Test Impact Analysis*, which recommends the highest priority tests for each build, based both on completed work and historical code coverage. These are examples, whose value Don Reinertsen beautifully summarizes in Figure 1.10.[22]

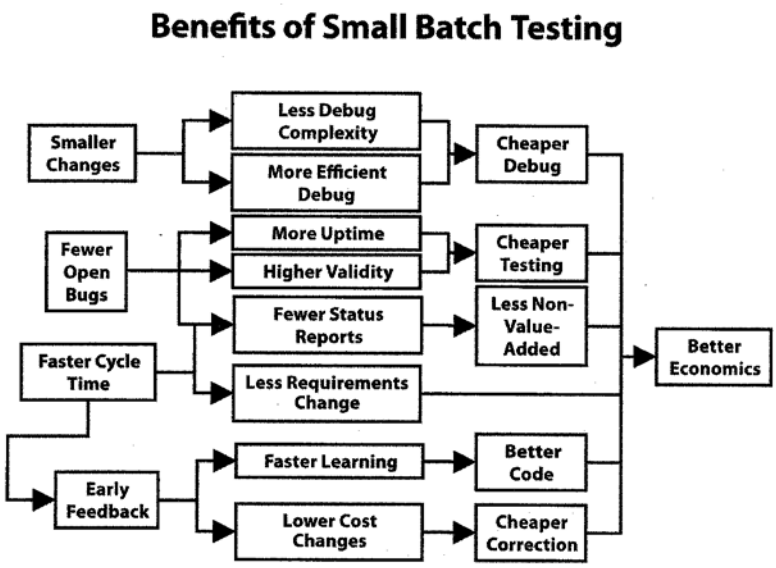


Figure 5-3 Smaller batch sizes produce a wide range of benefits. The impact on overall economics is surprisingly large.

FIGURE 1.10 Reinertsen shows the benefits of reducing work in process and batch size in fine detail. As he puts it, "The impact on overall economics is surprisingly large."

Transparency

Scrum and all Agile processes emphasize self-managing teams. Successful self-management requires transparency. Transparency, in turn, requires measurement with minimal overhead. Burndown charts of work remaining in tasks became an early icon for transparency. VS takes this idea further, to provide dashboards that measure not just the tasks, but multidimensional indicators of quality.

VS enables and instruments the process, tying source code, testing, work items, and metrics together. Work items include all the work that needs to be tracked on a project, such as scenarios, development tasks, test tasks, bugs, and impediments. These can be viewed and edited in Team Explorer, Team Web Access, Visual Studio, Microsoft Excel, or Microsoft Project.

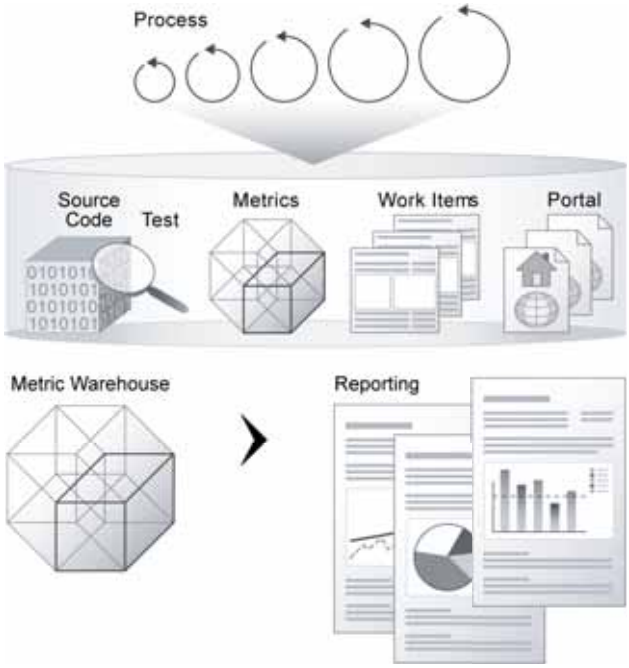


FIGURE 1.11 VS enables and instruments the process, tying source code, testing, work items, and metrics together.

A lot of ink has been spent in the last twenty years on the concept of Governance with regard to software development. Consider this quote from an IBM Redbook, for example:

> *Development governance addresses an organization-wide measurement program whose purpose is* ***to drive consistent progress assessment*** *across development programs, as well as the use of* ***consistent steering mechanisms****. [Emphasis added.]*[23]

Most of the discussion conveys a bias that problems in software quality can be traced to a lack of central control over the development process. If only we measured developers' activities better, the reasoning goes, we could control them better. Value-Up takes a very different attitude to command and control. Contrast the quote above with the following analysis.

> *Toyota has long believed that* ***first-line employees*** *can be more than cogs in a soulless manufacturing machine; they* ***can be problem solvers, innovators, and change agents****. While American companies relied on staff experts to come up with process improvements, Toyota gave every employee the skills, the tools, and the permission to solve problems as they arose and to head off new problems before they occurred. The result: Year after year, Toyota has been able to get more out of its people than its competitors have been able to get out of theirs. Such is the power of management orthodoxy that it was only after American carmakers had exhausted every other explanation for Toyota's success – an undervalued yen, a docile workforce, Japanese culture, superior automation – that they were finally able to admit that* ***Toyota's real advantage was its ability to harness the intellect of "ordinary" employees****.*[24]

The difference in attitude couldn't be stronger. The "ordinary" employees—members of the software team—are the ones who can best judge how to do their jobs. They need tools, suitable process, and a supportive environment, not command and control.

Self-Managing Teams

Lean turns governance on its head, by trusting teams to work toward a shared goal, and using measurement *transparency* to allow teams to improve the flow of value and reduce waste themselves. In VS, this transparency is fundamental and available both to the software team and its stakeholders. Consider, for example, the VS 2010 view of the status of testing shown in Figure 1.12.

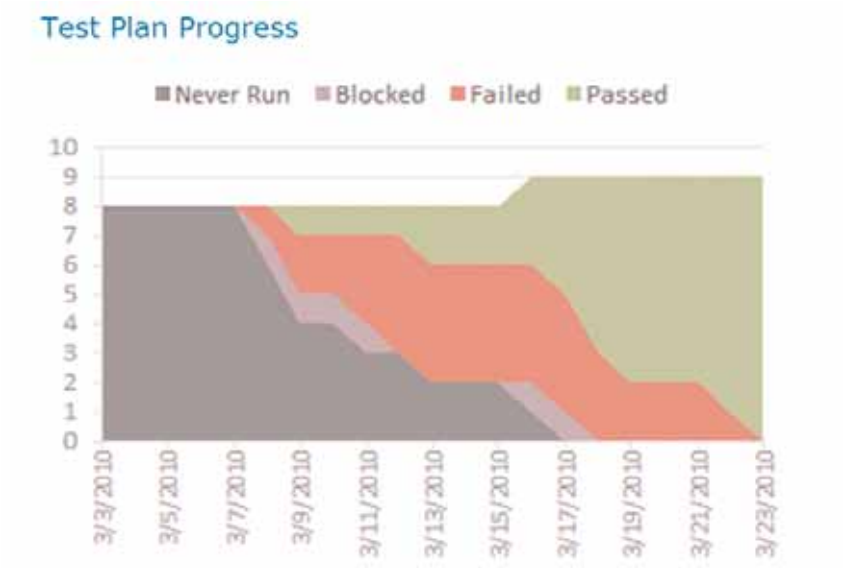


FIGURE 1.12 Test Plan Progress uses a cumulative flow diagram to track planned tests as they progress from *Never Run* to *Passed*.

In addition to the trends, every build in VS 2010 has its own web page that acts as an automated release note. It shows exactly what work was delivered into the build, what code changesets can be inspected, what tests were run, and where the output was deployed. Figure 1.13 shows an example.

Together, these dashboards act as an early warning system against typical dysfunctions, such as blocked builds, inadequate testing, regressions, poor test coverage, uneven progress, and more specific impediments to flow.

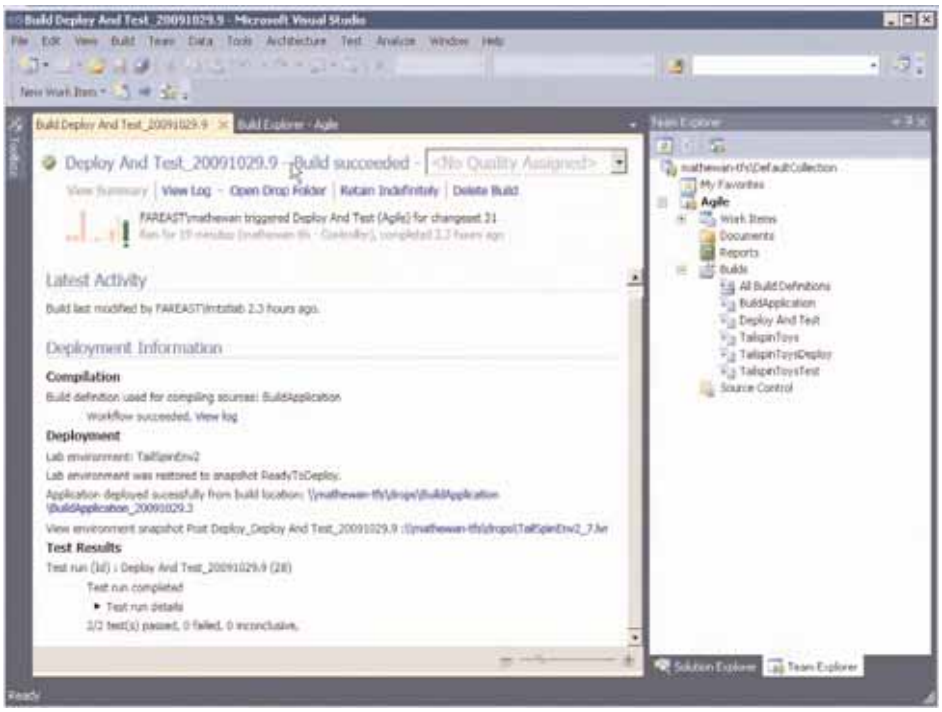


FIGURE 1.13 Every build has an automated release note, accessible from the dashboard or inside Visual Studio.

Back to Basics

It's hard to disagree with Lean expert Jim Womack's words,
The critical starting point for lean thinking is value. Value can only be defined by the ultimate customer.[25]

Similarly for software, the Value-Up Paradigm changes the way we work to focus on value to the customer, reduce the waste impeding the flow, and transparently communicate, measure, and improve the process. The auto industry took fifty years to absorb the lessons of Lean, until its customers' and investors' patience wore out. In mid-2009, on the day General Motors emerged from bankruptcy, CEO Fritz Henderson told a news conference in Detroit.

> *At the new GM, we're going to make the customer the center of everything. And we're going to be obsessed with this, because if we don't get this right, nothing else is going to work.*[26]

Six months later, when GM had failed to show suitable obsession, Henderson was out of a job. It may be relatively easy to dismiss the woes of Detroit as self-inflicted, but we in the software industry have carried plenty of our own technical debt that has cost many a CIO his job too.

Summary

For a long time, Scrum creator Ken Schwaber has said, "Scrum is all about common sense," but a lesson of the last decade is that we need supportive tooling too.[27] To prevent the practice from diverging from common sense, the tools need to reinforce the flow of value, reduce the waste, and make the process transparent. These Value-Up principles have been consistently reflected in five years of customer feedback that are reflected in VS 2010.

In practice, most software processes require a good deal of manual work, which makes collecting data and tracking progress expensive. Up front, such processes need documentation, training, and management, and they have high operating and maintenance costs. Most significantly, the process artifacts and effort do not contribute in any direct way to the delivery of customer value. Project managers can often spend 40 hours a week cutting and pasting to report status.

In contrast, the business forces driving software engineering today require a different paradigm. A team today needs to embrace customer value, change, variance, and situationally specific actions as a part of everyday practice. This is true whether projects are in-house or outsourced and whether they are local or geographically distributed. Managing such a process usually requires a Value-Up approach.

And the value-up approach requires supportive tooling. Collecting, maintaining, and reporting the data without overhead is simply not practical otherwise. In situations where regulatory compliance and audit are required, the tooling is necessary to provide

Endnotes

[1] Womack, James P., and Daniel T. Jones. *Lean Thinking : Banish Waste and Create Wealth in Your Corporation*. New York: Free Press, 2003, p. 150.

[2] Womack, James P., Daniel T. Jones, and Daniel Roos. *The Machine That Changed the World : How Japan's Secret Weapon in the Global Auto Wars Will Revolutionize Western Industry*. New York: Rawson Associates,, 1990.

[3] "G.M. and Ford Channel Toyota to Beat Toyota", *The New York Times*, March 7, 2010, p. BU1.

[4] Daisuke Wakabayashi, "How Lean Manufacturing Can Backfire ", *The Wall Street Journal*, January 30, 2010.

[5] http://www.agilemanifesto.org

[6] West, Dave and Tom Grant, "Agile Development: Mainstream Adoption Has Changed Agility Trends In Real-World Adoption Of Agile Methods", avaialble from http://www.forrester.com/rb/Research/agile_development_mainstream_adoption_has_changed_agility/q/id/56100/t/2, p.17

[7] Ken Schwaber, Jeff Sutherland, *Scrum: Developed and Sustained* (a.k.a. *Scrum Guide)* Feb 2010, http://www.scrum.org/scrumguides

[8] Kniberg, Henrik and Mattias Skarin, *Kanban and Scrum—making the most of both*, InfoQ, 2009, http://www.infoq.com/minibooks/kanban-scrum-minibook

[9] Goldratt, Eliyahu M,. *THE GOAL* (North River Press, 1986)

[10] Gerald M. Weinberg, *Quality Software Management, Volume I: Systems Thinking* (New York: Dorset House, 1992).

[11] Beck, Kent, and Cynthia Andres. *Extreme Programming Explained: Embrace Change.* Boston: Addison-Wesley, 2003.

[12] Reinertsen, Donald G. *The Principles of Product Development Flow: Second Generation Lean Product Development*. Redondo Beach, CA: Celeritas Publishing, 2009.

[13] West, op cit., p. 4.

[14] This variation of the diagram is available from http://msdn.microsoft.com/.

[15] Ken Schwaber, Jeff Sutherland, *Scrum: Developed and Sustained* (a.k.a. *Scrum Guide)* Feb 2010, http://www.scrum.org/scrumguides/

[16] David J. Anderson, *Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results* (Upper Saddle River, NJ: Prentice Hall, 2004), p. 77.

[17] Louis Brandeis, "What Publicity Can Do," in *Harper's Weekly*, December 20, 1913, available from http://www.law.louisville.edu/library/collections/brandeis/node/196

[18] Ohno, Taiichi. Toyota *Production System : Beyond Large-scale Production.* Cambridge, Mass: Productivity Press, 1988.

[19] Poppendieck, Mary B., and Thomas D. Poppendieck. Lean Software Development : An Agile Toolkit. Boston: Addison-Wesley, 2003.

[20] Kent Beck, Tools for Agility, Three Rivers Institute, 6/27/2008, http://www.microsoft.com/downloads/details.aspx?FamilyId=AE7E07E8-0872-47C4-B1E7-2C1DE7FACF96&displaylang=en

[21] Eli Goldratt, The Goal, clearly illustrates this problem.

[22] Reinertsen op. cit. p. 121

[23] IBM IT Governance Approach: Business Performance through IT Execution, Feb 2008, http://www.redbooks.ibm.com/Redbooks.nsf/RedbookAbstracts/sg247517.html, p. 35

[24] Hamel, Gary, "The Why, What, and How of Management Innovation", *Harvard Business Review*; Feb2006, Vol. 84 Issue 2, p72-84, http://search.ebscohost.com/login.aspx?direct=true&db=bch&AN=19406184&loginpage=Login.asp&site=ehost-live&scope=site

[25] Womack 2003, p. 16.

[26] *All Things Considered*, National Public Radio, July 10, 2009; http://www.npr.org/templates/story/story.php?storyId=106459662

What do the Analysts Think?

The analyst's reviews are out and they are very positive. Gain some insight into how the Analysts think about the ALM marketplace and where Visual Studio 2010 is placed by reading reports from Forrester and voke.

May 5, 2010

The Forrester Wave™: Agile Development Management Tools, Q2 2010

by Dave West and Jeffrey S. Hammond
for Application Development & Delivery Professionals

FORRESTER® Making Leaders Successful Every Day

May 5, 2010

The Forrester Wave™: Agile Development Management Tools, Q2 2010

Atlassian, CollabNet, IBM, Microsoft, MKS, And Rally Lead The Pack

by Dave West and Jeffrey S. Hammond
with Mike Gilpin and David D'Silva

EXECUTIVE SUMMARY

In Forrester's evaluation of Agile development management (ADM) tool vendors, we found that IBM and MKS led the pack with the best overall current feature sets. Atlassian, CollabNet, and Microsoft are also Leaders with capable products and aggressive strategies that will result in significant product improvements in 2010 and beyond. Rally Software Development is also a category Leader; it offers the best current balance of product capability and strategic outlook. HP, Serena Software, and VersionOne are Strong Performers offering competitive options. In the case of HP and Serena, their products are new introductions to the market and should improve as the vendors mature and gain customers. VersionOne is a stalwart in the Agile space that offers excellent planning capabilities but is less flexible than other products when it comes to reporting and integration with application life-cycle management (ALM) tools. And while the solution recently acquired by Micro Focus appeals to client-server and legacy developers, Micro Focus must clarify its future strategy for ADM before it can move into a leadership position.

TABLE OF CONTENTS

NOTES & RESOURCES

Forrester conducted lab-based product evaluations in October 2009 and interviewed 30 vendor and user companies including Atlassian, CollabNet, HP, IBM, Micro Focus, Microsoft, MKS, Rally Software Development, Serena Software, and VersionOne.

Related Research Documents

"Agile Development: Mainstream Adoption Has Changed Agility"
January 20, 2010

"Software Configuration Management Tool Adoption Trends In The Americas"
June 18, 2009

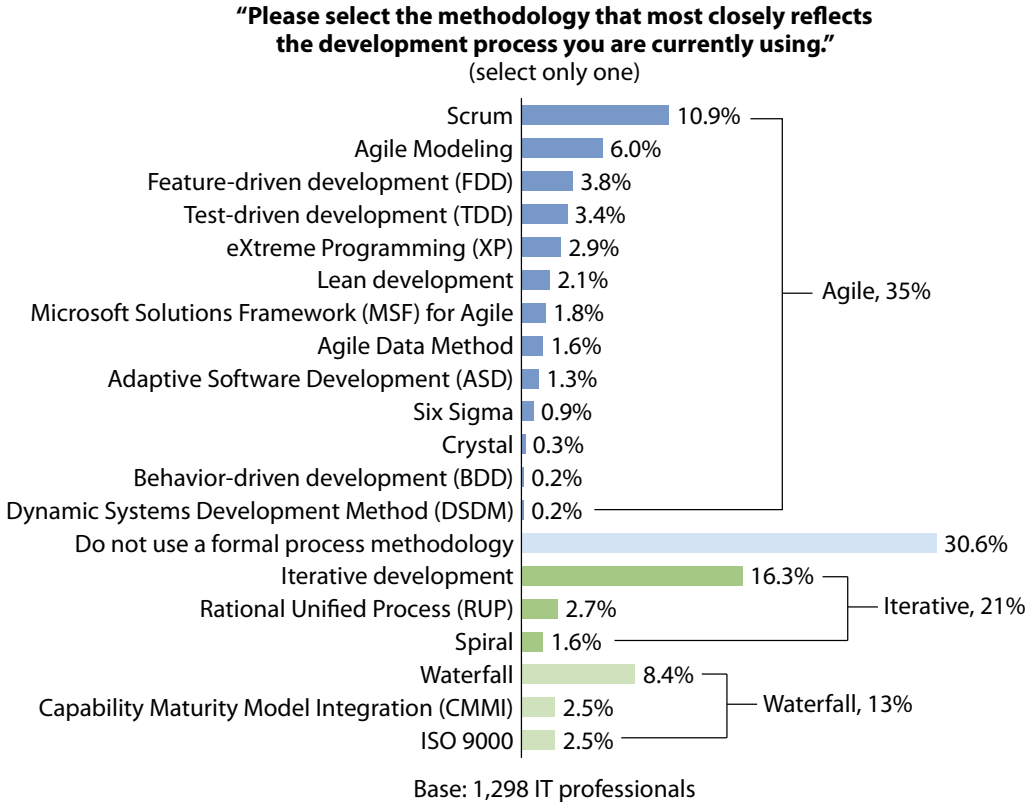
"Best Practices: Software Development Processes"
April 15, 2009

"Standardized Software Change And Configuration Management: Achievable Goal Or Wishful Thinking?"
October 19, 2007

AGILE DEVELOPMENT IS RAPIDLY BECOMING THE NORM

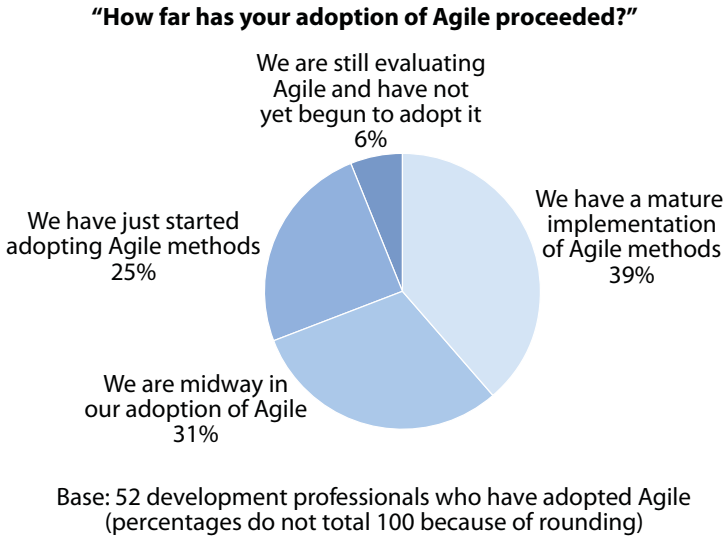
In a recent survey, 35% of surveyed organizations described their primary development method as Agile; Scrum, at 11%, was the most popular Agile development approach (see Figure 1). In a different survey, we questioned the nature of Agile adoption and found that 39% of the organizations we surveyed consider their implementation mature (see Figure 2). The mainstream business press is even starting to get on the Agile bandwagon, referencing its use at eBay as crucial to the success of eBay's business.[1] This increased level of adoption has serious implications for development organizations' tool use, changing not only the process model being followed but also the very nature of work undertaken and who is involved in that work.

