Microsoft

Microsoft Visual Studio

Team System

Drive Predictability with Visual Studio Team System 2008

White Paper

May 2008

For the latest information, please see www.microsoft.com/teamsystem

CONTENTS

INTRODUCTION

Predictability provides advantages in all areas of business to the benefit of everyone. Organizations rarely thrive in completely chaotic environments. This is especially true when dealing with software development. Imagine if you will, a new developer being hired for a project and not knowing what they were supposed to do, how they were supposed to do it, and having no clue where to get this information. The solution to this problem is usually, “sit with another developer for a few days to learn the ropes.” This situation is less than optimal.

Or how about trying to apply or request additional resources for a project. Just getting additional resources isn’t going to help a project if you can’t figure out where to apply the resource or maybe the expertise the resource requires. By understanding current situations as well as any developing trends you can plan ahead to avoid potential problems.

Avoiding surprises is a key benefit of predictability. On numerous occasions development teams will work through the night going into the last few weeks before delivery because they were not aware of how much work was left to do. Typically it takes a heroic effort to complete the software on time—and quality almost always suffers. Microsoft® Visual Studio® Team System 2008 provides the ability to help foster collaboration between stakeholders also helps avoid surprises. By using Team System as a hub for communication, everyone is aware of the issues.

Team System helps drive success by enabling consistent and predictable processes. It helps expose trends on development projects, which enables teams to focus resources and actively (and preemptively) manage risks which may otherwise not be apparent. Without the ability to step back and clearly and easily view project details, many issues may go unnoticed—but not with Team System. Injecting predictability into the software development process leads to a lower cost of software development and a higher return on investment.

PROCESS

Many companies hear the word process and automatically assume a monolithic structure that everyone has to follow down to the last letter. And in many ways this is one issue that has caused many more projects to fail than would have otherwise. You might think this is a strange view until you consider that somewhere along the way “process” became associated with lots and lots of paperwork—almost as if the software was an afterthought. There is paperwork to tell you how to do your job, what the steps of the software development life cycle are, what your responsibilities are, who you report to, what you’re responsible for and lastly, keeping track of every last thing you do and how long it took you to do it.

As mentioned previously, a new hire is a big deal. Companies can spend a lot of time and money getting new hires up to speed and sometimes it takes weeks before they do anything productive. Team System helps you solve these problems through the use of process templates. Simply put, a process template contains documentation which is relevant for the development team. By default, a process template contains a list of roles, responsibilities, document templates, and a configured Microsoft® SharePoint® site, known as the Team Portal, and standard reports. These templates can be customized by adding in other documents, links, and custom configurations. When a developer comes in and says, “What am I supposed to do?” the Team Portal serves as a single location that you can point to which tells the developer everything they need to know.

Beyond that, your development process is integrated into the daily activities of the development team through the process template so it becomes ingrained. Work items (project tracking forms which record information) are integrated as part of the process template and contain the information necessary for the developer to work, communicate, and collaborate with others on the development team. You can customize work items to record the information you need to better help you refine your own process.

Many project teams work differently, even in the same organization. Teams might use a slightly different process to develop software from project to project. With Team System, everyone doesn’t have to use exactly the same process—but everyone can use the same tools. Team members can customize each Team System project to use only what the project team feels is necessary. This saves a lot of needless work and retraining as developers move from project to project.

Team System includes two built-in process templates to help teams get started. These are the *Microsoft Solutions Framework (MSF) for CMMI Process Improvement* and *MSF for Agile Software Development*. Either template can be useful for teams with no, or limited, process experience. For more mature teams, the process templates provide a platform for quickly customizing a process for the team’s particular needs. I’ll explain these templates in the next few sections.

MSF for CMMI Process Improvement 4.2

The CMMI/PI template is certified by the Software Engineering Institute—Capability Maturity Model (SEI-CMM) as providing the necessary processes to meet the CMMI Level 3 rating. Essentially this means that if you use the Microsoft Solutions Framework for CMMI Process, combined with the process template, you will have a standard, repeatable process for performing software development.

MSF for Agile Software Development 4.2

The agile template is a lightweight template which has simplified states and work items compared to the CMMI/PI template. The terminology matches those used by agile teams as well as the process used by those teams. This process stresses unit testing, frequent builds, and short iterations—all hallmarks of an agile process. Using the agile process enables your team to react to change more quickly and focuses on quality first.