Figure 1 Agile Is Organizations' Primary Development Approach



56100 Source: Forrester Research, Inc.

Figure 2 Most Organizations View Their Agile Adoption As Mature



Source: Q3 2009 Global Agile Adoption Online Survey

56100 Source: Forrester Research, Inc.

Scaling Agile Requires Automation

Our interviews with application development professionals revealed that *scaling* Agility is a common issue — and that scaling Agile practices requires implementing tools. The vice president of a large financial company described the need for automation: "When you have one project on a whiteboard with Post-its, it is fine, but when you have five or six projects, the whiteboard approach just does not cut it. We haven't even got enough whiteboards." Automation is required because:

- **Sharing status is time-consuming.** This is particularly true when the team is spread across many locations and is working on many projects. The ability to quickly and easily share status information is crucial when the team self-selects work and changes direction based on that work's results.

- **Many Agile practices require automation.** As Agile implementations mature, teams adopt more-sophisticated practices associated with testing, architecture, and build. To be effective, these practices require a sound automation foundation that supports automated test integration, code comparison, and integrated build management.

- **Retrospectives require information.** As teams work through sprints, team members can make and record many important observations. These observations help improve the process and are a key input to retrospectives. Without automation, it is very hard to remember the status of a project at a particular moment or to be able to do analysis to improve working practices.

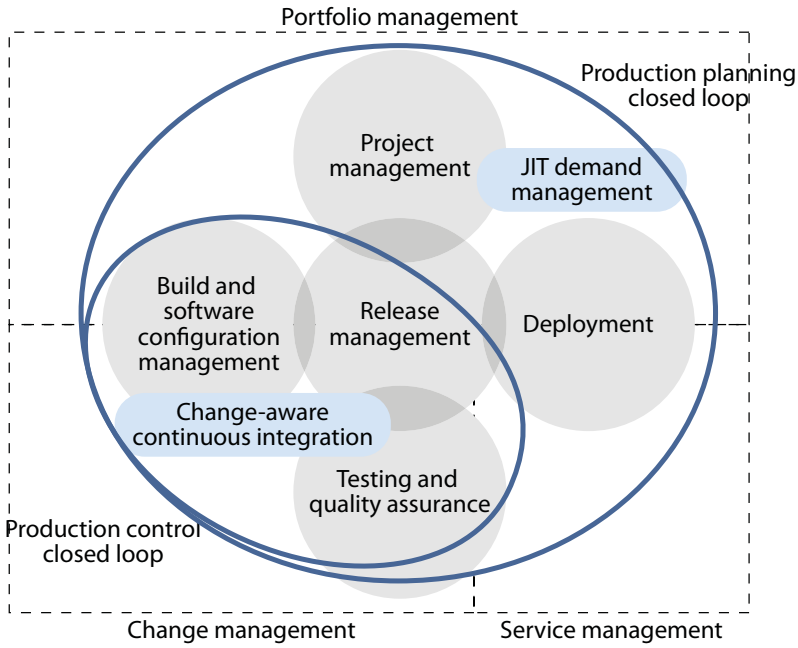
Managing Agile Projects With Two Closed Loops

How can teams best automate Agile development at scale? In our research, Forrester has found that, to scale Agile, teams should focus on two key process best practices (see Figure 3):

- **Projects must implement change-aware continuous integration (CI).** Ask any ScrumMaster worth her salt, and she'll readily tell you of the importance of continuous integration — integrating, building, and testing source code changes early and often to reduce rework and integration issues. But continuous integration is not enough: Effective production control requires teams to track how source code changes are related to individual defect fixes or enhancement requests. And it's even harder to manage a basic CI loop if you're using parallel development techniques with source code changes propagated across multiple baselines. Effective integration between change management and build and release tools allows project managers to easily answer questions such as "In which release will defect 5479 be fixed?" or "How many user stories have we delivered in the current build, and how many source code files did we touch?"

- **Application delivery leaders must implement just-in-time (JIT) demand management.** As Agile projects increase their velocity, it becomes even more important to make sure that they are building what business sponsors need. This means that teams can't connect with the business only occasionally, as they might in a waterfall or iterative process. Rather, development and delivery leaders must implement a "just-in-time" planning loop that connects business sponsors to project teams at frequent intervals to pull demand and, if necessary, reprioritize existing project tasks based on the latest information available. This is easier said than done. Traditional budgeting processes and portfolio management tools tend to focus on high-level objectives and yearly project cycles that are poorly connected to an Agile project's task management burndown list.

Figure 3 Two Closed Loops Drive Agile Automation



48153 Source: Forrester Research, Inc.

Dashboards Enable Visibility And Progress

Measurement and software development have historically been poor bedfellows; heated debates abound about the value of measuring x or y on development projects.[2] Agile changes this with a clear focus on progress, quality, and status metrics. It also changes who is interested in measures, making measurement one of the team's key responsibilities. This increased focus on dashboards requires teams to provide:

- **Progress information on tasks.** The team creates tasks and selects them for work, with individuals committing estimates and reporting progress against this work. Tasks become the primary unit of discussion in daily Scrum meetings. Tasks are also linked with other artifacts such as builds and test results.

- **Linkage between project artifacts and status information.** Project status is greatly affected by the status of key project artifacts such as tests, builds, and code. Agile projects require that teams report this information in a timely manner in a way that shows both the status and state of these artifacts. For example, teams must report the status of the build and its relationship with completed tests. This information allows the team to see which tests are outstanding and which have been completed. By aggregating this information across the project, the team can understand the project's true status.

- **Real-time information accessible by all.** The development team wants to know status in order to steer the project, but team members are not the only stakeholders who care about status. Cross-project dependencies, customer visibility, and the requirements of other external groups also require project status to be available in many different forms.

"Scrum, But . . . " Requires Process Customization

"I am using Scrum, but . . ." is often the way application development professionals describe their Agile process. In fact, one of Agile's strengths is that it encourages teams to select the Agile practices relevant to their particular situation. The result is that an individual process instance may look different from implementation to implementation, and teams can even combine traditional methods with more-Agile practices to create a hybrid approach. Hybrid approaches may impose more process rigor or control for certain activities. For example, a particular approach might dictate that a story cannot be marked done until a code review is undertaken or until test-coverage tools are executed. Hybrid approaches require:

- **Process-flow customization.** By adding control points in a task or story, it is possible to provide explicit control for a particular process flow. The type of story or task may also influence its process, with architecturally significant tasks having a different process flow than tasks associated with less-significant requirements.

- **Improved tool integration.** By tightly linking development tools such as those for code coverage, build management, and testing, it is possible to automate the process more explicitly and gather status information throughout its execution.

- **Customized reports and dashboards.** A Scrum-based, backlog-driven approach may form the basis of the daily reports, but many Agile teams augment standard burndown and velocity with other information associated with the process, such as milestones, build stability, and test coverage and status.

Frequent Planning Requires Integration

Planning within Agile projects happens on at least three levels: 1) product- or release-level planning; 2) sprint or iteration planning; and 3) individual planning. Planning also happens more frequently in Agile projects than in traditional ones, but differently. Mary Poppendieck describes the difference, explaining, "On Agile projects we like planning but do not like plans."[3] Frequent multilevel planning either pushes planning entirely out of traditional project management tools and into ALM tooling or else requires tight integration between project management and ALM tools. Agile project planning requires:

- **The ability to plan at many levels.** In addition to three levels of planning, many Agile projects extend the number of plans, adding program and product road maps. This requires many different views of planning elements and the ability to aggregate those elements into a high-level planning element.

- **Support for collaborative planning techniques.** Traditional planning is often done by one person who gathers input from subject-matter experts and then builds out the plan. Because of the frequency of planning activities, Agile techniques encourage a more collaborative approach to the planning discipline. Techniques such as "Planning Poker" may supplement traditional planning meetings.[4]

- **A frequently updated visual representation of the plan.** The traditional approach of printing out the Gantt chart and taping it to the wall does not work when the plans are constantly being updated. Instead, the updated plan should be visible to all parties involved in the project, enabling them to make decisions based on the most up-to-date view, which reflects what the team has learned so far.

- **Daily descriptions of tasks' status from teams.** With Agile, actual effort is recorded and contributes to reports such as velocity and burndown. This requires integration between the planning tool and the work the team is doing and encourages teams to capture actuals within the context of the integrated development environment (IDE), testing tool, or requirements tooling.

AGILE DEVELOPMENT MANAGEMENT TOOLS EVALUATION OVERVIEW

To assess the state of the Agile development management tools market and see how the vendors stack up against each other, Forrester evaluated the strengths and weaknesses of the top 10 vendors. In performing our analysis, we found that the vendors came from two different places:

- **Historic ALM vendors that have moved into the Agile market.** As Agile adoption continues to increase, ALM vendors continue to expand their tools' reach, adding explicit support for Agile and Agile-like processes.

- **Agile project management tools expanding their reach into the ALM space.** The Agile tools market has its share of vendors that provide explicit support for Agile team approaches. Increasingly, these vendors are taking their products in the direction of a broader ALM offering that can support pure Agile as well as hybrid approaches.

The Evaluation Criteria Focused On Managing, Executing, And Reporting On Agile Projects

After examining past research, user need assessments, and vendor and expert interviews, we developed a comprehensive set of evaluation criteria. We evaluated vendors against 152 criteria, which we grouped into three high-level buckets:

- **Current offering.** We evaluated the vendors against 117 criteria focused on core and advanced functionality, including project setup, project and portfolio planning, project execution, project reporting, and process customization.

- **Strategy.** To determine the vendors' vision, we assessed 20 strategy-related criteria, including the way in which enhancements are planned, vendors' own internal use of their tools, price, commitment, and history.

- **Market presence.** To evaluate the vendors' penetration in the current Agile development management market, we evaluated 15 market-presence-related criteria, including revenue, revenue growth, installed base, support, and regional focus.

Evaluated Vendors Have Healthy Growth Or Strong Market Presence And A Focus On Agile

Forrester included 10 vendors in the assessment: Atlassian, CollabNet, HP, IBM, Micro Focus, Microsoft, MKS, Rally Software Development, Serena Software, and VersionOne. Each of these vendors has (see Figure 4):

- **Healthy growth or strong market presence.** The vendors have disclosed either publicly or in confidence that their 2007 and 2008 revenue amounts showed a growing customer base or strong market presence.

- **Experience serving large enterprises.** To be included in the evaluation, vendors must have a strong focus on and track record with companies that have more than 1,000 employees and large endeavors or programs with teams of teams working on software development.

- **A focus on Agile/Lean development.** An Agile/Lean development process is the focus of this assessment; therefore, we vetted players with a strong focus on serving this process model. Tools show support for Agile processes by providing explicit support for Scrums, product backlog, and other Agile terms and/or by including a large amount of material describing how to use the tool in the context of an Agile process.

Figure 4 Evaluated Vendors: Product Information And Selection Criteria

Vendor	Product evaluated	Product version evaluated	Version release date
Atlassian	JIRA Studio	1.5	December 2008
	JIRA	3.13	September 2008
	GreenHopper	3.0	July 2008
	FishEye	2.0	June 2009
	Crucible	2.0	June 2009
	Bamboo	2.2	March 2009
	Clover	2.5	May 2008
	Atlassian IDE Connector for Eclipse	1.0	May 2009
	Atlassian IDE Connector for IntelliJ IDEA	1.5	June 2008
	Atlassian Crowd	1.6	December 2008
CollabNet	CollabNet TeamForge	5.2	April 2009
	CollabNet Enterprise Edition	5.3	May 2009
	CollabNet Subversion	1.6	March 2009
	CollabNet Lab Management	2.2	April 2009
HP	HP Quality Center Agile Accelerator	10.0	May 2009
IBM	Rational Team Concert	2.0	June 2009
Micro Focus	TeamFocus	2008	September 2008
	TeamDemand	2008	April 2009
	TeamInspector	2008	February 2009
	TeamAnalytics	2008 R2	March 2008
	StarTeam	2008 R2	September 2008
	CaliberRM	2008	March 2009
	SilkCentral Test Manager	2008	April 2008
Microsoft	Visual Studio Team System 2008 Team Foundation Server	2008	October 2007
	Visual Studio Team System 2008 Team Foundation Server Client Access License	2008	October 2007
	Visual Studio Team System 2008 Architect Edition	2008	October 2007
	Visual Studio Team System 2008 Development Edition	2008	October 2007
	Visual Studio Team System 2008 Database Edition	2008	October 2007
	Visual Studio Team System 2008 Test Edition	2008	October 2007
	Visual Studio Team System 2008 Team Suite	2008	October 2007
	Visual Studio Team System 2008 Test Load Agent	2008	October 2007
MKS	MKS Integrity	2009	July 2009
	MKS Integrity for IBM i	2009	July 2009
Rally Software Development	Rally Enterprise Edition	2009.3	June 2009
	Rally Community Edition	2009.3	June 2009
	Rally Quality Manager	2009.3	June 2009
	Rally Support Manager	2009.1	March 2009
	Rally Product Manager	2008.1	March 2008
	Rally Community Manager	2009.3	June 2009

Source: Forrester Research, Inc.

Figure 4 Evaluated Vendors: Product Information And Selection Criteria (Cont.)

Vendor	Product evaluated	Product version evaluated	Version release date
Serena Software	Serena Agile	2009 R3	July 2009
	Serena Business Mashups	2009 R1	July 2009
VersionOne	Agile Enterprise	9.1	July 2003
	Agile Team	9.1	May 2008
	Agile Platform	9.1	July 2006
	Agile Ideas	9.1	May 2009

Vendor selection criteria
The vendor shows healthy growth or a strong market presence.
The vendor has experience serving large enterprise vendors.
The vendor has a focus on Agile/Lean development.

Source: Forrester Research, Inc.

THE FORRESTER ADM WAVE REVEALS LEADERS AND STRONG PERFORMERS

Forrester's evaluation of Agile development management tools reveals a vibrant, competitive, and changing market consisting of six Leaders and four Strong Performers. The merging of traditional ALM features with Agile project management highlights that the Agile project portfolio management (PPM) market and ALM markets are consolidating. Many vendors continue to invest in program and project management as well as reporting and integration with development, testing, build, and deployment tools. The resulting solution provides a task-driven, Agile-oriented development management platform that consolidates planning, status, and real project metrics in one place. It also provides visibility into requirements, defects, and change requests for a system or product, allowing broader application life-cycle activities to be consolidated within the platform. The evaluation uncovered a market in which (see Figure 5):

- **MKS and IBM provide strong current offerings.** MKS excels in process configuration, security, and integration, while IBM demonstrates strength in the areas of undertaking work and task management. Both companies excel in the area of reporting and analytics, an increasingly important focus for organizations that assign a high value to their ability to deliver software.

- **Atlassian, CollabNet, and Microsoft have strong strategy.** Despite their strong strategies, these three vendors have weaker current offerings. Atlassian continues to broaden its engineering-oriented portfolio, while Microsoft Visual Studio 2010 will add to an already strong application life-cycle management tool set with project templates specifically focused on Agile delivery.

CollabNet continues to broaden its management offering through both acquisition and development by providing stronger support for Agile project and portfolio management and better integration with development, testing, and build tools.[5]

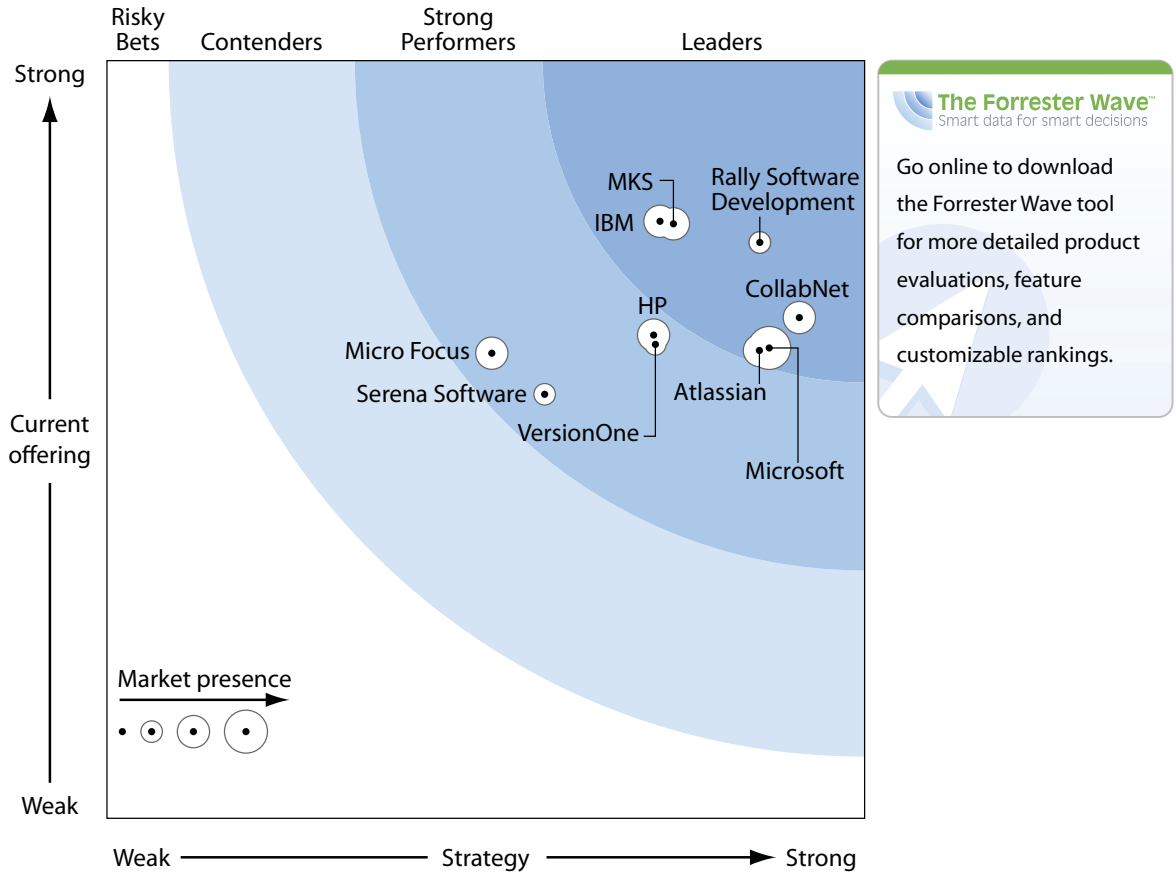
- **Rally offers the best combination of capability and strategy.** In our evaluation, Rally provides Agile project teams the strongest combination of current offering and strategy. In the current offering area, Rally in particular shows strong support for Agile project and release management. Like MKS and IBM, Rally provides strong reporting and analytics.

- **HP and VersionOne offer competitive options.** VersionOne's current offering is as strong as many in this space, but the vendor's lack of a demonstrated strategy and customizable reporting to support Agile reduces its current offering scores. HP's current offering, based on the Quality Center platform, has many strengths; however, until it offers clear integrations with other engineering tools and a much-improved analytics and reporting capability, its offering cannot offer as much support for large-scale, complex Agile implementations.

- **Micro Focus and Serena could be very strong contenders.** A recent acquisition has supplied Micro Focus with many of the parts necessary to build a credible and market-leading product in this space. However, at the time of the evaluation, it was difficult to see what Micro Focus' post-acquisition strategy would look like. Micro Focus' current offering score reflects Borland Software's lack of clear strategy over the past two years. Serena a presented a new offering to the Agile marketplace, which, though providing good support for Agile teams, misses the mark in terms of breadth, supporting Agile in a broader context and depth by providing integrations with practitioner tools.

This evaluation of the application development management tools market is intended to be a starting point only. We encourage readers to view detailed product evaluations and adapt the criteria weightings to fit their individual needs through the Forrester Wave™ Excel-based vendor comparison tool.

Figure 5 Forrester Wave™: Agile Development Management Tools, Q2 '10



Source: Forrester Research, Inc.

Figure 5 Forrester Wave™: Agile Development Management Tools, Q2 '10 (Cont.)

	Forrester's Weighting	Atlassian	CollabNet	HP	IBM	Micro Focus	Microsoft	MKS	Rally Software Development	Serena Software	VersionOne
CURRENT OFFERING	50%	3.09	3.30	3.17	3.93	3.07	3.10	3.92	3.80	2.80	3.13
Products included	0%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Platform support	5%	3.55	4.00	3.60	3.95	1.55	2.40	3.55	2.45	1.95	2.15
Management	15%	1.83	2.98	3.20	3.43	3.95	2.35	3.13	4.60	2.20	4.23
Running a project	25%	3.17	3.12	3.45	4.17	2.97	2.93	4.34	3.56	2.92	3.32
Administration	5%	3.50	4.00	4.50	4.00	4.50	4.50	3.50	5.00	5.00	4.50
Security	5%	3.94	3.79	3.21	3.63	3.40	3.04	5.00	3.11	2.30	2.94
Process configuration	10%	2.60	2.80	3.50	4.20	2.90	4.30	4.50	2.40	3.20	2.80
Analytics	15%	4.58	4.34	3.54	4.64	4.16	4.10	4.40	4.58	4.70	1.98
Life-cycle integration	20%	2.62	2.75	1.90	3.38	1.74	2.35	3.27	3.81	1.25	3.07
STRATEGY	50%	4.31	4.57	3.61	3.66	2.54	4.36	3.74	4.31	2.89	3.62
Product strategy	45%	4.40	4.40	3.60	3.60	2.30	5.00	3.70	4.40	3.00	3.00
Corporate strategy	20%	4.00	4.50	3.00	3.00	1.50	3.50	4.00	4.00	2.00	4.00
Price	0%	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Commitment	30%	4.60	4.80	3.80	4.80	4.00	4.20	3.40	4.60	3.80	4.40
History	5%	3.00	5.00	5.00	0.00	0.00	3.00	5.00	3.00	0.00	3.00
MARKET PRESENCE	0%	3.48	3.53	3.84	3.87	3.92	4.26	3.38	2.54	2.81	2.40
Installed base	20%	4.60	4.60	4.30	3.05	1.95	4.25	2.70	3.80	1.00	3.50
Financial strength	15%	3.00	2.50	3.00	3.00	4.00	2.50	1.50	3.00	2.50	3.00
Employees	5%	1.00	1.00	3.00	3.00	3.00	3.00	1.00	1.00	1.00	1.00
Support services	30%	2.80	4.60	4.60	4.60	5.00	4.60	4.60	2.20	4.60	2.20
Channel partnerships	10%	5.00	2.00	0.00	4.00	4.00	5.00	3.00	3.00	1.00	1.00
Global presence	20%	3.60	3.00	5.00	4.40	4.40	5.00	4.40	1.60	3.50	2.20

All scores are based on a scale of 0 (weak) to 5 (strong).

Source: Forrester Research, Inc.

VENDOR PROFILES

Leaders: Atlassian, CollabNet, IBM, Microsoft, MKS, And Rally Software Development

- **Atlassian adds to JIRA with comprehensive Agile project management capabilities.** JIRA provides a solid platform to extend change management into a more comprehensive Agile development management offering. Atlassian, via its acquisition of GreenHopper, added dashboards and planning capabilities to the very popular change management tool JIRA. By adding other Atlassian products to the mix, such as Confluence for collaboration and Bamboo for continuous integration, Atlassian provides a comprehensive development solution. Though the solution is aimed at software engineers, Atlassian continues to invest in this product family, adding additional capabilities for engineers while broadening the appeal of the products to a much wider software development audience.

- **CollabNet extends its platform with strong support for distributed Agile development.** Based on its experience with open source development and the Subversion configuration management tool, CollabNet has strong support for support for Agile project management and task management. Because of its distributed heritage, it offers a very secure platform with strong support for encryption and authorization. Its recent acquisition of Danube adds to its strategy for Agile development, providing additional thought leadership and development capability.

- **IBM, focusing on collaborative development, adds strong project management and analytics.** Based on the Eclipse and Jazz platforms, IBM Rational continues to raise the bar on building a complete development and delivery platform. With offerings for IBM System z and IBM System i, IBM's tool set has the most platform support. Its integration into Eclipse is very strong, providing comprehensive support for distributed Agile teams. IBM's focus on task management provides a great foundation for modern engineering practices, allowing integration into project management and engineering practices, as well as the ability to capture long-term metrics and analytics. In parallel with the development of IBM Rational Team Concert, IBM developed an open integration standard: Open Services for Lifecycle Collaboration (OSLC).[6]

- **Microsoft provides the most comprehensive platform for .NET development.** With extensive project support and comprehensive integration into the Visual Studio development environment, Team Foundation Server continues in the tradition of Microsoft products with an easy install and simple configuration. Out-of-the-box process configurations supporting Agile and other popular process models make adoption simpler. The VS2010 release, which was not evaluated, demonstrates a firm commitment to Agile with improvements to planning, reporting, and task management.

- **MKS provides a robust development management solution.** MKS provides extensive task and workflow management coupled with good life-cycle integration. The MKS Integrity platform is the most secure product we evaluated, offering comprehensive support for encryption, authorization, and electronic signatures, making it very attractive for industries where

compliance and audit are high priorities for developers. Scoring for MKS Integrity suffered a bit because MKS does not use a comprehensive Agile development approach. MKS uses an iterative approach for major releases and a Scrum-like Agile method for small or patch releases. MKS does offer an Agile, Scrum-like template out of the box, which, coupled with the very flexible and extendable workflow model, would provide strong support for organizations that follow a "Scrum, but . . ." or hybrid Agile approach.

- **Rally Software continues to extend its Agile heritage.** Coming from an Agile project management background, Rally continues to add functionality to extend the reach of its development management environment. This includes project and portfolio management with resource management and tracking; demand management with improved ways to capture and prioritize demand from customers; and life-cycle integration with strong integration with multiple configuration management tools. Rally continues to develop strong thought leadership around the practice of software delivery, and it has a strong services group with lots of experience around enterprise Agile adoption, benchmarking, and assessments.

Strong Performers: HP, Micro Focus, Serena Software, And VersionOne

- **HP builds on its heritage of testing with its Agile Accelerator configuration.** Built on the HP Quality Center product, the Agile Accelerator provides a configuration that enables Agile teams to quickly start work, providing out of the box a set of customizations for work management, workflow, task management, and reporting. Because of its deep roots in testing, the Agile Accelerator provides easy integrations into the testing discipline and with associated tools. Other integrations into portfolio management and service desk enable Agile teams to take advantage of application and product knowledge. Integrations with tools outside the HP stack were much weaker; HP scored the lowest of all evaluated tools in life-cycle integrations. However, tool integration is a key part of HP's long-term strategy, and Forrester expects to see improved integrations, with a particular focus on source code and configuration management.

- **Micro Focus could build out an offering that appeals to client-server and legacy developers.** The paint of the acquisition was still wet when we evaluated TeamFocus, TeamDemand, TeamInspector, TeamAnalytics, StarTeam, CaliberRM, and SilkCentral Test Manager products, formerly of Borland but now part of the Micro Focus portfolio. The product set provides strong support for managing an Agile team and includes good reporting capabilities. The data warehouse aspect of the offering shows promise but lacks prebuilt integrations, relying instead on the team to build out its own information requirements. To support the evaluation, numerous products had to be combined, and integration between these products proved complex and sometimes nonexistent. This demonstrates a lack of strategy across the product line, which Micro Focus is in the process of resolving.

- **Serena enters the Agile market with a strong focus on Scrum team management.** By hiring a number of key people from the Scrum movement, Serena built from the ground up a product aimed at helping teams work more effectively in Scrum projects. Serena Agile's interface is easy to navigate and provides support for running a project. Because of its hosted nature, Serena Agile is easy for teams to set up. Analytics and reporting is another strong area for Serena Agile. The product lacks depth in the areas of life-cycle integration and planning. Serena Agile is a new product with a limited customer base; however, by combining its experience in change management with its business information mashup tool, Serena could build on its new-entrant product to move into a more favorable position in future comparisons.

- **VersionOne extends its Agile project management capabilities with improved integrations.** VersionOne delivered one of the first tools that supported Agile development projects, providing support for planning, reporting, and execution for distributed Agile teams. It continued to add to this thought leadership with a broader project/portfolio management offering coupled with improved integrations with other development tools. Its support for the Agile community extends into a very active community and key sponsorships for a number of face-to-face events. This provides clear feedback that it can apply to its product strategy to ensure that its products stay in line with current Agile thinking and best practices.

A Wide Range Of Product Pricing Indicates A Market In Transition

As ALM vendors add Agile capabilities and Agile planning vendors integrate more deeply with developers tools, it's clear that two market segments are collapsing into one. One classic hallmark of a market in convergence is that price/value ratios fluctuate as new vendors challenge existing market leaders for share. In the Agile development management space, these normal fluctuations are intensified by the complete commoditization of individual ALM tool segments such as software configuration management (SCM) and build. The result? The expected license and maintenance costs for a team that is just getting started vary wildly (see Figure 6). For example, a 10-person development team with 30 occasional users can get started for around $6,100 per year over three years with Atlassian (or as little as $10 a year if casual users can get away with read-only access), while it will cost the same team $26,400 per year to use MKS.[7] While the Forrester Wave methodology does not allow a solution's cost to factor into the evaluation process, we nonetheless believe that development teams should consider it when building a shortlist for further product evaluations. In particular, when evaluating Agile development management solutions:

- **Consider the impact of casual users**. Developers and testers tend to use ADM tools for hours every day and need a dedicated license. But other users, such as business sponsors or project managers, may need to access these tools far less frequently. These casual users can significantly add to the cost of deploying a project if each requires a dedicate license. If your organization has a lot of casual users, prioritize products that include floating license options or low-cost read-only licenses.

- **Note that pricing curves are not linear.** If you plot the prices of various ADM solutions across team size, you'll notice that they aren't strictly linear. The cost of additional server licenses or low-cost entry additions that top out at a dozen users can entice a small team but bring long-term higher costs to the entire organization. Also consider the impact that server-based pricing may have. If you can efficiently load up one large server, then you can limit your total licensing costs; however, this may not be possible if you have separate teams with their own development infrastructures.

- **Don't ignore application platform affinity.** The real price may vary depending on what tools and runtimes you already own. For example, if your organization already maintains Microsoft Developer Network (MSDN) premium licenses, then you already have client access licenses (CALs) that allow developers to access Microsoft Team Foundation server.

Figure 6 Annualized Costs For Three Years

Vendor (all prices per year)	**Small team** (10 daily, 30 casual users)	**Medium team** (50 daily, 80 casual users)	**Large team** (200 daily, 500 casual users)
Atlassian	$6,100	$12,800	$22,300
CollabNet	$5,894 (plus casual user licenses)	$27,291 (plus casual user licenses)	$158,275 (plus casual user licenses)
HP	Did not disclose	Did not disclose	Did not disclose
IBM	$6,615	$47,417	$409,450
Micro Focus	Did not disclose	Did not disclose	Did not disclose
Microsoft	$10,300	$30,543	$147,504
MKS	$26,400	$68,000	$233,000
Rally Software Development	Free (plus casual user licenses)	$15,000 (plus casual user licenses)	$50,000 (plus casual user licenses)
Serena Software	$6,300	$51,840	$409,860
VersionOne	$13,920	$45,240	$243,600

48153 Source: Forrester Research, Inc.

SUPPLEMENTAL MATERIAL

Online Resource

The online version of Figure 5 is an Excel-based vendor comparison tool that provides detailed product evaluations and customizable rankings.

Data Sources Used In This Forrester Wave

Forrester used a combination of three data sources to assess the strengths and weaknesses of each solution:

- **Hands-on lab evaluations.** Vendors spent one day with a team of analysts who performed a hands-on evaluation of the product using a scenario-based testing methodology. We evaluated each product using the same scenarios, creating a level playing field by evaluating every product on the same criteria.

- **Vendor surveys.** Forrester surveyed vendors on their capabilities as they relate to the evaluation criteria. Once we analyzed the completed vendor surveys, we conducted vendor calls where necessary to gather details of vendor qualifications.

- **Customer reference calls.** To validate product and vendor qualifications, Forrester also conducted reference calls with two of each vendor's current customers.

The Forrester Wave Methodology

We conduct primary research to develop a list of vendors that meet our criteria to be evaluated in this market. From that initial pool of vendors, we then narrow our final list. We choose these vendors based on: 1) product fit; 2) customer success; and 3) Forrester client demand. We eliminate vendors that have limited customer references and products that don't fit the scope of our evaluation.

After examining past research, user need assessments, and vendor and expert interviews, we develop the initial evaluation criteria. To evaluate the vendors and their products against our set of criteria, we gather details of product qualifications through a combination of lab evaluations, questionnaires, demos, and/or discussions with client references. We send evaluations to the vendors for their review, and we adjust the evaluations to provide the most accurate view of vendor offerings and strategies.

We set default weightings to reflect our analysis of the needs of large user companies — and/or other scenarios as outlined in the Forrester Wave document — and then score the vendors based on a clearly defined scale. These default weightings are intended only as a starting point, and we encourage readers to adapt the weightings to fit their individual needs through the Excel-based tool. The final scores generate the graphical depiction of the market based on current offering, strategy, and market presence. Forrester intends to update vendor evaluations regularly as product capabilities and vendor strategies evolve.

Survey Methodologies

The Forrester/Dr. Dobb's Global Developer Technographics® Survey, Q3 2009, was fielded to 1,298 application development and program management professionals who are readers of Dr. Dobb's magazine. For quality assurance, respondents are required to provide contact information and answer basic questions about themselves. Forrester fielded the survey from July 2009 to August 2009. Respondent incentives included a summary of the survey results and a chance to win one of five $50 gift certificates.

Forrester fielded its Q3 2009 Global Agile Adoption Online Survey to 60 technology professionals from our ongoing Technology Industry Research Panel. The panel consists of volunteers who join on the basis of interest and familiarity with specific technology industry topics. For quality assurance, panelists are required to provide contact information and answer basic questions about their firms' revenue and budgets. Forrester fielded the survey from August to October 2009. Respondent incentives included a summary of the survey results.

Exact sample sizes for the surveys used in this report are provided on a question-by-question basis. Surveys are not guaranteed to be representative of the entire application development population. Unless otherwise noted, statistical data is intended to be used for descriptive and not inferential purposes.

If you're interested in joining one of Forrester's Research Panels, you may visit us at http://Forrester.com/Panel.

ENDNOTES

[1] Agile's adoption at eBay was described in: Douglas MacMillan, "Can eBay Get Its Tech Savvy Back?" *BusinessWeek,* June 11, 2009 (http://www.businessweek.com/magazine/content/09_25/b4136048144243.htm).

[2] Forrester published a report that discusses this debate, highlighting the value in sizing but describing why it has been historically hard. The same problem occurs in many other aspects of development ranging from quality to architecture. See the July 27, 2009, "Software Size Matters, And You Should Measure It" report.

[3] Mary Poppendieck describes the approach to planning her book: Mary Poppendieck and Tom Poppendieck, *Lean Software Development: An Agile Toolkit*, Addison-Wesley, 2003.

[4] Planning Poker was invented by Mike Cohn and is described in some detail on the Planning Poker Web site (http://www.planningpoker.com/).

[5] CollabNet recently acquired Danube. Danube provided training and tools for Scrum teams with its free and for-sale products ScrumWorks and ScrumWorks Pro. Due to the timing of the acquisition, no products or services from Danube were included in this evaluation.

[6] Open Services for Lifecycle Collaboration (OSLC) is an open standard aimed at making it easier for tool vendors to interoperate by providing a standard set of interface standards based on a RESTful architecture. More details can be found at the Open Services for Lifecycle Collaboration Web site (http://open-services.net/html/Home.html).

[7] In October 2009 Atlassian introduced its 10 for $10 program, where license proceeds go to charity. Since the program's introduction, it has raised $470,000 for Room to Read, Atlassian's designated charity. For more information, see http://www.atlassian.com/starter/.

MARKET MOVER ARRAY™ REPORT:

Testing Platforms

By Theresa Lanowitz, Lisa Dronzek | August 11, 2010

voke®
Moving markets beyond the status quo!

MARKET MOVER ARRAY™ REPORT:

Testing Platforms

By Theresa Lanowitz, Lisa Dronzek | August 11, 2010

SUMMARY

Software has expanded its reach to become responsible for business processes, consumer purchases, transportation, communications, and devices that are always on and, in some cases, life-critical. The stakes of making sure that proper testing occurs at all levels are greater than ever. Testing is a comprehensive and critical part of the entire lifecycle.

Today's business executive must be able to guarantee working software free of defects to avoid compromising business, safety, or security. This Market Mover Array™ report examines the history of the testing market and analyzes the vendors vying to move the market beyond the status quo.

TABLE OF CONTENTS

MARKET MOVER ARRAY OVERVIEW

Software has expanded its reach to every aspect of our lives; it is no longer relegated to a confined area of deployed applications to execute a business task. Software is responsible for business processes, consumer purchases, transportation, communications, and devices that are always on and, in some cases, life-critical.

The tolerance for software problems, especially in the enterprise, has traditionally been quite high. It is widely accepted that software will experience problems. However, as software becomes a necessity in everything that is used, made, touched, and experienced, the stakes of making sure it has been properly tested at all levels are greater than ever. Software problems are now feared by consumers and business executives alike. The realization of the significance of testing and the conversation about it have rapidly moved from the practitioner to the executive.

Just as the role of software has changed, the complexity and necessity of validating its intended and actual use has undergone a transformation. Testing has changed from primarily validating a deployed application focused on business to being a comprehensive and critical part of the entire lifecycle. Today, testing lives in every phase of the lifecycle for all products and applications, regardless of scale and scope.

Today's business executive must be able to guarantee working software that is free of defects to avoid compromising business, safety, or security. This Market Mover Array™ report examines the history of the testing market and analyzes the vendors vying to move the market beyond the status quo.

⊙ TESTING MARKET OVERVIEW

Over the past several years, the testing market has been stagnant. Market leader HP (with the former Mercury offerings) set the tone for what was acceptable and needed in the market.

Since 1999, the two most significant market-moving events have been:

- Rational's (now IBM) introduction and broad acceptance of the Rational Unified Process (RUP)
- Mercury's introduction of Business Technology Optimization (BTO)

Each of these two events created a rumbling in the market, making the industry take notice and competitors react.

Since Mercury's introduction of BTO in 2002, innovation in the testing market has been stalled. Mercury owned and defined what testing was, who should use it, and how it should be considered. The market allowed Mercury to claim its role as the victor. There were

many competitors, but all followed Mercury's lead which allowed Mercury to retain its top position unopposed.

Arguably, times have changed since the last spark of innovation rolled out of Mercury. In 2006, HP moved in and paid $4.5 billion for Mercury and its title of testing market champion. Additional major market consolidation has occurred, with IBM acquiring both Rational for $2.1 billion and Telelogic for $845 million, and Micro Focus acquiring Borland (owner of the Segue technology) and Compuware's testing assets for a total of $115 million in 2009. Acquisitions of Mercury, Rational, Segue, and the Compuware testing assets meant the traditional testing vendors were now part of larger entities with far more than testing offerings in their product portfolios. Still, the market is beholden to the definition of testing that Mercury established and the standards Mercury set for marketing, support, and overall customer experience.

In the 2007 to 2008 timeframe, creative startups with new and complementary offerings to the traditional HP solutions began to emerge. The testing market was subtly starting to awaken from its slumber. Virtualization was taking hold of the landscape and startups were moving in to capitalize where HP and the old guard had left off. Developers were in need of testing solutions beyond unit testing, the line of business demanded that requirements be given attention in testing, and mobile applications became commonplace. Software, and the need to test it, was exploding well beyond the norms defined by HP.

In the midst of this testing market awakening, Microsoft announced Visual Studio 2010 with a focus on testing. Microsoft's announcement caused the market to pause. In 2004, when Microsoft entered the market with an organic and integrated application lifecycle solution, it was assumed that Microsoft would help shape the testing landscape. For a number of reasons, Microsoft's early entry did not gain the expected market traction. However, Visual Studio 2010, still organic and integrated, is focused on making the testing experience modern.

We believe that Microsoft's entrance to the market with Visual Studio 2010—with virtual lab technology integrated into that platform—will be as significant as the entries of RUP and BTO. voke predicts that virtual lab technology will be the hub of the modern application lifecycle (see *voke Market Snapshot™ Report: Virtual Lab Management* – March 22, 2010).

Additionally, the testing market is no longer solely defined by the need to test business applications. New vendors in the market are offering solutions well beyond the traditional and narrow confines of the market. We are embarking on the early stages of convergence between embedded systems and IT.

STATE OF THE TESTING MARKET

In mid-2010 this is the state of the testing market:

- HP is the undisputed market share leader of traditional testing tools focused on testers testing business applications.
- HP faces challenges from smaller vendors with less expensive and innovative solutions.
- HP faces challenges from larger vendors that see the testing component of the application lifecycle as critical. HP is unique among the larger test vendors with its defacto platform independence.
- HP's Performance Center with its flagship LoadRunner product is the gold standard for application performance testing. Putting LoadRunner in the cloud—where the economics make this technology available to organizations of all sizes and all budgets—is the most significant move by HP since the acquisition of Mercury.
- IBM and MKS are making concerted efforts to attract traditional testing customers with end-to-end lifecycle solutions.
- Microsoft has launched a solid and innovative product and is counting on the loyalty of the Microsoft developer universe to entrench the Visual Studio brand in the testing community.
- Microsoft is revolutionizing manual testing with powerful new solutions to increase productivity and break down the barriers between development and testing. The market is taking notice of Microsoft, sparking both future innovation and marketing reactions.
- The types of software in need of testing have changed and expanded. Traditional larger vendors do not have all of the solutions for the new demands placed upon testing.
- Vendors such as Micro Focus and Original Software are evolving their product lines to meet modern testing demands and offer compelling alternatives to the market-leading platforms.
- Convergence of traditional IT applications, embedded systems, and device software is emerging, and vendors are offering robust and proven solutions. IBM made the move into systems with its Telelogic acquisition, the MKS platform spans both the enterprise and engineering organizations, and Fanfare specializes in a testing platform for embedded systems and devices. Coverity, Electric Cloud, and Klocwork deliver solutions beyond traditional IT into systems and engineering organizations.
- Electric Cloud is unique in that its customers are driving the need to extend the Electric Cloud solution to be an integrated framework for build/test/deploy for both development and QA. Electric Cloud delivers an additive solution for organizations

using other test platforms, or provides a standalone solution to organizations just embarking on automation.

- Coverity is the first vendor since Mercury to leverage the power of a business message to expand its customer base and change the conversation from a technical and developer-centric message about code analysis to an explanation of the business necessity of delivering quality, safe, and secure software. This is the first sign of marketing targeted beyond the practitioner that the market has witnessed in many years. Moving to a business-focused marketing message amplifies the importance of testing for the overall health of the business.
- Professional services spin-offs, QMetry and TOMOS, lead the innovation in the cloud with modern testing platforms, delivering ease of use and the benefits of cloud economics.
- While virtualization is prevalent in the data center, it has yet to make a broad impact in the testing market (see *voke Market Snapshot™ Report: Virtual Lab Management* – March 22, 2010). Microsoft and Replay Solutions are leading innovation in the testing market by leveraging the power of virtualization to simplify complex development and testing challenges. Microsoft offers the only integrated virtual lab solution in its full-featured lifecycle solution. Replay delivers the power of rapid defect replication and identification via virtualization through either an on-premise or cloud solution. Replay's product provides an additive solution to any test platform and is a must-have for organizations developing Java applications regardless of the presence or absence of other testing solutions.
- While HP is the undisputed market share leader, the testing market is on the cusp of major transformation. The market is ready to accept the next true leader that emerges and is capable of advancing the market.