Differences between the Agile and CMMI Templates

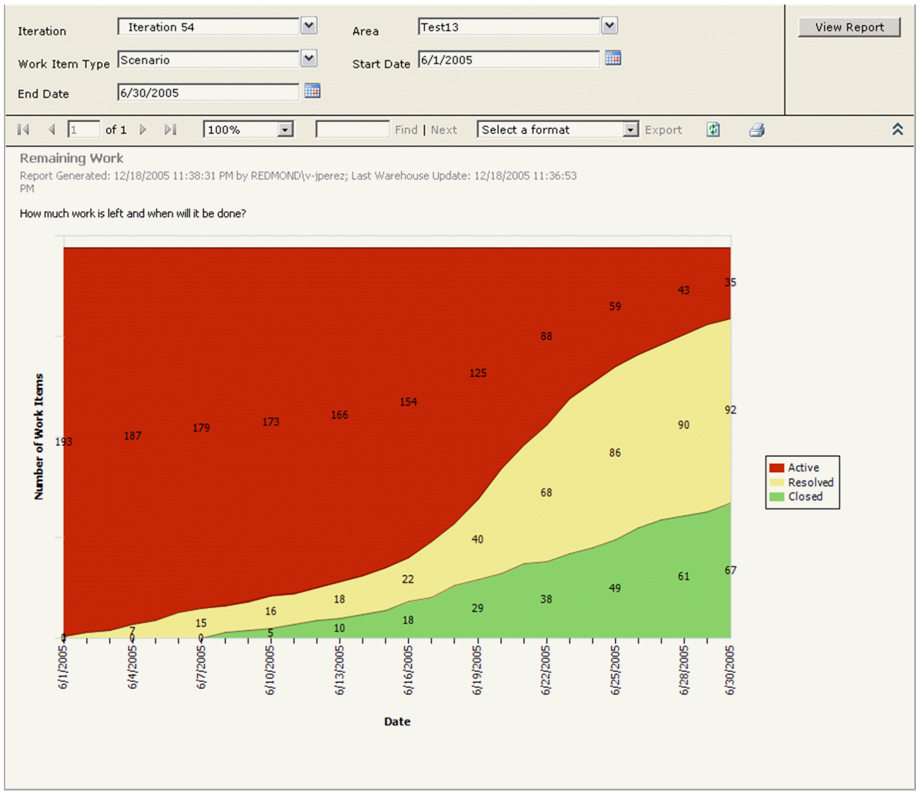
The CMMI/PI template provides several features which the agile template does not. It provides for a “proposed” state which is the initial state for all new work items. It also provides some additional reports over the agile template. Finally, the CMMI/PI template features a Change Request work item type, which enables you to use Team System as a change management tool.

For any given work item there are some differences in the information captured by each template. Also, the CMMI/PI template uses a Requirements work item instead of a Scenario work item for capturing requirements since agile development uses User Stories rather than the more detailed documentation typically found in a formal project. With Team System, however, no matter which template you use, the tool works the same.

RESOURCES

Have you ever felt that you are getting into trouble on a project? Maybe things aren't getting done as quickly as they should or the customers are dissatisfied with the software they are receiving. Lots of problems can materialize during the software development process. You might solve these problems by getting additional resources. But what type of resource do you ask for? Do you need more developers or more testers? How about better user participation or maybe a faster build machine? Maybe you need an independent developer to come and audit your team's code. Many teams have resource problems and don't have a good way to solve them. Team System can help you predict where and when you'll need resources.

This is where the reports in Team System really have a positive impact on the development process. Maybe you have a situation where your developers write code quickly but Quality Assurance cannot review the code fast enough. How could you know this? Team System has you covered—you simply run the Remaining Work report and you get an instant view of how much work is currently under development, how much is ready for testing, and how much has completed testing (Figure 1).


Iteration: Iteration 54
Area: Test13
Work Item Type: Scenario
Start Date: 6/1/2005
End Date: 6/30/2005

Remaining Work
Report Generated: 12/18/2005 11:38:31 PM by REDMOND\v-jperez; Last Warehouse Update: 12/18/2005 11:36:53 PM

How much work is left and when will it be done?

Figure 1. Remaining Work report

This report helps you determine if you should get more developers or more testers. If the number of items remaining to complete is not going down (red area in Figure 1), you need more developers than testers. But if the number of items remaining is going down and the number of items to test is going up (as in Figure 1—the yellow area), you need more testers. This is a simple and straightforward example of how the reporting in Team System helps teams do better.