The testing market is finally awakening from its long, quiet slumber. Powerful new technology has emerged to simplify complex challenges related to software development and testing. From an innovation perspective, this is one of the most exciting times in the history of the testing market. Only time will tell which new leader will emerge as the preeminent thought leader and marketer capable of delivering a business message and able to carry the mantle as the voice of testing.

HP with its undisputed market share leadership and Microsoft with the most complete set of organic and integrated technology for testing are about to square off. At this point, Microsoft has a newly-created testing solution that delivers solutions for many of the most profound testing problems. HP with its significant install base must continue to expand its technology to allow its customers to preserve and extend their testing investments.

August 11, 2010

Both companies are aware of what must occur to retain and gain customers from the innovation and technology perspective. It is in the marketing area where both HP and Microsoft must focus efforts to convince customers that their businesses will thrive through the investment in one of these software titans. HP and Microsoft will ultimately battle for a significant portion of the testing market. Which company will be market savvy and bold enough to engage in marketing efforts to move the current testing market beyond the status quo? The answer will unfold over the next several years and will be decided certainly upon technology and innovation, but ultimately upon the ability of one of these vendors to lead and define the market in a way that connects with the demands and needs of not only the testing organization, but the entire business.

The market is in need of a "new Mercury". The market needs a vendor capable of delivering the necessary technology, but more importantly, a vendor capable of changing the conversation to a business conversation about the importance of software to the business. We believe the market needs an infusion of boldness to move and redefine the market. Watch this market closely as both large and small vendors attempt to change the discussion to reflect the needs of the business.

⊙ MARKET MOVER ARRAY — LOOKING FORWARD

The voke Market Mover Array: Testing Platforms is a new and unique way of examining vendors in the testing market. voke believes that going against the grain to analyze where the market is headed, or needs to be headed, provokes the right questions and answers for the future. Because the market share leaders are by default well known, making predictions and providing analysis of where the market *will* go provides a valuable new perspective. Innovation is required to deliver what is necessary for the future, and adept marketing is essential to change the point of view. Analysis that is based primarily on market share is indicative of what has already happened in a market. While market share is important, it is not the defining component in voke's analysis. Instead, this report focuses on the current and future state of the testing market, and analyzes vendors based on innovation and technology, as well as marketing ability. These factors are critical to moving markets beyond the status quo.

Innovation can occur inside any organization, large or small. When a market has reached a point of needing new technology to deliver solutions to new problems, innovation will be sparked. In this report, we look at vendors in the testing market that are delivering innovative technology. And, because innovation can occur in any size organization, some of the vendors we discuss do not have significant market share, but, do have significant innovation. The market needs such innovation to move forward.

August 11, 2010

As history has repeatedly shown, innovation and technology must always be accompanied by marketing. In this report, we examine how testing vendors are shaping the conversation in the market through marketing efforts. Testing has long been perceived as a tactical component of the lifecycle. However, the critical nature of software and the impact it has on the success of business is advancing the testing initiative from the practitioner to the executive. Marketing is a critical element of how the vendor and its products are perceived and ultimately purchased.

⊙ MARKET MOVER ARRAY METHODOLOGY

The Market Mover Array is research designed to identify notable vendors in a particular market and is a core piece of voke's research taxonomy. voke brings independence and uniqueness to viewing markets in a dynamic and forward-moving state.

The Market Mover Array is plotted against two axes: "Innovation and Technology" and "Marketing Ability". Each of these axes contains seven components against which the represented vendors were rated.

The Innovation and Technology components are:

- Product
- Technology
- Ease of use
- Product works as advertised
- New technology solving a classic problem
- Easy solution to complex problems
- Integration

The Marketing Ability components are:

- Product offerings
- Pricing
- Positioning
- Promotion
- Thought leadership
- Execution
- Executive leadership

August 11, 2010

The result of each vendor rating was charted and placed into one of four bands on voke's Market Mover Array: Testing Platforms (Figure 1).

The four bands of the Market Mover Array are:

Transformational — vendors that are changing the tone and direction of the market. These vendors may include newer entrants to the market that have either innovative technology or the ability to take a long view of the market and articulate it. These vendors are typically challenging the pivotal vendors to innovate either in terms of technology or marketing acumen.

Pivotal — vendors that are crucial to the continuation and evolution of the market through their customer base, thought leadership, or technology innovation. Vendors with significant market share are under immense pressure to maintain status in the market through innovation. New and emerging vendors with compelling technology are vying for broader market awareness and solidifying their marketing voice.

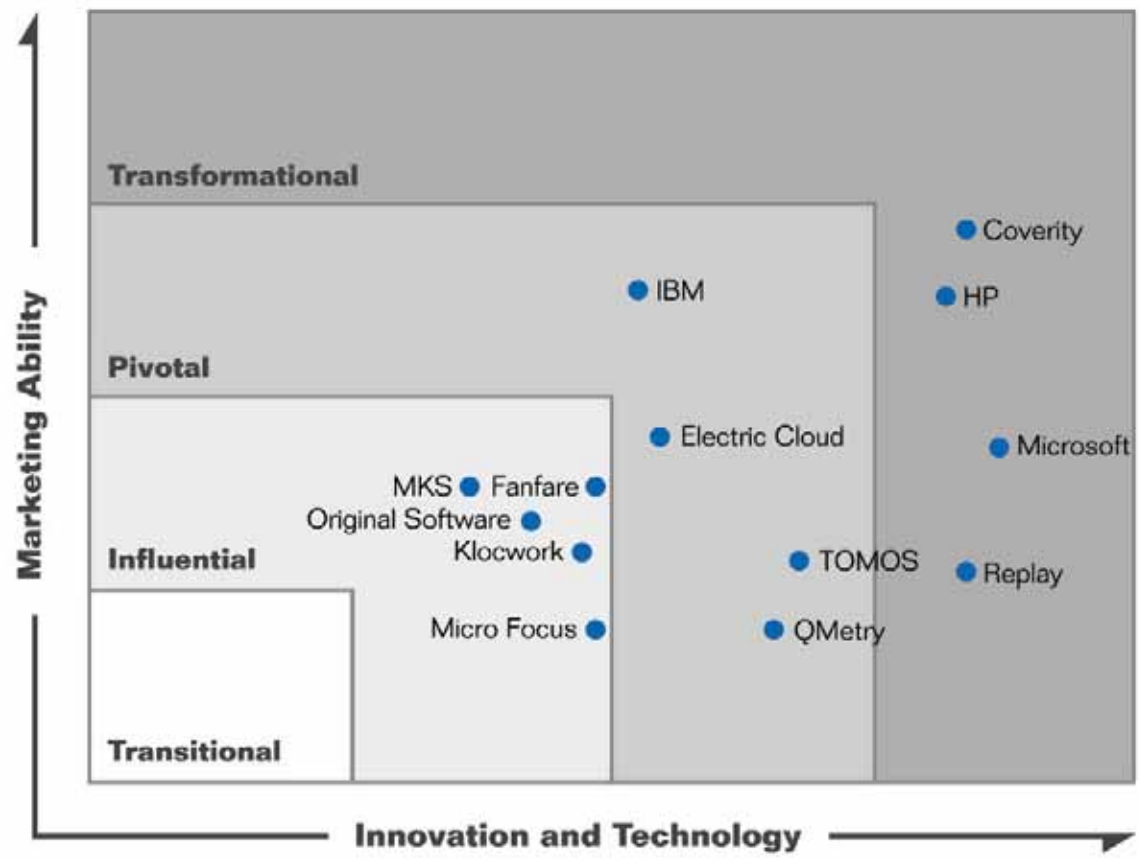
Influential – vendors of importance that are making an impact on the market and are viewed as compelling alternatives, or that fill specialized needs in the market.

Transitional — vendors that are at a crossroads in the market, either just entering the category or established vendors in the process of reinventing their products or positioning. Regardless of their status of either entering the market or reinvention, these vendors have less of a leadership impact on the market.

⊙ MARKET MOVER ARRAY: TESTING PLATFORMS

The Market Mover Array: Testing Platforms chart below identifies vendors with testing solutions in one or more categories of the rapidly expanding testing market.

Figure 1. voke Market Mover Array: Testing Platforms



MARKET MOVER ARRAY VENDORS

Think about the needs of your organization. What type of application, service, or hardware do you need to test? How do you want to test it? Where in the lifecycle do you want to test? What are your organizational constraints? Where is the software being used? How is the software being used? How is software controlling the hardware? How critical is the software? What are the implications and cost of a software failure? How do you know where to start with quality? What software problems are you experiencing?

August 11, 2010

All of these questions and answers are unique to each and every organization. The testing market is flourishing with areas of innovation that are necessary to the new types of applications, software, products, and services that must be tested today.

Use this report to understand the solutions available in the market and determine if you need to expand your existing toolset to achieve your quality objectives.

The Market Mover Array: Testing Platforms includes analysis of the following vendors:

- HP and IBM — market share leaders with comprehensive solutions and professional services
- Microsoft and Replay Solutions — delivering innovative game-changing technology
- Coverity, Electric Cloud, and Klocwork — extending essential innovation beyond the test team
- Fanfare — offering a testing platform for embedded systems and devices
- MKS — delivering a collaborative test management environment for product engineering teams and IT organizations
- QMetry and TOMOS — delivering lightweight cloud alternatives for testing
- Micro Focus and Original Software — challengers to the traditional testing vendors

An in-depth analysis of each of the 2010 Market Mover Vendors follows this Overview.

To learn more, visit www.vokeinc.com to watch voke's *What Tool Should I Buy* webcast series. This provocative series of on-demand webcasts features conversations with voke analysts and technology vendors discussing the latest tools and how to justify the purchase of new solutions.

⊙ MICROSOFT

Market Mover Rating: Transformational

Consider this vendor if your organization is:

- Developing with Microsoft solutions
- In need of a virtual lab solution
- Considering modern, high-value alternatives to existing traditional testing platforms
- Struggling with test automation and reuse
- Experiencing contention between testing and development teams

Overview

Microsoft's Visual Studio 2010 is an organically created lifecycle solution designed to eliminate the tedious tasks associated with testing and improve communication between testers and developers. Microsoft bolstered its quality offering by integrating transformational new technology available in Visual Studio Test Professional 2010 and Visual Studio 2010 Ultimate. Test Professional is a highly innovative and groundbreaking testing solution that is revolutionizing manual testing. The Microsoft testing solution with integrated virtual lab management is poised to be as significant to the testing market as the introductions of RUP and BTO.

This game-changing release shows that Microsoft intends to make testing a strategic component of the Visual Studio brand. Microsoft has seamlessly integrated technology that will likely force innovation in the market. Microsoft is using new technology to solve age-old problems.

Product Portfolio and Programs

The Microsoft solution delivers a modern testing platform for code analysis, automated testing, performance testing and manual test execution, enabling alignment between development, testing, and operations.

Microsoft testing products are included in the Visual Studio product line. Unfortunately, this developer-centric packaging obscures this exciting technology from capturing widespread awareness in testing organizations, especially for test automation and load testing.

- Visual Studio 2010 Professional — for complete developer testing; includes unit-testing capabilities within the IDE that can generate all the method stubs necessary for compiling unit tests, which help to ensure each unit of code is performing correctly.
- Visual Studio 2010 Premium — for advanced developer testing; in addition to the

Professional features, Premium includes: coded UI tests, database unit test, database change management/data generation/deployment, test impact analysis, code metrics, static code analysis, code coverage, and performance profiling.

- Visual Studio 2010 Ultimate — for comprehensive testing; in addition to Premium, Ultimate includes: load and web performance testing, and all components of Test Professional 2010.
- Visual Studio Test Professional 2010 — for testing organizations to simplify test planning, manual test execution and integrated defects and lab management.

With its comprehensive testing solution, Microsoft delivers a unified software testing platform for both developers and testers. This robust testing capability puts Microsoft in a unique position of aligning developers and testers with a common platform and language, enabling efficient communication about defect identification and resolution.

With the acquisition of Teamprise, Microsoft has incorporated Team Explorer Everywhere to deliver a bridge to the Java platform and make Visual Studio a heterogeneous solution. The inclusion of support for the Java platform makes Microsoft Visual Studio 2010 an enterprise-ready solution.

Microsoft has introduced new, innovative, and game-changing testing technology that eliminates the time-consuming and tactical challenges faced by testing teams and leverages virtualization technology to improve communication and collaboration between developers and testers.

- Integrated virtual lab management — delivers an environment as close to production as possible for testing and eliminates contention over defect replication. Developers directly connect to environment snapshots associated with each defect. Virtual lab technology further aligns development, testing, and operations by enabling a full build-deploy-test workflow.
- IntelliTrace — records execution of specified events to allow developers to look back at a past state of the application with debugging information and pinpoint elusive defects.
- Actionable defect reports — automatically generate rich diagnostic information for each defect. Defects are how developers and testers communicate. With Microsoft Test Professional 2010 for test execution, rich defects are rapidly filed and the activity of testing is simplified.
- "Fast Forward for Manual Testing" — captures an action recording of the actual steps a tester took while interacting with the application during testing. Testers can play back action recordings against a new build to regress each test case. This eliminates the tedious manual re-execution of the steps required to set up a scenario, enabling rapid regression testing. Automation engineers can easily leverage the recording to add or

extend each scenario for build verification tests or a suite of automated test scenarios.

- Real-time interactive test case generation — with the option to record an exploratory test scenario, the tester's action steps are automatically captured for easy reuse in a defect report or to dynamically create a new test case. This feature enables rapid test case creation based on real world use of the product, eliminating the tedious manual creation and documentation of test cases.
- Automated UI Tests — known as Coded UI Tests, allow developers or test engineers to leverage existing action recordings created during a manual test execution to generate code to replicate those exact same steps. Developers can also create action recordings within the Visual Studio IDE. In either case, assertions can be added to these coded tests to effectively allow developers or automation engineers to easily leverage these recordings to add or extend each scenario for build verification tests or a suite of automated UI test scenarios.
- Test Impact Analysis — test impact data shows a list of recommended tests based on code changes and identifies which defects are addressed in a given build. This enables managers to assess the state of a new build and provides testers the information they need to optimize regression testing.
- Integrated Test Case Management — powered by Team Foundation Server is a common platform serving both developers and testers and other lifecycle stakeholders to enable different roles in the organization to plan, develop and maintain tests with the ability to link and trace test cases to defects, code, or requirements.

Microsoft's lifecycle platform is unique in that it fosters a common language for QA and development. Both developers and test engineers use the same language whether developing code or automating test cases. Microsoft recognizes that developing test automation requires the same skills as developing any other software. This is a distinct advantage over testing tools that use a proprietary scripting language for automation and enables developers and automation engineers to work together on automation and increases the skills and career path options of automation engineers.

From an organizational perspective, Test Professional 2010 brings developers and testers closer together by eliminating previous points of contention over defect replication, detection, reporting, and repair. Eliminating the developer/tester contention allows these teams to focus on more strategic efforts in delivering the best software possible to the customer.

With integrated virtual lab technology, actionable defect reports, rapid regression testing, and real-time interactive test case generation, Microsoft has eclipsed the competition in providing modern useful tools for delivering high value from testing.

Customer Profile/Experience

Microsoft traditionally tends to focus on homogeneous Microsoft shops and the small and medium-sized business (SMB) market. Its testing products and Team Explorer Everywhere give Microsoft the opportunity to become one of the market share leaders for testing platforms at the enterprise level.

To gain significant market share, Microsoft must put as much emphasis and energy on marketing this technology as it did in creating and developing it. Microsoft understands the developer community better than any vendor in the industry. Microsoft must now focus its efforts on truly understanding, connecting with, and delivering solutions for the testing community.

Microsoft will certainly attract its current installed base to its Test Professional 2010 offering. The new and innovative testing features and integrated virtual lab management will attract competitors' customers. Customers of a competitive solution who have struggled with test automation will find the Microsoft solution easy and intuitive. The Microsoft combination of virtual labs, actionable defect reports, rapid regression testing and real time interactive test case generation catapults the testing activity to new levels of productivity. Microsoft's challenge is to keep those new customers satisfied by continuing to enhance and innovate its testing offering and improving its positioning and packaging beyond the developer to a broader target audience across the entire application lifecycle.

Net/Net

Microsoft's testing technology is some of the most innovative to be released to the testing market in the past several years. Microsoft is capable of moving the testing market to the next level, taking a significant market share position, and shifting the conversation from technology to business differentiation. However, Microsoft must realize a market shift is not solely based on technology and innovation. The quintessential software vendor must put as much emphasis on marketing this breakthrough product as it has on designing and developing it.

We expect Microsoft to continue innovating its technology to transform the testing market. Microsoft has every opportunity to become the new testing market leader by shifting its marketing to a new and much needed business orientation. Microsoft Visual Studio 2010 is setting the standard for the future of testing.

August 11, 2010

Contact Information

Corporate Headquarters

voke, inc.
2248 Meridian Boulevard
Suite H
Minden, NV 89423
USA

Phone: +1-866-895-9045
Web: www.vokeinc.com
Blog: www.voke.blogspot.com

⊙ ABOUT VOKE

voke, founded in 2006, is a modern analyst firm focused on the edge of innovation. voke's primary coverage area is the application lifecycle and its global transformation, including virtualization, cloud computing, embedded systems and device software.

voke provides in-depth coverage of core vendors and innovators in the application lifecycle market. Companies of all sizes, financial firms, and venture capital organizations turn to voke to harness strategic marketing, independent and impartial market observations and analysis to move markets beyond the status quo. Please visit www.vokeinc.com to subscribe to voke research.

vOke®
Moving markets beyond the status quo!

November 16, 2009

Microsoft Ups The ALM Ante With Its Bet On Teamprise

Expect VS 2010 To Accelerate The Shakeout In The Application Life-Cycle Management Market

by Jeffrey S. Hammond
with Mike Gilpin and Adam Knoll

EXECUTIVE SUMMARY

Microsoft's Team Foundation Server (TFS) has proven very popular with .NET developers but not so much with Eclipse developers. This presents a problem for Microsoft, because many of its largest customers develop for both .NET and Java and want a consolidated application life-cycle management (ALM) solution that will support development teams regardless of what platform they use. Over the past few years, Microsoft has pointed to a partner's product — SourceGear's Teamprise Client Suite — as its recommended solution to the heterogeneity problem. This has proven unsatisfactory to many customers, so after a long internal debate, Microsoft has acquired the Teamprise code base. Microsoft will release an updated version of Teamprise as part of its Visual Studio 2010 release train and will reduce the new solution's overall per-developer cost. The resulting product combination will prove much more attractive to large enterprises, which will now have the option of a lower-cost ALM solution for all the platforms they use, supported by the full force of Microsoft.

MICROSOFT'S ALM STRATEGY FOR VS 2010 EMBRACES OUT-OF-THE-BOX HETEROGENEITY

On Monday, November 9, Microsoft announced its asset acquisition of Teamprise from its original developer, SourceGear.[1] With this move, Microsoft becomes the sole owner of the copyrights and source code for all future versions of the Teamprise Client Suite after version 3.3 (the current version). This acquisition also covers the Teamprise Plug-in for Eclipse, the Teamprise Explorer, and the Teamprise Command-Line Client. Collectively these products provide native cross-platform clients and Eclipse IDE integration into the major ALM subsystems of Microsoft's ALM solution, Team Foundation Server (TFS).

Microsoft Must Add Heterogeneity To Build On Its Initial ALM Success

So what's behind this acquisition, and why now? After all, the Teamprise Client Suite is not exactly a new product, and for the past few years, Microsoft has professed comfort with supporting non-.NET developers through its partner channel and its relationship with SourceGear. The acquisition was driven by certain realities of the ALM market:

- **Microsoft's large customers demand ALM support for more than just .NET.** While it's easy for individual practitioners to self-identify as ".NET developers" or "Java jocks," the development culture of most enterprises is more complex. In fact, 51% of global 2000 organizations — those with 20,000 employees or more — report that they use both .NET and Java application platforms (see Figure 1).

Forrester also hears application development executives saying that they want to consolidate the number of moving parts that compose their ALM strategies (see Figure 2). Most of these leaders just don't see the logic in maintaining separate ALM environments for different application platforms, especially if it drives up acquisition, administration, and maintenance costs.

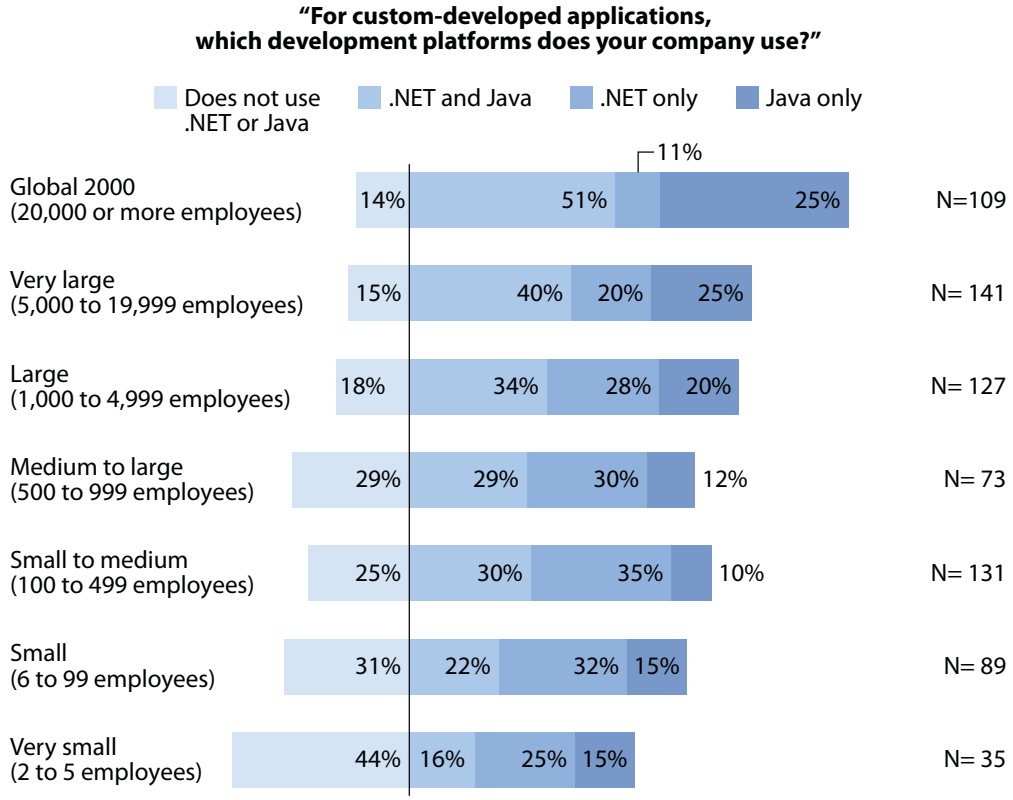
- **Development shops also need ALM clients that run on alternative operating systems.** Despite Windows desktops' dominance, not all developers use Windows machines to write code. In our latest developer survey, we found that about 30% of developers write code on computers running operating systems other than Windows, including 16% who use Linux (see Figure 3). Acquiring the Teamprise Client Suite allows Microsoft to address this segment of the ALM market, as it runs on Linux, Mac OS X, Solaris, AIX, and HP-UX. While it's a stretch to say that Microsoft would target development shops that are exclusively non-Windows, this mixed-mode ALM client support makes it easier for Microsoft to position TFS as a single ALM solution even in cases where customers are committed to sustaining some areas of non-Windows development.

- **Mission-critical ALM processes need the support of a trusted partner.** When ALM processes go awry, everyone feels the pain. Therefore, it's easy to understand why application development decision-makers prefer to seek solutions from large, well-established ALM vendors. Extended support hours, a single throat to choke, and the capability to put boots on the ground to assist in a companywide rollout matter at lot when you're a development executive spending seven figures on an ALM solution. While Teamprise Client Suite is a technically capable product, we've repeatedly heard from Forrester clients that a partner-based ALM add-on from a small, private company was a nonstarter for them — it was simply too risky a bet to make, regardless of the product's quality.

- **Today, Eclipse users rarely consider Microsoft their primary ALM provider.** Microsoft has been successful in building market traction and brand awareness for TFS in the past five years. Overall, application development professionals view Microsoft as a leading ALM provider (see Figure 4). However, only 4% and 9%, respectively, of developers from the Americas and Europe who use Eclipse as their primary integrated development environment (IDE) identify Microsoft as their primary ALM provider. And Microsoft can't afford to ignore Eclipse developers, because there are a lot of them. Multiple surveys over the past two years show strong adoption of Eclipse as a primary IDE, especially for Java development (see Figure 5).[2]

With the Teamprise acquisition, Microsoft gets a cross-IDE, cross-platform code base that allows the TFS team to field an integrated solution that provides customers with unified support and a Microsoft commitment to ALM heterogeneity — at least as soon as it can release its updated version of the Teamprise Client Suite.

Figure 1 Most Large Enterprises Use Both Java And .NET

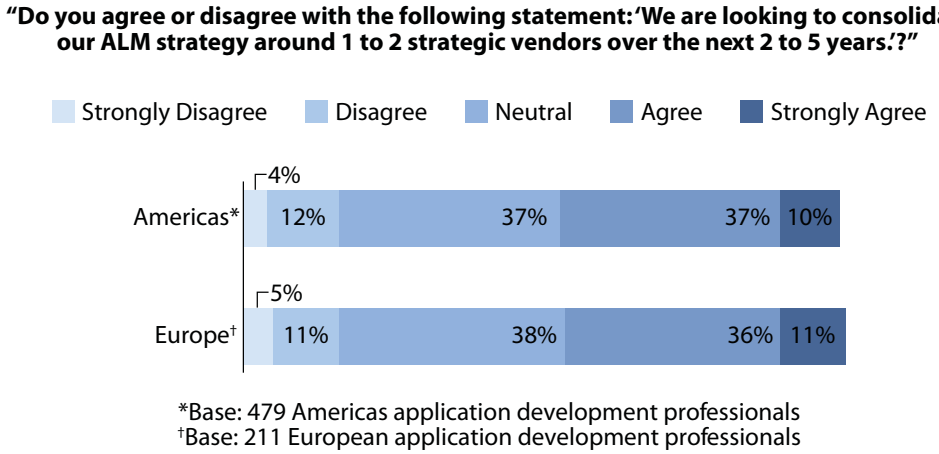


Base: 917 platform software decision-makers at North American and European enterprises and SMBs (percentages may not total 100 because of rounding)

Source: Enterprise And SMB Software Survey, North America And Europe, Q4 2008

55748 Source: Forrester Research, Inc.

Figure 2 Development Shops Want To Consolidate Their ALM Tool Sets

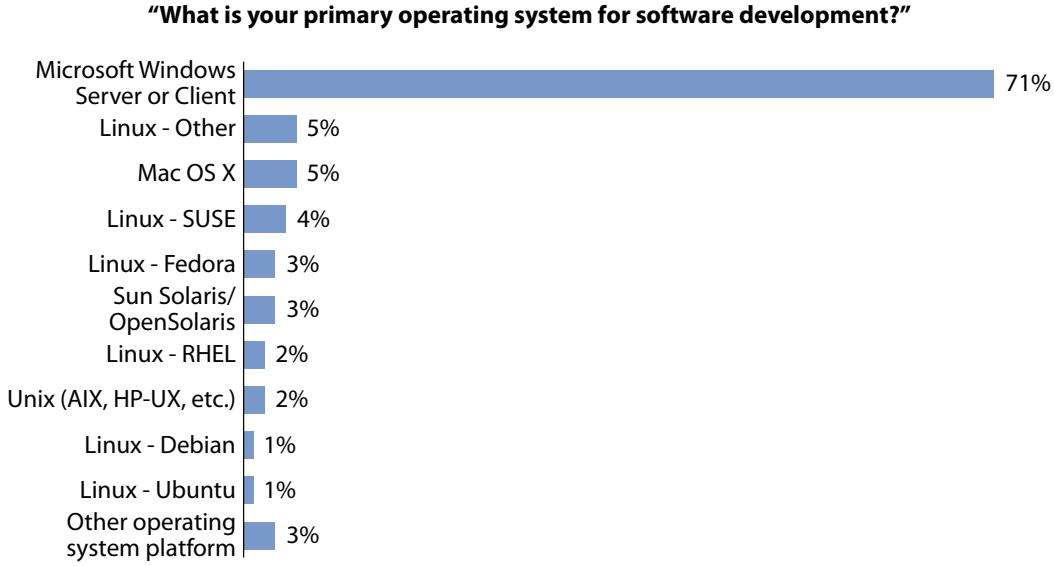


*Base: 479 Americas application development professionals
†Base: 211 European application development professionals

*Source: Q1 2009 Americas Application Life-Cycle Management Usages And Trends Online Survey
†Source: Q4 2008 European Application Life-Cycle Management Usage And Trends Online Survey

55748 Source: Forrester Research, Inc.

Figure 3 Operating System Use By Developers

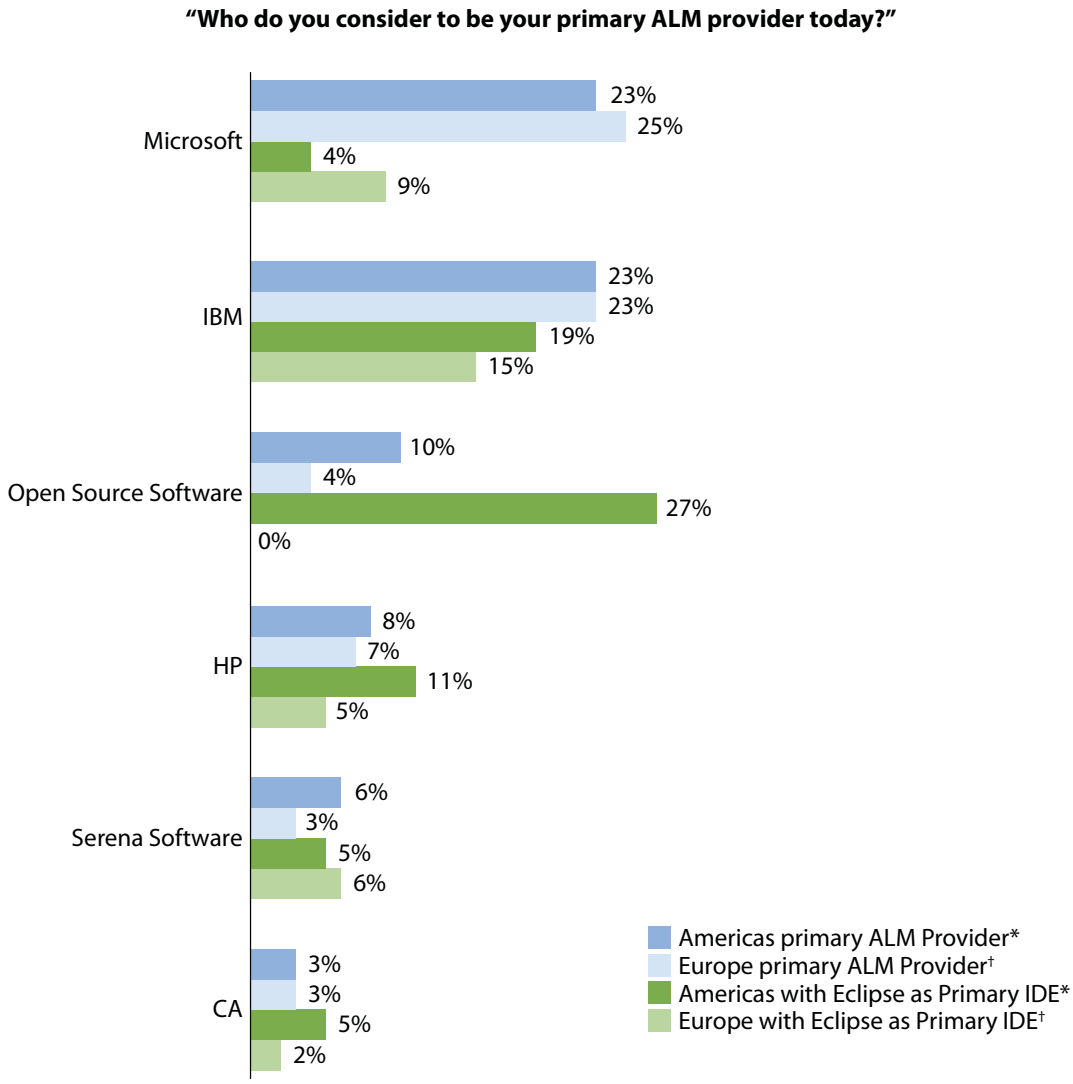


Base: 1,298 application development professionals

Source: Forrester/Dr. Dobb's Global Developer Technographics® Survey, Q3 2009

55748 Source: Forrester Research, Inc.

Figure 4 Organizations Most Often Cite Microsoft Or IBM As Their Primary ALM Vendor

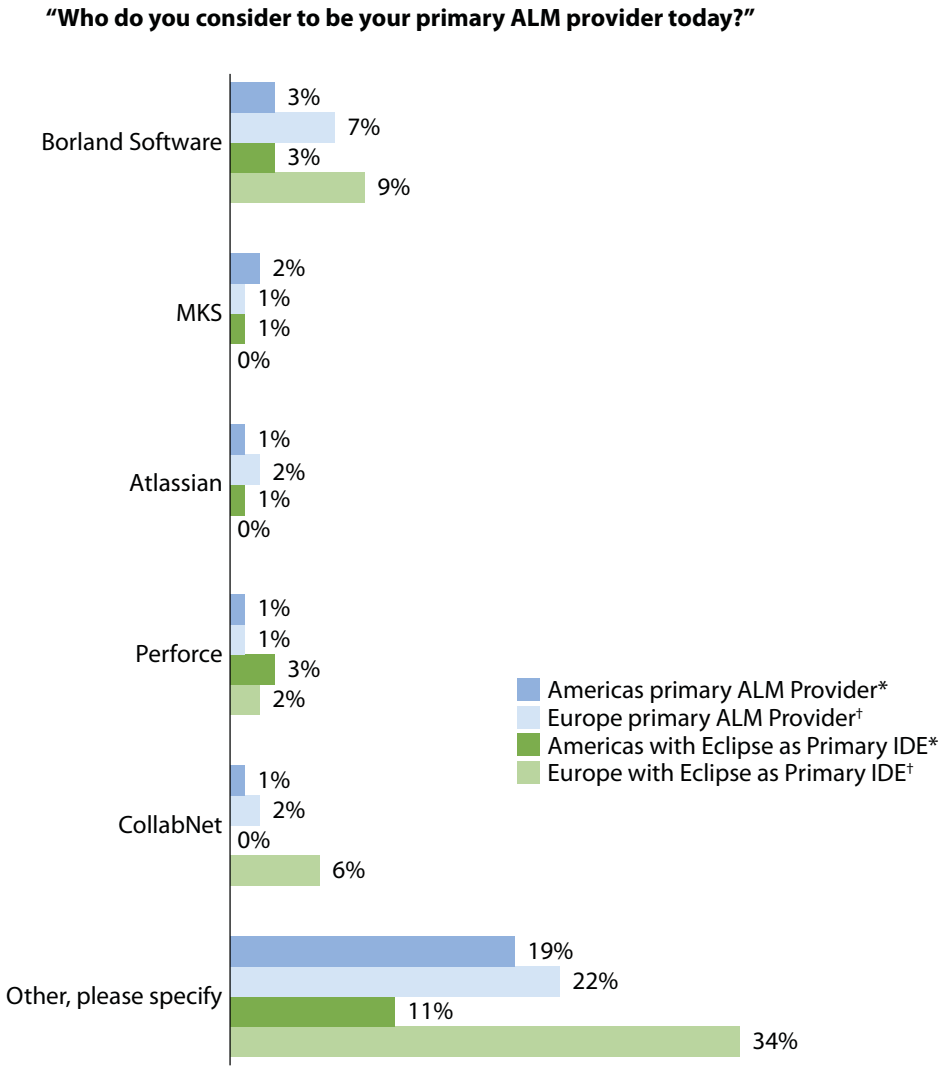


*Base: 395 Americas application development professionals
†Base: 211 European application development professionals

*Source: Forrester's Americas Application Life-Cycle Management Usages And Trends Online Survey, Q1 2009
†Source: Forrester's European Application Life-Cycle Management Usage And Trends Online Survey, Q4 2008

55748 Source: Forrester Research, Inc.

Figure 4 Organizations Most Often Cite Microsoft Or IBM As Their Primary ALM Vendor (Cont.)

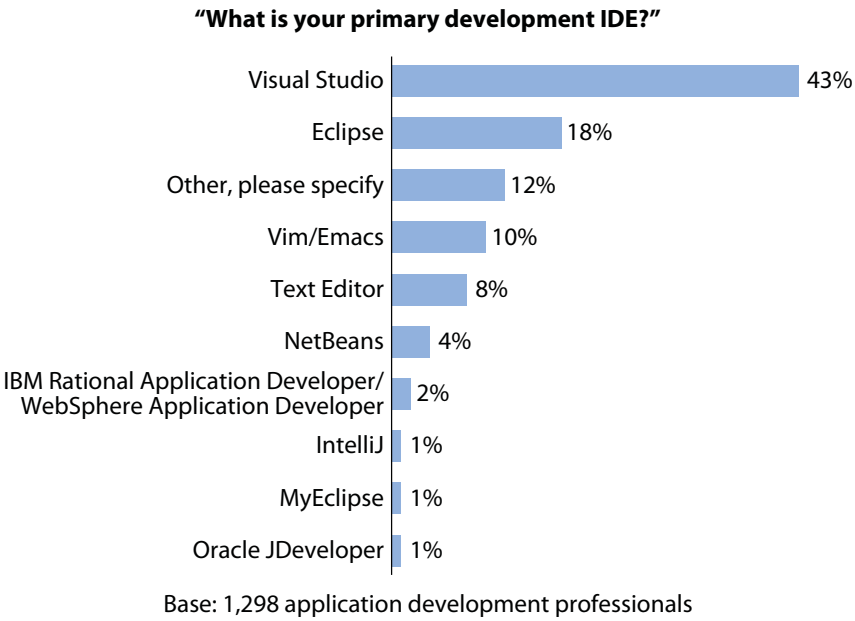


*Base: 395 Americas application development professionals
†Base: 211 European application development professionals

*Source: Forrester's Americas Application Life-Cycle Management Usages And Trends Online Survey, Q1 2009
†Source: Forrester's European Application Life-Cycle Management Usage And Trends Online Survey, Q4 2008

55748 Source: Forrester Research, Inc.

Figure 5 Eclipse Is A Popular IDE



Source: Forrester/Dr. Dobb's Global Developer Technographics® Survey, Q3 2009

55748 Source: Forrester Research, Inc.

Don't Expect An Integrated Solution Until The Launch Of Visual Studio 2010

Although Microsoft officially acquired the Teamprise Client Suite code base on November 4, application development professionals can't go out and buy the product from Microsoft today. Microsoft's acquisition wasn't a complete takeover of SourceGear, so there will be a more gradual transition than you might expect from your experience with other acquisitions of nonpublic companies. The transition of Teamprise will play out in several steps:

1. **Microsoft gets copyright, code, and staff.** Microsoft isn't just taking possession of a hunk of source code; more importantly, several developers that maintain the current code base will become Microsoft employees. These developers will relocate to Microsoft's offices in North Carolina, where core components of TFS are developed. The net benefit to customers will be a smoother transition of ownership, as existing expertise with the current code base will aid knowledge transfer and help spin up additional development resources if needed to complete subsequent steps in the product integration process.

2. **Microsoft will "greenwash" the Teamprise Client Suite 4.0 code base.** While the existing versions of Teamprise are designed to technically integrate with TFS, there will still be work to do to turn the code base into a full-fledged Microsoft product. This "greenwashing" will

include technical tasks such as integrating the Teamprise release cadence into the TFS team's development and governance processes as well as business-focused tasks such as updating product catalogs and the Microsoft Developer Network (MSDN). While none of these tasks should be particularly difficult or risky, there's still a certain amount of effort that will take the combined team time to work through before it will be able to prepare a new release that's up to official Microsoft standards.

3. **Microsoft will release an updated, improved product within six months.** One immediate benefit to customers is that Microsoft can eliminate the current forced separation between Teamprise and the TFS Client Access License (CAL) that current users must also purchase to access the TFS server. When Microsoft releases the new version, it will be packaged as a single product that combines updated Teamprise code and a single TFS CAL at an anticipated list price of $799 — less than the current price of each separate component. Microsoft has also indicated that the Teamprise code will be included in Visual Studio (VS) 2010 with MSDN Ultimate. This means that developers who work in both Eclipse and Visual Studio will always be able to access TFS from both for no additional cost. While the TFS team has not set an official release date for the new version, its goal is to ship as close as possible to the general availability of VS 2010.

4. **SourceGear will terminate support when Microsoft's new product ships.** While Microsoft is prepping an updated version of the Teamprise Client Suite for release next year, SourceGear will continue to sell and support the existing 3.3 release. On the day that the Microsoft release ships, SourceGear will terminate support for all versions of Teamprise. At that point, existing Teamprise customers will be offered the new Microsoft version at no cost, provided they own both a Teamprise v3.x license and the corresponding TFS CAL. Shops that are already using Teamprise and that need to purchase new licenses in the interim should not hesitate to do so, but they should attempt to negotiate a price that reflects the next version's reduced price.

If Heterogeneity Is The Price Of Increased Profits, Expect Microsoft To Pay It

Microsoft's original ALM strategy was clear: serve the needs of Visual Studio developers and blunt the impact of IBM's acquisition of Rational Software on the .NET developer base.[3] But over the past three years, as TFS has worked to establish a leading presence in the ALM market, the potential for profit from the lucrative ALM market touched off a heated debate inside the corridors of Redmond. That debate now appears over, and the advocates of larger profits from a bigger share of the ALM market won. The price: A Microsoft-branded, heterogeneous ALM solution, with the Teamprise Client as the expedient route. What makes this outcome even more interesting is that it's not an isolated incident; Microsoft's embrace of PHP with the Web Platform Installer, the donation of roughly 20,000 lines of source code under a GNU Public License (GPL) for inclusion in the Linux kernel, and the organization of CodePlex.org are all events that share a common theme. If becoming more open and heterogeneous results in more business and better profits, then Microsoft will do so; to understand and predict Microsoft's future behavior, you just need to "follow the money."

WHAT IT MEANS

MICROSOFT'S TEAMPRISE ACQUISITION WILL ACCELERATE ALM COMMODITIZATION

Microsoft has come a long way since its initial release of Team Foundation Server in 2006. With the Teamprise acquisition, it's now clear that Microsoft sees TFS as a platform that is ready to stand on its own and drive significant deals at large enterprises that demand a heterogeneous ALM solution. Expect Microsoft to use a combination of new features in VS 2010, heterogeneous capabilities from Teamprise, and aggressive pricing to put significant pressure on other ALM players, especially those that offer integrated software change and configuration management (SCCM). The result of these tactics is that in 2010, the market reference price for integrated, heterogeneous SCCM will pass below the $1000 barrier on its way to full commoditization. Application development professionals should use these new economic realities to pressure their current ALM suppliers for better deals or prepare to justify the benefits of higher-priced ALM solutions to senior management.