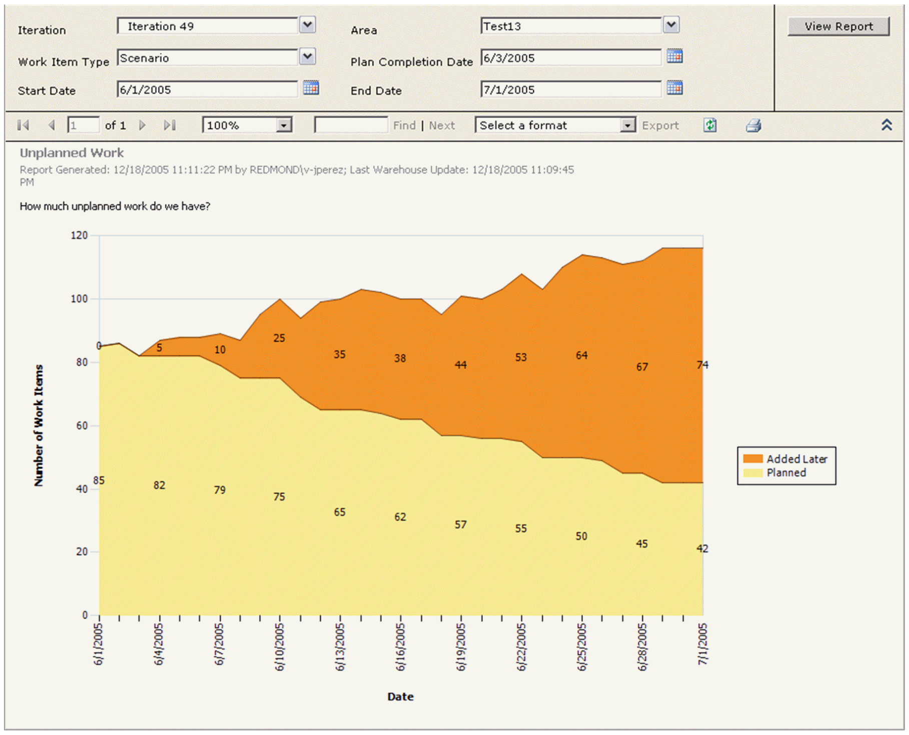
How about a more difficult situation? What if requirements are not being completed fast enough to give the development team enough work? Getting users' time to work with a development team can be difficult in the best of circumstances, but trying to prove that they are the bottleneck is almost impossible. Using a combination of reports in Team System you can show not only how many requirements are being completed but the number of overall requirements for the project and what stage they are in. These simple metrics can provide definitive proof of a need—without teams having to do any intensive work to gather this information.

AVOIDING SURPRISES

Team System offers some unique, pre-built report tools that will help teams avoid many of the typical surprises that historically derail a project. Many teams are often “caught off guard” by the amount of work left to finish as the release date approaches. Budgets are cut, release dates are advanced, scope increases, developers underestimated or other things occur which causes last minute scrambling. In fact, most development teams work nights and weekends getting the software completed just before the release. Typically, the team delivers the project with the caveat that the team will “fix it in maintenance”. In general, teams should not be caught off guard because there are indicators along the way that tell when something is not right. Team System also helps by providing a collaborative environment which helps eliminate the barriers to effective problem solving.

This type of last minute heroics approach to software development leads to dissatisfied customers, upset developers, and poor quality software. This has also led to one of the more intriguing “rules” in software development—75% of the money for development is spent on maintenance (keeping the business running) versus 25% on new development. Ideally you want the opposite situation—75% on new development versus 25% on maintenance. The “fix it in maintenance” approach is one of the reasons that this imbalance exists—and it is fixable.

Another common software development problem is scope creep. We’ve all experienced it or watched it cause a project to crash and burn. Many teams may not have even realized that increasing the scope of the project caused the problems they experienced. And even if they did, they couldn’t quantify it. Team System includes an Unplanned Work report that enables users to see what the scope creep is and what its’ impact is on the release schedule (Figure 2).



Iteration: Iteration 49
Area: Test13
Work Item Type: Scenario
Plan Completion Date: 6/3/2005
Start Date: 6/1/2005
End Date: 7/1/2005
Select a format

Unplanned Work
Report Generated: 12/18/2005 11:11:22 PM by REDMOND\v-jperez; Last Warehouse Update: 12/18/2005 11:09:45 PM

How much unplanned work do we have?

Figure 2. Unplanned Work report

With this information, teams can discuss the issue with the users to minimize and better manage scope creep.

Understanding the impact of changes on a development schedule is critical but typically there isn’t enough information to get a good handle on the problem. But what if you could get enough information to understand an impact without taking days to study the problem because you already had the metrics and just had to change the extrapolation a little bit? A case in point is the Unplanned Work report mentioned above. Another case is the Remaining Work report, which you can use to tell you how much work remains, but it also lets you see trends and perform extrapolations. This can also help you inform management of the impacts of their decisions.

Team System also features the Quality Indicators report, which provides information related to the quality of the code. Users don’t tend to like code released with a lot of bugs or other issues. By implementing unit testing and gathering metrics from the code you can provide detailed information such as the number of bugs remaining, the number of tests which failed (and passed), code coverage rates, and other information (Figure 3).

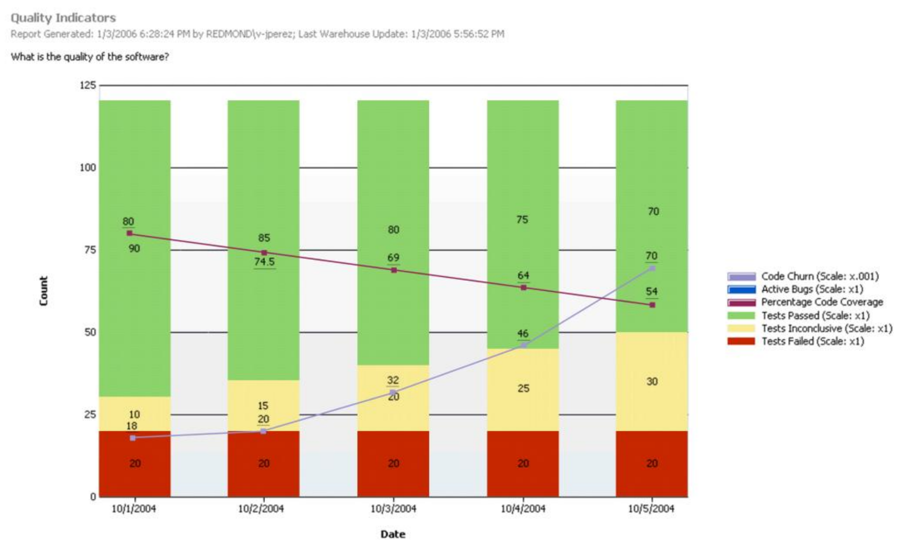


Figure 3 – Quality Indicators report

Figure 3 shows a disturbing trend: the number of passing tests is decreasing, the code coverage is decreasing, and the code churn is increasing. This indicates that the bugs being fixed are larger than maybe the team thought they were and that they are spending more time to close the bugs. While this isn't definitively pointing to a problem, it does indicate that there may be something wrong. It lets a team realize that, based on the information they have, they have cause for concern—no one will be surprised when it turns out a larger problem has been uncovered.

This report helps you get a handle on what's going to happen come release day and afterwards. You'll be able to judge just how much of your budget is going to go to maintenance and how much will be available for new work. It also lets you set customers expectations for the quality and completeness of the application in the initial release.

ESTIMATING

By its' very nature, estimating software projects is hard. I mean really hard. And there are many ways to estimate software projects. Software estimation experts recommend using a variety of techniques on a project and to look for a "convergence" between the estimates to determine if the estimates are good. One of the best techniques for estimating software projects is to use historical data. According to Steve McConnell in his book, **Software Estimation: Demystifying the Black Art** (Microsoft Press 2006), projects that use historical estimation tend not to have overruns. That's a pretty powerful reason to use historical information. But most organizations don't effectively maintain a history of effort.

Using Team System, it requires little effort to record the information needed to estimate future projects based on past results. Team System work items can record information such as the number of lines of code, length of time to complete, and other useful information. In this manner, organizations can use their own data for estimating future tasks. Using historical information also factors in political realities and organization-specific conditions. The more projects you do, the better your ability to estimate becomes. By extracting data from the Team System data warehouse and analyzing it, an organization can begin to improve its estimating ability and better control cost.

CONCLUSION

In Team System, process never gets in the way of developers doing what they need to do—build software. Team System never gets in the way because it isn’t onerous; it adds a few simple steps that end up saving time in the long run. And a simple, easy to use and well followed process leads to the one thing that all companies seek—process improvement. Until you have a process you can’t improve the process. Until your development teams use the process you can’t know what needs to be improved. Team System gets you on the path to creating better processes with less work because it works the way you do.

ABOUT THE AUTHOR

Jeff Levinson is the Application Lifecycle Management Practice Lead for Northwest Cadence which specializes in Team System and process improvement. You can reach Jeff at Jeff.Levinson@nwcadence.com.

This white paper was developed in partnership with A23 Consulting.