SUPPLEMENTAL MATERIAL

Methodology

Forrester's Enterprise And SMB Software Survey, North America And Europe, Q4 2008, was fielded to 2,227 IT executives and technology decision-makers located in Canada, France, Germany, the UK, and the US from companies with two or more employees. This survey is part of Forrester's suite of Business Data Services studies. Forrester fielded the survey from December 2008 to February 2009. e-Rewards fielded this survey online on behalf of Forrester. e-Rewards provided incentives to survey respondents. We have provided exact sample sizes in this report on a question-by-question basis.

Forrester's Business Data Services fields eight business-to-business technology studies in 19 countries each calendar year. For quality control, we carefully screen respondents according to job title and function. Business Data Services ensures that the final survey population contains only those with significant involvement in the planning, funding, and purchasing of IT products and services. Additionally, quotas are set for company size (number of employees) and industry as a means of controlling the data distribution and establishing alignment with IT spend calculated by Forrester analysts.

In addition to sampling error, one should bear in mind that the practical difficulties in conducting surveys can introduce error or bias into the findings of opinion polls. Other possible sources of error in polls are probably more serious than theoretical calculations of sampling error. These other potential sources of error include question wording, question ordering, and nonresponse. As with all survey research, it is impossible to quantify the errors that may result from these factors without an experimental control group, so we strongly caution against using the words "margin of error" in reporting any survey data.

These statements conform to the principles of disclosure of the National Council on Public Polls.

We have illustrated only a portion of survey results in this document. For access to the full data results, please contact bds@forrester.com.

Forrester's Q1 2009 Americas Application Life-Cycle Management Usages And Trends Online Survey was fielded to 472 application development professionals. Forrester fielded the survey in the first three weeks of March 2009. Forrester's Q4 2008 European Application Life-Cycle Management Usage And Trends Online Survey was fielded to 299 application development professionals. Forrester fielded the survey in the first three weeks of November 2008. For both surveys, respondents were asked to comment on multiple ALM topics to guide ongoing ALM research. For quality assurance, panelists are required to provide contact information and answer basic questions about their firms' size and development activities.

The Forrester/Dr. Dobb's Global Developer Technographics® Survey, Q3 2009, was fielded to 1298 application development and program management professionals who are readers of Dr. Dobb's magazine. For quality assurance, respondents are required to provide contact information and answer basic questions about themselves. Forrester fielded the survey from July 2009 to August 2009. Respondent incentives included a summary of the survey results and a chance to win one of five $50 gift certificates.

Exact sample sizes for the surveys used in this report are provided on a question-by-question basis. Surveys are not guaranteed to be representative of the entire application development population. Unless otherwise noted, statistical data is intended to be used for descriptive and not inferential purposes.

If you're interested in joining one of Forrester's Research Panels, you may visit us at http://Forrester.com/Panel.

ENDNOTES

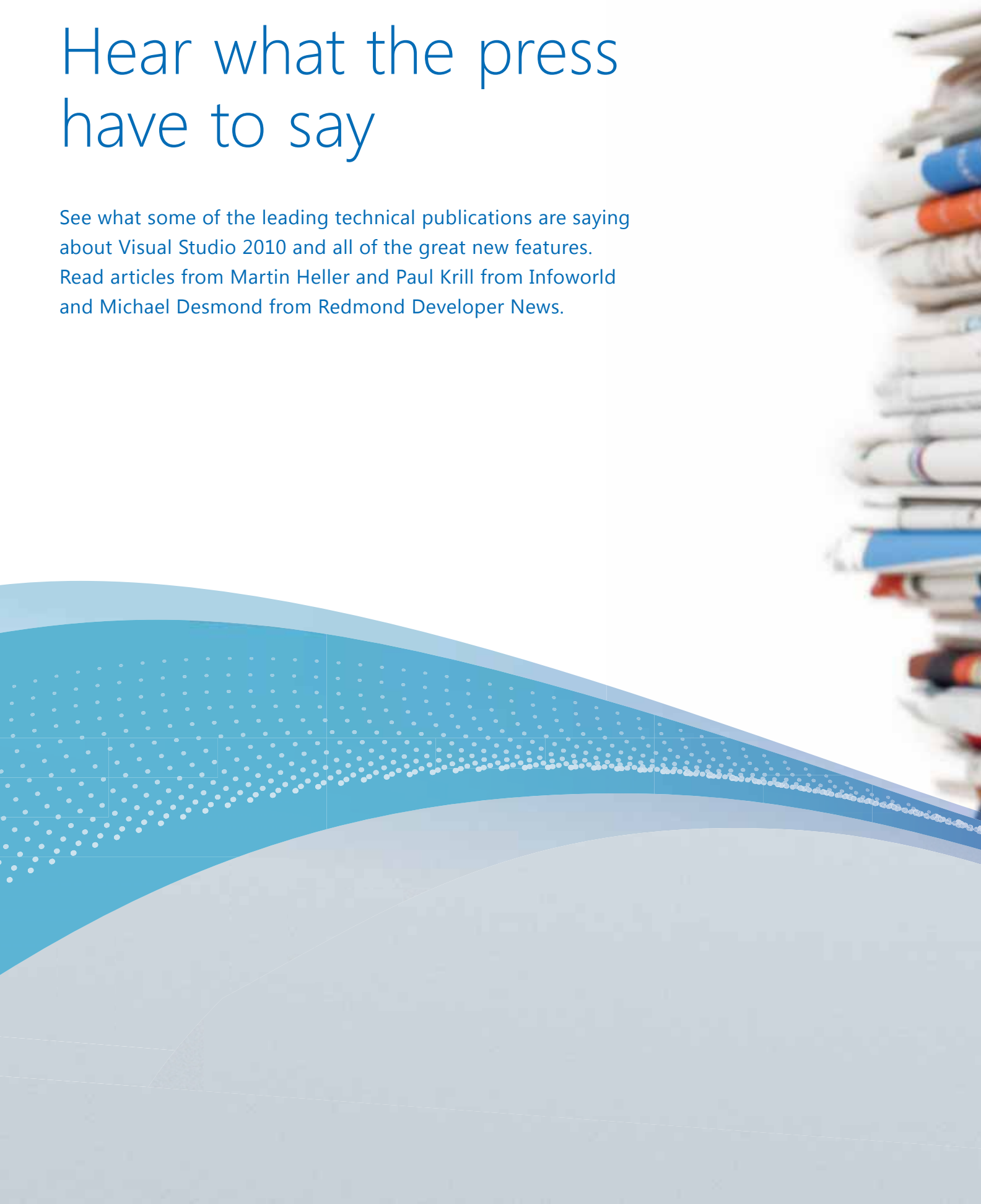
[1] Source: "Microsoft Acquires Teamprise Assets, Provides Cross-Platform Support for Visual Studio," Microsoft press release, November 9, 2009 (http://www.microsoft.com/presspass/press/2009/nov09/11-09teamprisepr.mspx).

[2] Over the past two years, Forrester has polled for IDE usage in multiple surveys, including the Forrester/1105 Media June 2007 North American IDE Usage Online Survey, the Enterprise And SMB Software Survey, North America And Europe, Q4 2008, the Q4 2008 European Application Life-Cycle Management Usage And Trends Online Survey, and the Q1 2009 Americas Application Life-Cycle Management Usages And Trends Online Survey.

[3] Microsoft had a strong partnering relationship with Rational Software Corporation prior to IBM's acquisition of Rational in 2003. During the partnership period, Rational ClearCase was a popular choice of Visual Studio shops that needed a full-featured SCCM tool (especially Visual C/C++ shops). Even in Forrester's Q1 2009 Americas Application Life-Cycle Management Usages And Trends Online Survey and Forrester's Q4 2008 European Application Life-Cycle Management Usage And Trends Online Survey, 27% and 26% of IBM Rational ClearCase users, respectively, reported that they consider Microsoft to be their primary IDE provider.

Hear what the press have to say

See what some of the leading technical publications are saying about Visual Studio 2010 and all of the great new features. Read articles from Martin Heller and Paul Krill from Infoworld and Michael Desmond from Redmond Developer News.



InfoWorld review: Visual Studio 2010 delivers

A no-brainer upgrade for Microsoft-oriented developers, Microsoft Visual Studio 2010 marks a major advance in functionality and ease

As a daily user of Visual Studio from its inception, and of Visual C++ and Visual InterDev before that, I have been following the evolution of Microsoft's development environment quite closely. In the Visual Studio 2010 IDE, Microsoft has taken several large steps away from its legacy code. That was a gutsy and potentially risky move on the part of the Visual Studio team, but one that worked out well and will lay the foundation for future product growth.

By Martin Heller | InfoWorld | APRIL 13, 2010

Visual Studio 2010 is a major upgrade in functionality and capability from its predecessor. It includes some major rewriting of core features, as well as many new features. Developers, architects, and testers will all find areas where the new version makes their jobs easier. Despite the higher pricing for this version, most serious Microsoft-oriented shops will upgrade to Visual Studio 2010 and never look back.

[Also on InfoWorld: See Martin Heller's Strategic Developer blog for a scrolling tour of Visual Studio 2010 highlights and ongoing coverage of Microsoft development technologies.]

Raising the bar for IDEs

The most obvious large step is revamping the core editing and designer views to use the Windows Presentation Foundation (WPF). I covered this and the related improvements in the UI and debugger in my review of Visual Studio 2010 Beta 1. Initially there were some performance penalties associated with this, but now almost everything I do in the Visual Studio 2010 IDE happens faster than it would in Visual Studio 2008, even columnar text selection. I particularly like the navigation improvements in the code editor. Both the "Navigate to" and "Call Hierarchy" features have proven invaluable to me recently as I learned a large C++ code base.

Another obvious large step is to revamp IntelliSense and start to support Test-Driven Development (TDD). As I discussed in my reviews of Visual Studio 2010 Beta 1 and Visual Studio 2010 Beta 2, IntelliSense has been redesigned, the easily corrupted IntelliSense .NCB file has been eliminated, and the whole system has become more sensible about offering to do what you might actually want instead of cavalierly completing your typing with irrelevancies. At this point, Visual Studio 2010 is usable for test-driven development (TDD), although I wouldn't yet call it a real TDD-oriented system. For me, that would require another view of a project that emphasized a Red, Green, Refactor development cycle.

A third obvious large step is targeting and supporting multiple versions of the .Net Framework (2.0 through 4.0) in an intelligent way. A fourth is the enhanced debugging capabilities, both in debugging threaded applications and in historical debugging for managed assemblies. There are greatly enhanced sets of tools for architects and testers, such as

Test Center Scorecard InfoWorld

	Capability	Ease of Development	Documentation	Performance	Value	Overall Score
	30%	30%	15%	15%	10%	
Microsoft Visual Studio 2010	10	9	9	9	9	9.3 EXCELLENT

Sequence Diagrams, Dependency Graphs (for managed assemblies), better bug reporting, and reproducibility. Then there is the new, non-brain-damaged help engine; support for Azure and Silverlight; support for Windows 7, SharePoint 2010, and Office 2010; functional programming with F#; and so on. Silverlight is targeted by its associated .Net version, but Visual Studio 2010 actually supports Silverlight versions 1 through 4.

I mentioned some of the improvements to Team Foundation Server (TFS) when I discussed Beta 2. One thing I didn't mention that deserves some attention is the new ability to gate check-ins for selected developers. Gated check-ins were one of the key features Linus Torvalds wanted for Git that he didn't have in Subversion. If you think about an open source project like the Linux kernel, you really don't want inexperienced developers checking changes into the trunk unless all the unit tests have passed and a senior developer has reviewed the code. TFS can do that now (not that Linus would ever consider using it).

Since Beta 2, .Net Framework 4.0 has picked up touch support in WPF. Visual Studio 2010 proper sports a cleaner start page, adds SQL Azure support, and behaves better when running in virtualized environments. The Visual Studio 2010 installer has been improved to the point where my install of the released bits didn't require any reboots at all. The bugs and performance issues that I and others reported in the release candidate have all been fixed.

Visual Studio 2010 vs. Visual Studio 2008

If you look at all the enhancements to Visual Studio 2010 compared to Visual Studio 2008 SP1, it's clear that the product has grown quite a bit in terms of capabilities, with lesser gains in performance and ease of development. And yet if you compare today's scores with the ones I gave to Visual Studio 2008 SP1 in August 2008, you'll find they're identical.

Times have changed, as have the scales we use for scores. I gave Visual Studio 2008 SP1 a 10 for capability; in its time, compared to its competition, it deserved it. Visual Studio 2010 gets the same 10 rating for capability now, but if I had to rerate Visual Studio 2008, it would now get about an 8.5.

Top 10 new and improved in Microsoft Visual Studio 2010

- Cleaner and more capable code editing and design views
- Improved performance
- Smart targeting of multiple .Net Framework versions
- Improved code navigation (without needing compilation)
- Better support for test-driven development

- Enhanced debugging for threads and historical debugging for managed code
- Support for Azure cloud, SharePoint 2010 sites, Silverlight RIAs, and Office 2010
- Improved capabilities for architects and testers
- Includes F# (functional programming) and supports dynamic languages
- Improved team management and code check-in controls

Similarly, ease of development in Visual Studio 2010 rates a 9 out of 10 -- because, after all, it doesn't yet write the code for you. Visual Studio 2008 SP1 got the same 9 in its time, but now it would get an 8. How could we put up with the old IntelliSense? After using the new IntelliSense, I hate to go back to Visual Studio 2008.

There's more documentation in Visual Studio 2010, but it's describing more product. The new help engine is much better than the old one, but that just makes the old one look bad in retrospect. I hate going back and watching the local search grind on the hard disk for several minutes at a time. Now that I've used the new help, the old help would only get a 7.5.

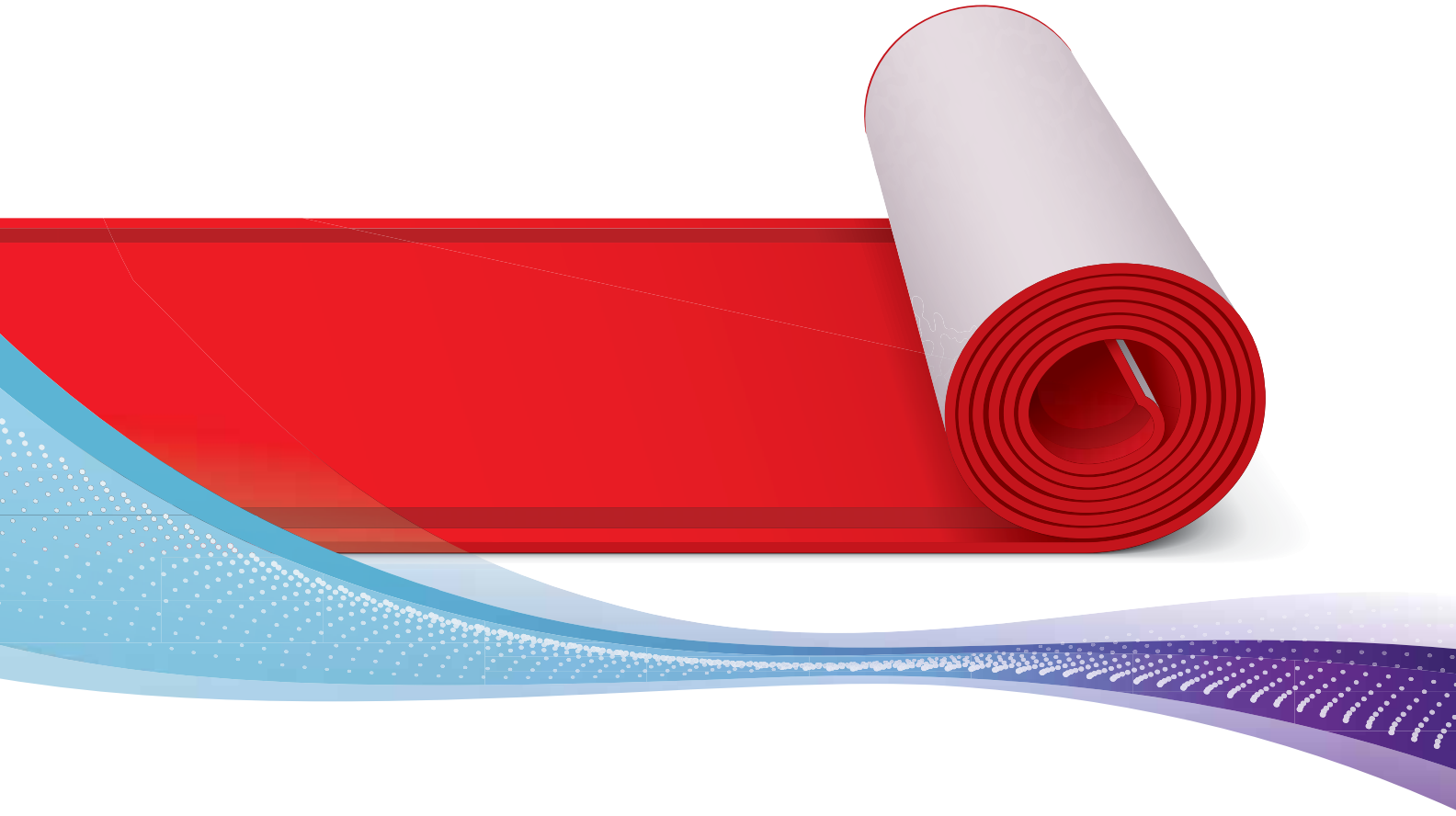
Visual Studio 2010 performance is only slightly better than Visual Studio 2008 performance; the old one was about a 9.1, and the new one is a 9.3. With our convention that component scores must be integers, that difference doesn't show up in the scorecard.

The change in value is a different issue. Visual Studio 2010 has greatly expanded capabilities and a commensurate increase in price. Some of you will gasp at the new pricing. If you or your management falls among this group, try thinking about what your time is worth to your employer -- and don't forget to add health care and overhead costs. If your budget won't cover the upgrade you want, perhaps you'll have to settle for the upgrade you need, which might mean dropping down to a less expensive edition. But a calculation of the ROI for good development tools like Visual Studio 2010 often comes back with a justification for much higher prices than you're emotionally willing to pay.

My company has already decided to upgrade to Visual Studio 2010. We'll keep shipping our product compiled with an older version of Visual Studio until we come out with our own next major product version or we need a compelling feature we can only get from the new frameworks and libraries. But we'll still have the benefits of using Visual Studio 2010 for development. Most likely, your company will want to do the same.

Microsoft to roll out Visual Studio upgrade

Silverlight, .Net Framework also receive updates this week



Without further adieu, Microsoft will release on Monday its Visual Studio 2010 software development system and the accompanying .Net Framework 4 platform, followed later in the week by the release of the Silverlight 4 rich Internet plug-in software.

By Paul Krill | InfoWorld | APRIL 12, 2010

The three products are upgrades to key Microsoft technologies and have been touted by the company for some time now. The Visual Studio 2010 platform is anchored by the Visual Studio IDE and also features application lifecycle management tools.

[InfoWorld columnist Martin Heller recently wrote about making the switch to Visual Studio 2010.]

"We've got a huge host of great productivity features, things like multiple monitor support," enabling developers to enter code on one monitor and view the design on another, said Sean McBreen, senior director of developer marketing at Microsoft.

Visual Studio 2010 is accompanied by .Net Framework 4. Among other highlights of Visual Studio 2010 is a new editor based on WPF (Windows Presentation Foundation), providing what Microsoft described as a more flexible, feature-rich environment.

"[Visual Studio 2010 is] completely redesigned, it's now built on top of WPF," said Gill Cleeren, a .Net architect at consulting and software development firm Ordina, which is a Microsoft business partner.

"It's a complete new interface that you're getting," said Cleeren, a beta user of Visual Studio 2010 and a leader of the Belgian Visual Studio User Group. He added the performance issues that had beset Visual Studio several months ago have been fixed.

Other features in the 2010 release include capabilities for SharePoint application development via dashboards, as well as support for Windows 7 multitouch and ribbon interfaces. Tools also are offered for building applications for the Windows Azure cloud computing platform.

An IntelliTrace capability for developers and testers enables reproduction of bugs, so they can be eliminated.

"What IntelliTrace brings is the ability to record a video of your application," said John Robbins, a Visual Studio 2010 beta tester at Microsoft partner Wintellect. IntelliTrace, he said, makes debugging much faster.

Version 2010 also includes support for Web development via ASP.Net Model View Controller. Integrated phone design surfaces are offered for building Windows Phone 7 applications. Visual Studio 2010 includes the Professional, Premium, and Ultimate editions of the platform, as well as the new Visual Studio Test Professional 2010 tool for application testers.

"There's just all sorts of excitement around the testing tools," said Windows application developer Dave Zimmerman, administrator of the Minnesota Visual Studio User Group. Testing capabilities make it easier to automate test scripts for a larger variety of applications said Zimmerman, who manages projects at Microsoft partner Intertech.

The new version of Team Foundation Server, Microsoft's application lifecycle management server, features agile project planning capabilities via workbooks.

"We've done a huge amount to provide more insight into how a project's going," McBreen said.

Microsoft also will announce on Monday availability of the Microsoft-branded version of Team Explorer Everywhere, enabling Eclipse IDE users to use Microsoft's TFS as a back-end ALM server for software development projects.

.Net Framework 4 features capabilities for parallel programming.

"Parallel programming is an incredibly hard thing to do, so we provide a set of APIs to greatly simplify the models that you can think about," said McBreen. "We abstract a lot of that underlying complexity for the developer."

"The ability to do multithreaded applications [is] much easier" with .Net Framework 4, Zimmerman said.

Additionally, the client footprint in version 4 has been decreased by more than 80 percent, making it easier to get applications up and running faster, Microsoft said.

Version 4 also offers additional support for industry standards and increases language choice. The Dynamic Language Runtime in .Net Framework 4 lets developers choose between functional languages, such as C#, VB.Net, and F# and dynamic languages, including IronPython and IronRuby.

The framework can be installed side-by-side with .Net Framework 3.5, so existing applications will not break when developers install .Net Framework 4.

Silverlight 4, the latest version of Microsoft's rich Internet browser plug-in, adds capabilities such as extended out-of-browser capabilities and printing and webcam support.

"You can run an application directly from an icon on your desktop," said McBreen.

It also offers more than 60 customizable controls for building rich, interactive applications, Microsoft said. Charting capabilities are featured.

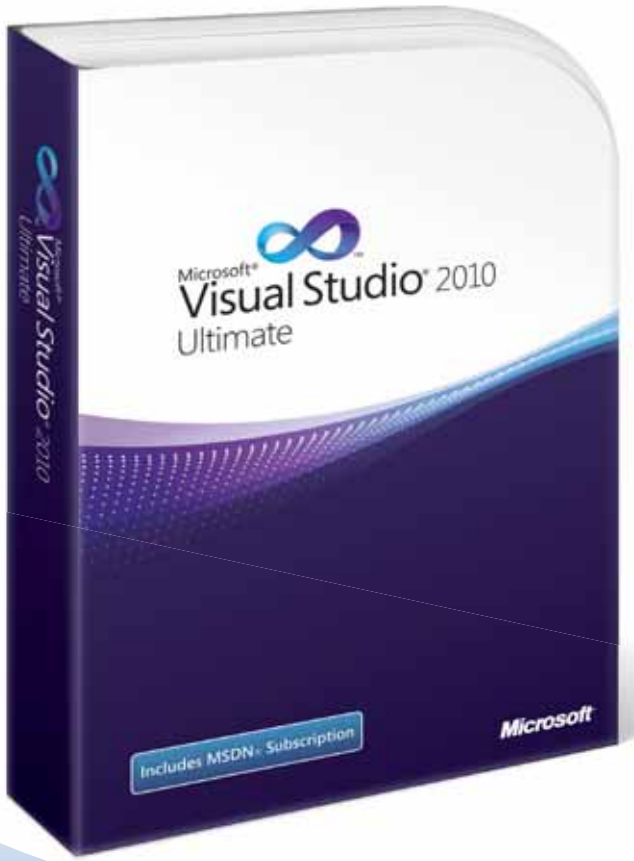
Silverlight 4 will be released to the Web later this week.

In a blog post late last week, Microsoft's S. Somasegar, senior vice president of the Microsoft developer division, stressed Microsoft's own use of Visual Studio 2010 technologies internally.

"One of the things that stands out clearly for me is our practice of dogfooding the various pieces of Visual Studio throughout the product cycle," Somasegar said.*" Here at Microsoft, we use the term 'dogfooding"* to refer to the internal use of a pre-release product in our daily work -- after all, until our product is good enough for us to use every day, it's not good enough for the rest of the world either!"

This story, "Microsoft to roll out Visual Studio upgrade," was originally published at InfoWorld.com. Follow the latest news in software development at InfoWorld.com.

Visual Studio 2010 and Silverlight 4 Released



Microsoft today is officially launching Visual Studio (VS) 2010 at the Microsoft Visual Studio Conference & Expo in Las Vegas. A major update of Microsoft's flagship integrated development environment (IDE), VS2010 incorporates an advanced user interface, powerful integration and customization technology, as well as improved ALM capabilities and extended support for key Microsoft developer platforms.

By Michael Desmond | REDMOND DEVELOPER NEWS | 04/12/2010

In addition to the VS2010 launch, Microsoft released the final version of .NET Framework 4, an important update to its managed framework that delivers a host of underpinning technologies for .NET application developers. Also released on Monday was the final version of Silverlight 4, Microsoft's rich Internet application (RIA) platform. Silverlight 4 boasts key improvements for business application developers, including out-of-browser execution and improved data binding.

"We're excited to celebrate the launch of Visual Studio 2010 with developers around the world today," Bob Muglia, president of the Server and Tools Business at Microsoft, said in a statement before the keynote. *"The functionality of Visual Studio 2010, .NET Framework 4 and Silverlight 4 creates a powerful and unique combination, opening up new opportunities for developers to build applications that take advantage of new and existing devices, as well as emerging platforms like cloud services."*

VS2010 in the Spotlight

Among the most visible new capabilities of VS2010 is the redesigned user interface, based on Windows Presentation Foundation 4 (WPF4). The new UI enables a more flexible and visual developer workspace, with precise rich text handling and multi-monitor support. VS2010 also promises significantly enhanced extensibility, with the Managed Extensibility Framework (MEF) enabling third-parties to seamlessly integrate functionality to the IDE.

VS2010 targets support for several critical Microsoft platforms. SharePoint tooling in VS2010 has been extensively overhauled, enabling a first-class experience for SharePoint developers in the new IDE. VS2010 provides native support for SharePoint projects and templates, as well as one-slick deploy/debug/test capability. Windows Azure is also aggressively supported in the new release, providing a family ASP.NET-like experience for developers creating apps for the cloud.

VS2010 also provides new tooling for Silverlight 4, Windows Phone 7 mobile development and ASP.NET Model-View-Controller (MVC) projects, which enable developers to separate the appearance and core business logic of Web applications. While tooling for Silverlight 4 is not present in the current version of VS2010, Microsoft expects to add a Silverlight 4 designer soon. Developers will be able to download the Silverlight 4 tooling free from Microsoft.

Application lifecycle management (ALM) is an important area of focus in VS2010, which no longer limits key test and collaboration functionality to Team System versions of Visual Studio. IntelliTrace, described by Dave Mendlen, Microsoft senior director of developer marketing as a "time machine" for developers and testers, captures an application's execution history so a bug can be literally played back on a tester's machine. IntelliTrace promises to help end the scourge of non-reproducible bugs. Improved test and collaboration resources are also included in the release.

"The enhanced testing features in Visual Studio 2010 automate the majority of common tasks and streamline the flow of information across our team," said Steve Schlonski, vice president, Xerox Global Services, Global Technology and Offering Development. *"This has led to a significant productivity increase; when you combine this with the ability to have a single unified view of project status, it dramatically drives down project risk."*

At the show, Microsoft released a new collaboration product, called Visual Studio Team Explorer Everywhere 2010. A Team Foundation Server (TFS) tool for developers working outside of Visual Studio, VS Team Explorer Everywhere appeals to shops doing Java development on Windows and on Mac platforms, said Microsoft.

At the show Microsoft also announced that it was extending its Ultimate Offer program for two weeks beyond the original April 12 end date. The Ultimate Offer gives existing Visual Studio customers with a premium MSDN subscription the ability to step up to a higher level SKU when they move to VS2010. For instance, a developer shop using Visual Studio 2008 Professional would be eligible to upgrade to Visual Studio 2010 Premium under the program.

About the Author

Michael Desmond is editor in chief of Visual Studio Magazine and former editor in chief of Redmond Developer News. He has served as senior editor of news at PC World and executive editor at Multimedia World magazine, and has written for dozens of publications and Web sites. Desmond has also written four computing books, including Microsoft Office 2003 in 10 Simple Steps or Less.

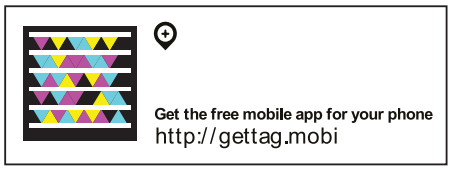
Additional Resources

Take a look at some of the additional benefits that Visual Studio 2010 has to offer for delivering innovative, high quality applications, get more details about what is included with Visual Studio Team Foundation Server 2010 and Visual Studio 2010 Ultimate and see what some of our partners have to offer.

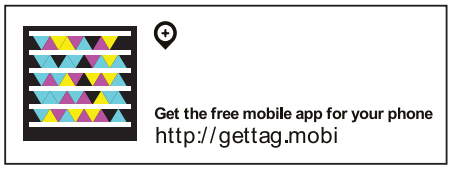
To download any of the papers you have read in this booklet go to *http://www.microsoft.com/visualstudio* and *http://www.microsoft.com/visualstudio/test*

To view videos related to Visual Studio ALM visit our YouTube Channel *http://www.youtube.com/VisualStudioALM*

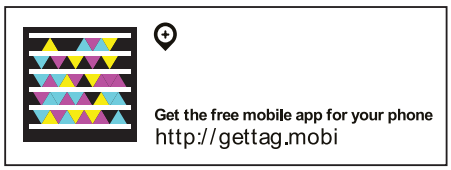
Try Visual Studio 2010 today - *http://www.microsoft.com/visualstudio/en-us/download*

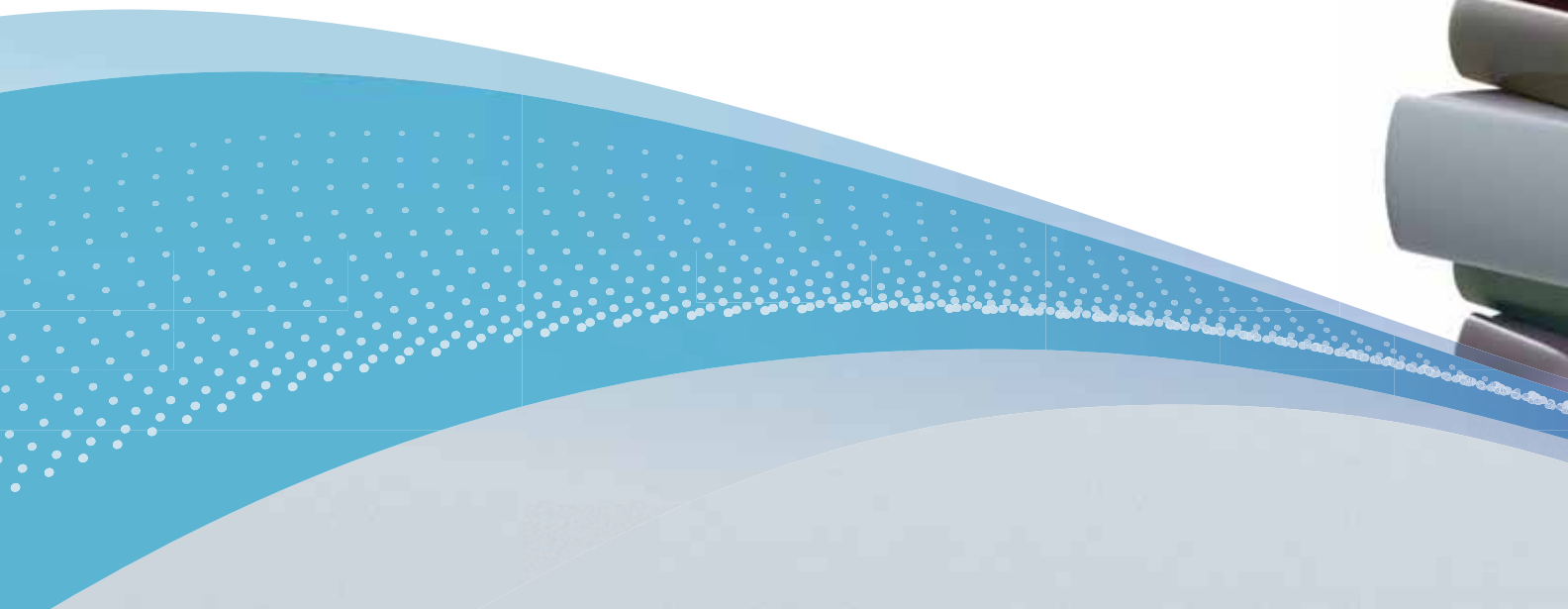


Follow Visual Studio on Twitter - *http://twitter.com/visual_studio*



Follow Visual Studio on Facebook - *http://www.facebook.com/microsoftvisualstudio*





Turn business ideas into solutions

Microsoft Visual Studio 2010 provides an integrated environment of tools and server infrastructure that simplifies the entire application development process. Deliver business results using productive, predictable, customizable processes and increase transparency and traceability throughout the lifecycle with detailed analytics. Whether creating new solutions or enhancing existing applications unleash your creativity with powerful prototyping, architecture and development tools that let you bring your vision to life targeting an increasing number of platforms and technologies including cloud and parallel computing. Realize increased team productivity by utilizing advanced collaboration features and use integrated testing and debugging tools to find and fix bugs quickly and easily creating high quality solutions while driving down the cost of solution development.

Microsoft® Visual Studio® 2010

Solution benefits

- **Increase efficiency across the Application Lifecycle**

  Project artifacts are stored in a central repository that facilitates in context collaboration reducing waste in hand-over time between tasks and streamlines the development process allowing team members to focus on delivering value over transitioning information between roles.

- **Manage Projects in Real-time**

  Powerful reporting and rich dashboards provide historical trending and real-time visibility into overall project health. Real-time metrics and leading indicators give you early warnings of potential problems and enable you to be proactive and to make data-driven decisions and course corrections.

- **Transparency and Traceability**

  Visual Studio 2010 provides full traceability and lets you track progress and quality back to business intent and customer requirements. In addition, seamless integration with Microsoft Project, enable business stakeholders and project managers to gain insight into inflight projects to understand how they support the business needs.

- **Focus on Quality**

  Powerful automated test and diagnostic features like IntelliTrace™ make the "no-repro" discussion a thing of the past and effectively eliminate the endless back-and-forth and wasted time to reproduce bugs. Test Impact Analysis helps teams to prioritize and focus their testing efforts and Gated Check-in prevents costly and time consuming build breaks from happening.

- **Align Test Professionals with the Application Lifecycle**

  Visual Studio Test Professional 2010 is a specialized toolset that delivers a brand new experience for test planning and manual test execution. Help eliminate waste across the application lifecycle and embrace change through seamless integration with Team Foundation Server – including actionable bugs , Test Impact Analysis, and Fast Forward for Manual Testing. Align test teams with the application lifecycle, break the silos across your development and testing/quality assurance teams and enable them to collaborate more effectively.

- **Reduce Operational Costs by Virtualizing Test Environments**

  Visual Studio 2010 Lab Management can help you drastically reduce costs associated with configuring, provisioning and maintaining test environments. Seamless integration with Team Foundation Server enables teams to rapidly provision multi-server virtual environments at a known state for test execution and build automation, effectively reducing wasted time and resources in your development and test lifecycle.

- **Consolidate Disparate Environments**

  Visual Studio Team Explorer Everywhere 2010 provides seamless Team Foundation Server support for Eclipse users and enables teams to reduce costs by consolidating disparate version control, work item tracking and automated build systems.

- **Continuous Process Improvement**

  Visual Studio 2010 can help you drive incremental improvement of products, processes, and quality over time, with the goal of reducing waste to improve resource efficiency, product quality and customer satisfaction. Support for concepts like Continuous Integration, automated code analysis and code reviews, and build and test automation coupled with flexible process templates and trending data enable teams to continuously monitor and improve facets of their application delivery process.

- **Embrace the Agile Application Lifecycle**

  With Visual Studio 2010 it has never been easier for software development teams to adopt Agile software development methodologies like Scrum. The Product Backlog can be used to create and manage the product stories while the Iteration Backlog can be used to manage the distillation of stories to tasks, plan iterations, estimate the team's velocity, load balance across the team, and track progress.

- **Deploy Applications with Confidence**

  Visual Studio® Load Test Virtual User Pack 2010 lets you simulate real-world application performance, stress and load conditions. Teams can validate that their solutions are ready for customers and that the applications will behave in a predictable manner under various conditions.

Reduce risk, Increase reward.

Take quality assurance to the next level with the Microsoft Visual Studio 2010. This breakthrough tester-developer collaboration experience supports complete plan-test-track workflows and full traceability of requirements and user stories. Capture rich diagnostics like IntelliTrace™ logs and indexed videos into your bugs; model real-world scenarios and loads for your applications; and further minimize regression testing using powerful UI automation and build integration features. Utilize cross-subject area reporting and Microsoft business intelligence tools for unparalleled visibility and insight into test progress and project health with leading indicators like Test Coverage and Bug Find/Fix Rate. Visual Studio 2010 is the end-to-end solution you need to improve software quality and effectively manage the way your teams work.

Microsoft® Visual Studio® 2010

Solution benefits

- **A Breakthrough Collaboration Experience for All Teams**

  Focus more on quality by streamlining development and test processes using Visual Studio 2010's integrated experience. Align test teams with the application lifecycle, break the silos across your development and testing/quality assurance teams, and give them the tools they need to work together more closely. Project artifacts are stored in a central repository that facilitates in-context collaboration, greatly increasing the fidelity of information transitioned between roles.

- **End-to-End Traceability of User Stories and Requirements**

  Easily access user stories and requirements through Team Explorer[2], which lets you trace them from birth through implementation to testing and onto release. Team Explorer also serves as your single launch point for cross-subject area reporting, allowing you to view links from test cases to user stories or requirements.

- **Extensible and Integrated Tooling Support**

  Prevent costly and time-consuming build-breaks and architectural validations using Gated Check-in. Rich, extensible APIs provide vast opportunities for extending artifacts and developing customized processes. The flexible architecture behind Microsoft Visual Studio 2010 Quality Offering helps you meet business-critical needs even as it improves the way your development and test teams work together.

- **Advanced Features for Test Prioritization**

  Test prioritization has never been easier. With a built-in capability to collect Test Impact data, you can access a list of recommended tests due to code changes. Now you can drive the right decision about when to pick up a new build, greatly reducing wasted time and effort.

- **Powerful Load Testing Tools**

  Guarantee that your solution can cope with real-world demands and behave in a predictable manner regardless of the situation. With rich profiling tools and real-world scenario modeling, you can stress your application throughout its development with simulated load conditions and validate that critical requirements such as end user responsiveness, system stability and scalability are met.

- **Streamline with Virtual Lab Management**

  Integrated with Lab Management, Test Professional 2010[1] helps significantly reduce the cost of setting up, configuring, provisioning and maintaining complex multi-machine test environments. Easily and rapidly provision virtual environments at a known clean state. Lab Management lets you have more test cycles per machine while ensuring predictable build-deploy-test workflows.

- **Automate Repetitive Tasks**

  Easily capture data-driven test iterations on your first pass. Use Fast Forward for Manual Testing to quickly navigate to the right places within your test cases to continue verification steps in subsequent passes. Plus, common steps can be converted to shared steps and be used throughout your test plan. Accelerate your testing lifecycle further with advanced build tools to automate an entire test plan or generate test code to test UI elements of your application using Coded UI Tests, all as part of an existing build process.

- **Rich, Actionable Bugs for Developers**

  Automatically capture extensible and rich diagnostic information like IntelliTrace[3] logs, indexed videos and screenshots for each new bug. When used with Visual Studio 2010 Lab Management, environment snapshots can be attached to bugs enabling developers to connect to the test environment. Now you can effectively reduce the effort consumed in the endless game of "bug Ping-Pong", making the "no-repro" discussion a thing of the past.

- **Reduce Risks Associated with Release**

  Leverage powerful Microsoft BI tooling like Excel, Reporting Services and SharePoint to deliver rich, detailed and interactive progress and status reports for overall visibility into project health and historical trending. Real-time quality metrics and leading indicators like Bug Find/Fix Rate, Test Coverage and Test Pass Rate provide early warnings of potential problems for proactive release management.

- **MSDN Subscription Included**

  An MSDN Subscription, included with both Visual Studio 2010 Ultimate and Visual Studio Test Professional 2010, is a convenient way to cost-effectively develop and test applications on the Microsoft platform with a simple licensing model and broad high quality information resources.

Quality code ensured

Make "no-repro" a thing of the past with integrated, extensible diagnostics and rich bugs that let developers fix issues – the first time and every time. Test Professional 2010[1] lets you explore applications without test step guidance so you can uncover complex bugs and manage more test scenarios, effectively increasing overall test coverage.

Simplicity through integration

With tight integration to Team Foundation Server as your collaboration hub, teams no longer have to work in silos using different tools, languages and processes to address the shared task of improving software quality. Leverage real-time reporting for data-driven decisions. Spot quality issues early and make course corrections as needed.

Creativity unleashed

Take control of your tasks with a rich, modern UI that enables a clean plan-test-track workflow. Focus on high value tasks like discovering defects and building your testing solution. With predictable builds, test prioritization features and traceability of user stories and requirements, you can finally embrace software change.

1 To use Test Professional you need Team Foundation Server (licensed separately unless purchased with an MSDN subscription)

2 To use Team Explorer you need Team Foundation Server (licensed separately unless purchased with an MSDN subscription).

3 IntelliTrace, available with Microsoft Visual Studio 2010 Ultimate, allows you to look back at a past state of your application with debugging information. Events of interest can be recorded through the IDE or Microsoft Visual Studio Test Manager 2010, available with Test Professional 2010 and Visual Studio 2010 Ultimate.

The ultimate toolset for software development

Microsoft Visual Studio 2010 Ultimate provides an integrated environment of tools and server infrastructure that simplifies the entire application development process. Deliver business results using productive, predictable, customizable processes and increase transparency and traceability throughout the lifecycle with detailed analytics. Whether creating new solutions or enhancing existing applications unleash your creativity with powerful prototyping, architecture and development tools that let you bring your vision to life targeting an increasing number of platforms and technologies including cloud and parallel computing. Realize increased team productivity by utilizing advanced collaboration features and use integrated testing and debugging tools to find and fix bugs quickly and easily creating high quality solutions while driving down the cost of solution development.

Microsoft® Visual Studio® 2010 Ultimate

Creativity Unleashed

Share your vision and build on the creative strengths of your team. Powerful prototyping, editing and visual design tools allow you to create what you can imagine. Spend less time debugging and more time creating code thanks to advanced code analysis and debugging tools. Explore new opportunities for your applications by unlocking the power of the processor on your local machine and in the cloud.

Simplicity Through Integration

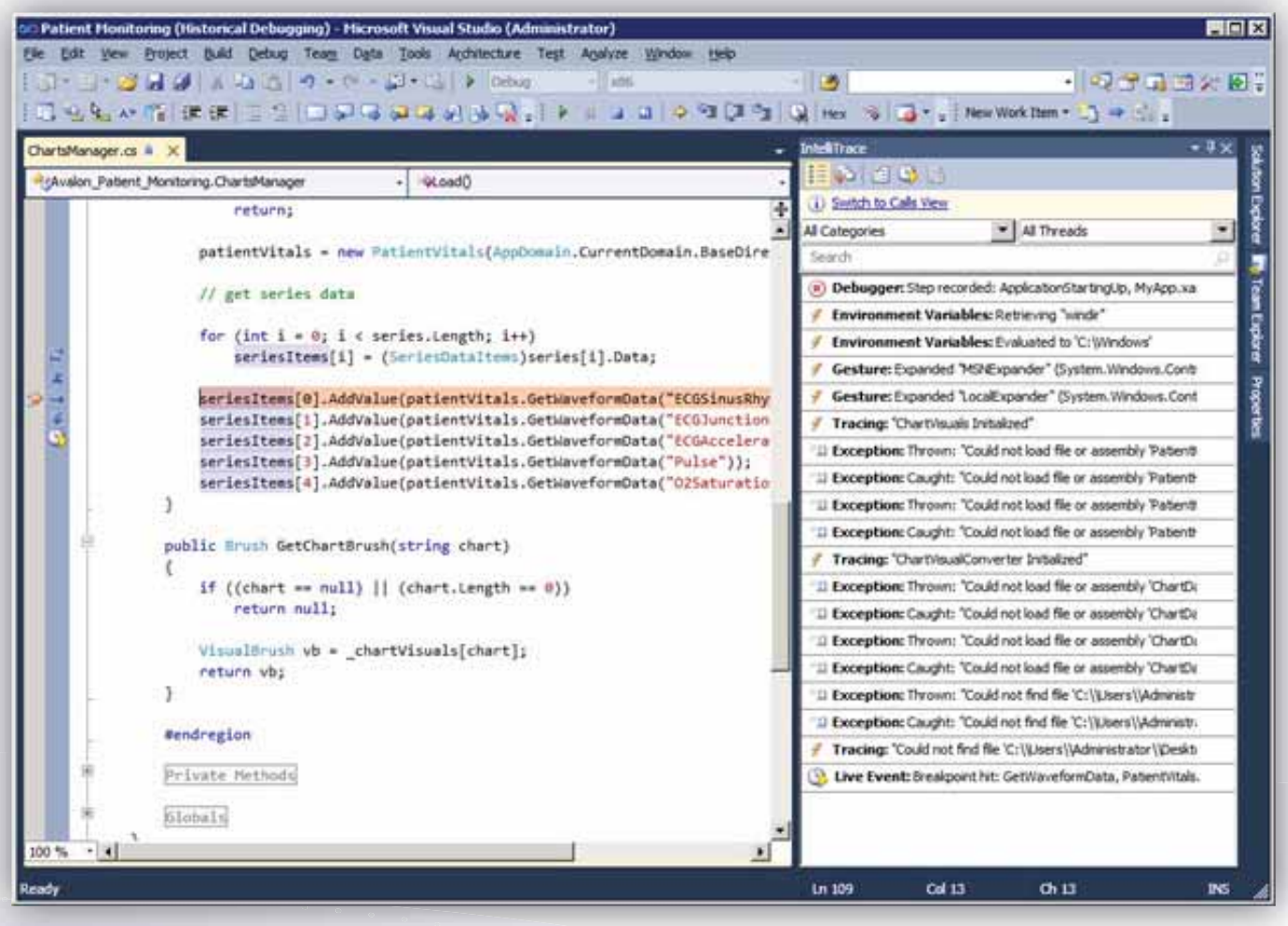
An integrated environment where developers can use existing skills to model, code, debug, test, and deploy a growing number of application types. Visual Studio 2010 Ultimate simplifies common tasks and helps all team members explore the depth of the platform and partner community.

Quality Code Ensured

Powerful tools for managing projects, maintaining source code and finding and fixing bugs. Both testers and developers can use manual and automated testing coupled with advanced debugging tools to ensure they are building the right application, the right way.

Reliably Reproduce Issues

IntelliTrace™ makes the "no-repro" discussion a thing of the past. Testers can file rich and actionable bugs with system information including an environment snapshot that enables developers to reproduce the reported bug in the state it was found.



Top Ten Reasons to Buy Visual Studio 2010 Ultimate

- **IntelliTrace™ Eliminates 'No Repro'**
  Step through code that was previously executed on the same or another machine in order to identify what happened during the code execution and significantly cut down on time spent reproducing a bug.

- **Understand Existing Architectures**
  The Architecture Explorer and UML sequence diagrams help you explore and understand your existing code assets and their inter-dependencies.

- **Ensure Architectural Compliance**
  Use layer diagraming to define and communicate logical application architecture and optionally enforce architecture rules with check-in policies as code is created.

- **Profile Application Performance**
  Measure the performance of your applications and locate performance bottlenecks in CPU utilization, Memory consumption, database interactions, Jscript/Ajax call patterns and concurrency synchronization. Use the performance profiler in conjunction with Load Testing to identify performance bottlenecks under stress and load.

- **Discover Common Coding Errors**
  Code Analysis is now simple to configure with customizeable rule sets that can be targeted at specific scenarios or areas of emphasis. Enforce rule sets with the code analysis check-in policy to reduce common coding errors in application or database code before they get into production.

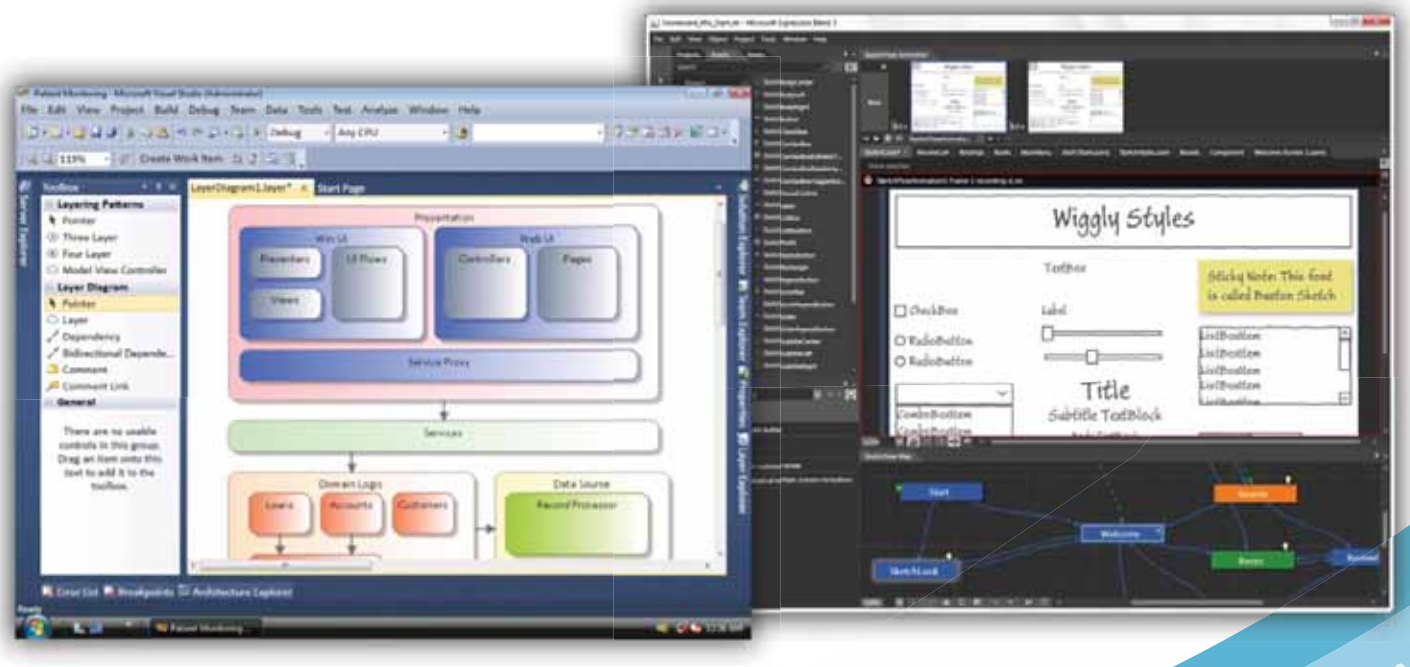
- **Prototype User Interface Ideas Quickly**
  By using SketchFlow in Expression Studio you can quickly deliver a functioning prototype that looks and feels like handwritten mockups.

- **Build Collaboration Solutions on Microsoft SharePoint®**
  New support for SharePoint development, including tooling for Web Parts, Lists, Workflows, and Events and more, so you can bring great new customized collaboration tools to your company.

- **Automate User Interface Testing**
  Use Coded User Interface Tests to automate the testing of UI elements in your applications. Visual Studio 2010 automatically generates test code that can be executed manually or incorporated as part of your build process to automate UI regression testing.

- **Create, Manage and Execute Tests**
  Test and Lab Manager lets you easily define your testing effort for a specific iteration in your project, run manual tests and measure your progress. In addition to creating and managing test assets like plans, suites and test cases you can also create and manage virtual lab configurations for your test environments.

- **MSDN Subscription Included**
  MSDN Subscriptions are a convenient way to cost-effectively develop applications on the Microsoft platform with a simple licensing model and broad high quality information resources.



Collaborate, communicate, deliver

Microsoft Visual Studio Team Foundation Server 2010 is the collaboration platform at the core of Microsoft's application lifecycle management solution. Team Foundation Server 2010 automates the software delivery process and enables organizations to manage software development projects throughout the IT life cycle. Team Foundation Server 2010 enables everyone on the team to collaborate more effectively, be more agile and deliver better quality software while building and sharing institutional knowledge. Project artifacts and data from work item tracking, source control, builds, and testing tools are stored in a data warehouse and powerful reporting and dashboards provide historical trending, full traceability and real-time visibility into quality and progress against business intent.

Microsoft® Visual Studio® Team Foundation Server 2010

Creativity Unleashed

Designed from the ground up as a collaboration platform to automate the software development process. Let Team Foundation Server 2010 manage your software development process and facilitate collaboration so that your teams are freed up from mundane non-coding tasks and empowered to unleash their creativity.

The collaboration platform at the core of Microsoft's application lifecycle management solution.

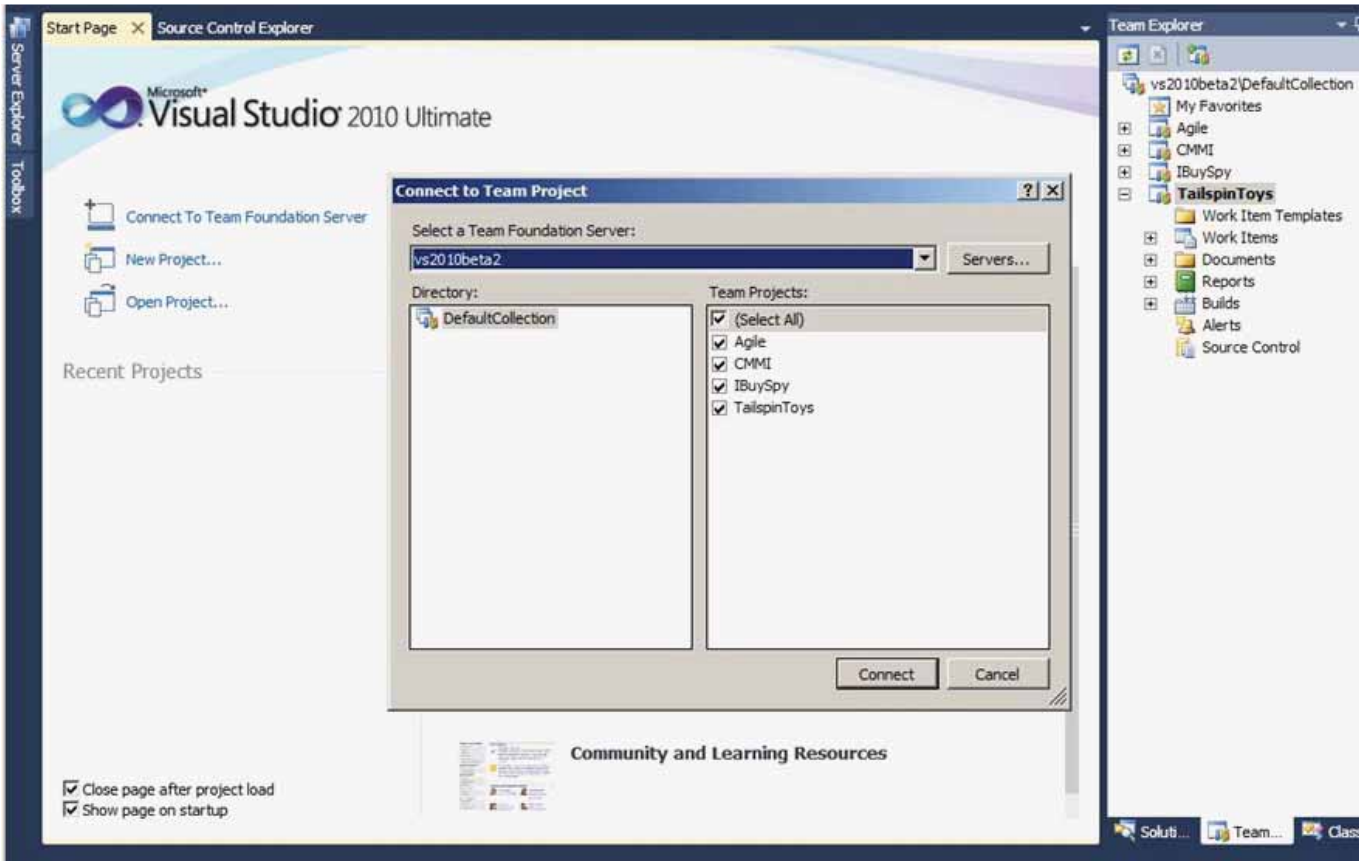
Visual Studio Team Foundation Server 2010 provides workflow automation and integrated processes, tools and project artifacts to simplify the collaboration of people across functional disciplines. The central repository enables rich collaboration in the context of the tasks the team is working on.

Simplicity Through Integration

Tools, processes and project artifacts come together to simplify the collaboration of people across various functional disciplines, anabling them to work better together and be more productive.

Quality Code Ensured

Monitor quality and progress in real time. Ensure that defects and regtressions are discovered at the earliest possible moment with powerful features like build automation and gated check-ins.



Top Ten Reasons to adopt Team Foundation Server 2010

- **Streamline The Flow of Data Across Your Entire Team**
  Project artifacts are stored in a central repository that facilitates in context collaboration reducing waste in hand-over time between tasks and streamlines the development process allowing team members to focus on delivering value over transitioning information between roles.

- **Reduce Risk with Real-time Visibility**
  Powerful reporting and dashboards provide historical trending and real-time visibility into overall project health. Real-time metrics give you early warnings of potential problems that enable you to be proactive and to make data-driven decisions and course corrections.

- **End-to-end Traceability**
  Define, query and report on custom relationships between requirements, work items and test cases. Full traceability lets you track progress and quality back to business goals and customer requirements.

- **Lightweight Agile Planning Tools**
  The new Excel Agile Planning Workbook makes it easy for teams to adopt Agile software development methodologies like SCRUM. Create and manage the user stories and product backlog, estimate the team's velocity, and break the project down into iterations. The Iteration Backlog enables you to plan iterations and track progress.

- **Project and Portfolio Management**
  Integration with Microsoft Project and Office Project Server enables business stake holders and project managers to gain insight into the health of inflight projects, understand how they support the business needs and help identify ways to improve existing processes.

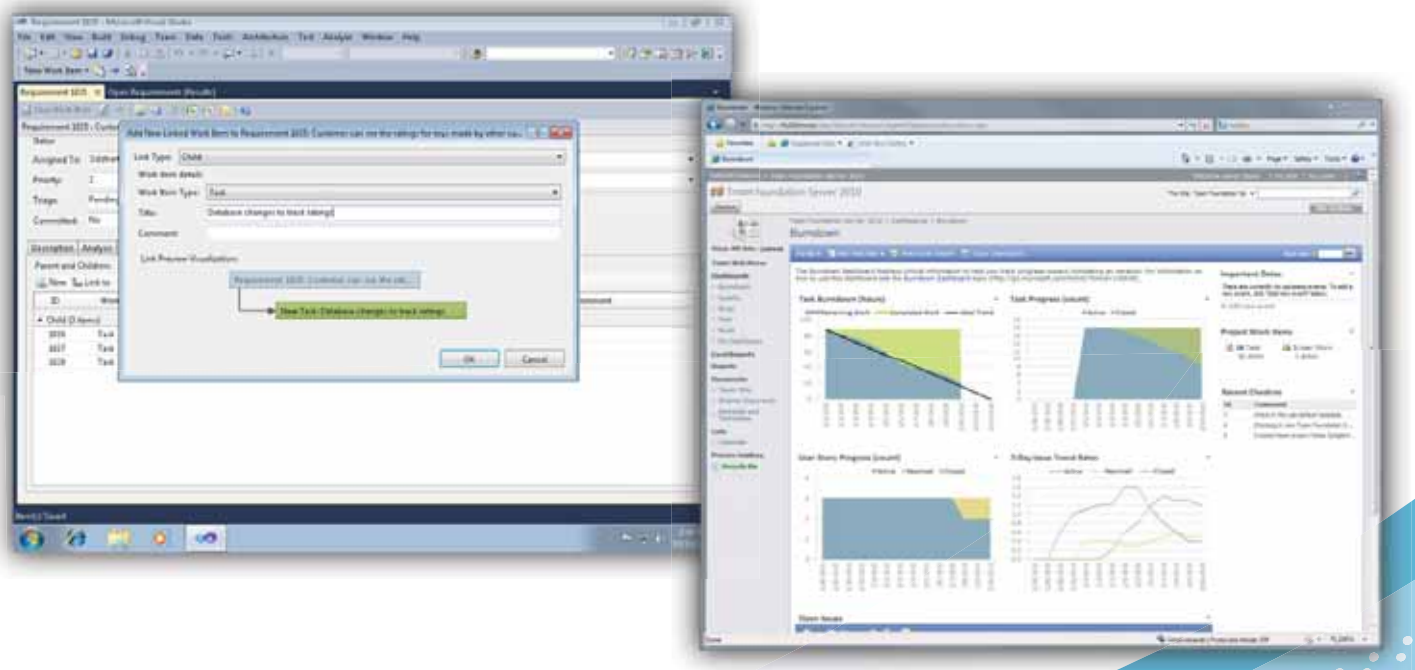
- **Simplified Installation for Smaller Teams**
  Smaller teams and individual developers can choose the new Basic Install option to leverage the power of Team Foundation Server 2010 without the footprint of the full installation.

- **Understand Parallel Development**
  Reduce the complexity in branching and merging with powerful new visualization tools. Understand the scope, organization and maintenance of your source code and easily identify, track and manage changes across branches.

- **Prevent Broken Builds**
  The new gated check-in feature helps teams working in the same branch to prevent costly and time consuming build breaks by testing code in isolation before it goes into the full repository.

- **Flexible Build Automation**
  Windows Workflow based builds with powerful features like build queuing and build agent pooling enable teams to easily customize, manage and scale out their build environments.

- **Enterprise Scalability**
  Network Load Balancing, 64-bit server support and new project collection isolation features enable large teams to scale Team Foundation Server 2010 installations to meet their demands.



Visual Studio 2010 Feature Comparison

	Visual Studio 2010			Visual Studio Test Professional 2010 with MSDN
	Ultimate with MSDN	Premium with MSDN	Professional with MSDN	
Development Platform Support				
Windows, Web and Cloud Development	•	•	•	
Office and SharePoint Development	•	•	•	
Architecture and Modeling				
Architecture Explorer, Layer Diagram and Dependency Validation	•			
Read-only diagrams (UML, Layer, DGML Graphs)	•	•		
Database Development				
Deployment, Change Management, Test Data Generation, Unit Testing	•	•		
Debugging, Diagnostics and Testing				
IntelliTrace™ (Historical Debugging), Web Performance Testing, Load Testing[1]	•			
Static Code Analysis, Code Metrics, Profiling, Code Coverage, Test Impact Analysis, Coded UI Test	•	•		
Unit Testing	•	•	•	
Test and Lab Management				
Microsoft® Test Manager 2010, Test Case Management, Manual Test Execution, Fast Forward for Manual Testing, Rich Actionable Bug Filing	•			•
Virtual environment setup & tear down, Provision environment from template, Checkpoint environment through Test Manager 2010	•			•
Team Foundation Server				
Version Control, Work Item Tracking, Build Automation, Team Portal, Reporting & Business Intelligence, Agile Planning Workbook, Team Explorer	•	•	•	•
MSDN Subscription benefits				
Priority support in MSDN Forums, MSDN Magazine, Flash newsletter, Online Concierge	•	•	•	•
Technical support incidents	4	4	2	2
Microsoft® e-learning collections (typically 10 courses or 20 hours)	2	2	1	1
Windows® Azure™ Platform	•†	•†	•†	
MSDN Subscription - Software for Production Use				
Microsoft® Visual Studio® Team Foundation Server 2010 plus one CAL	•	•	•	•
Microsoft® Office Professional Plus 2010, Project Professional 2010, Visio® Premium 2010, Expression Studio 3	•	•		
MSDN Subscription - Software for Development and Test Use[2]				
Windows (client and server operating systems), Microsoft® SQL Server®, Toolkits, Software Development Kits, Driver Development Kits	•	•	•	•
Microsoft® Office, Dynamics®, All other Servers, Windows Embedded	•	•		

† Azure benefits vary by subscription level; see the MSDN Subscription site for details: http://msdn.microsoft.com/subscriptions/ Subject to change and subject to availability.
1. May require one or more Microsoft® Visual Studio® Load Test Virtual User Pack 2010.
2. Per-user license allows unlimited installations and use for designing, developing, testing, and demonstrating applications.
UML is a registered trademark of Object Management Group, Inc.