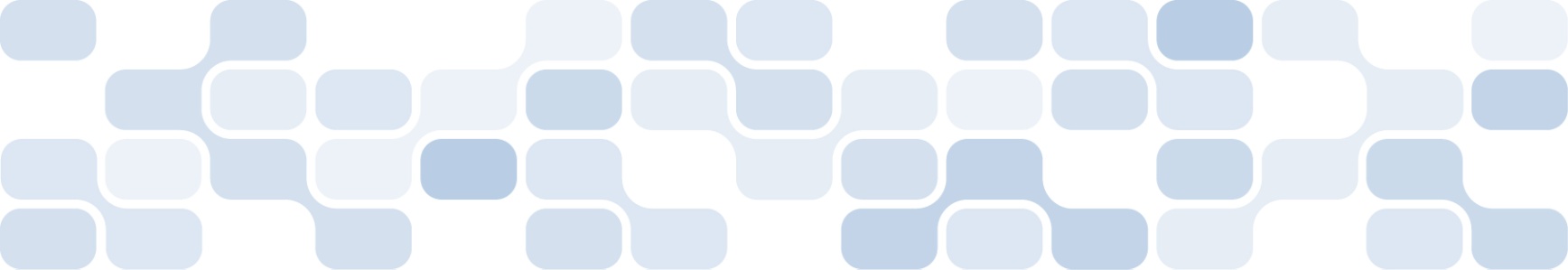


Quality from the UI to the Database with Visual Studio® Team System

White Paper

For the latest information, please see [www.microsoft.com/vstudio](http://www.microsoft.com/teamsystem)

****

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS SUMMARY.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form, by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2007 Microsoft Corporation. All rights reserved.

Microsoft, Excel, MSDN, SharePoint, SQL Server, Visual SourceSafe, Visual Studio, and the Visual Studio logo are trademarks of the Microsoft group of companies.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Contents

Introduction 1

Collaborate and Communicate 2

Drive Predictability 3

Manage Team Workflow 4

Use Familiar Tools 5

Ensure Quality, Early and Often 6

Integrate Work Frequently 7

Make Real-time Decisions 8

Summary 9

# Introduction

As a developer, quality should always be a primary concern when developing and maintain applications. Today, technology runs the world and its use in this role will only continue to grow. If your software lacks quality, it can have a significant impact, not just on your application but on the many applications and people that rely on it. The Visual Studio® Team System 2008 Development Edition and Visual Studio® Team System 2008 Database Edition provide tools to help you build quality in to new applications or root out the causes of poor code in existing applications.

How do you determine if your code has good quality? The simple answer is “the user is happy”. But what about passing security tests, meeting performance goals and developing refactorable and maintainable code? This is just a small subset of what many developers think about when they try to describe features of quality code and only leads to another question. How do you measure quality and provide proof that the code is truly at the level of quality desired? With the Visual Studio Team System 2008 Development Edition and Visual Studio Team System 2008 Database Edition you can positively qualify and quantify the term quality.

# Raising Code Quality with Code Metrics

You can use the Code Metrics tool in Visual Studio Team System 2008 Development Edition to measure the maintainability of an existing application and analyze the results to determine where the most complex code exists. You can use the Code Metrics results to drill down to the method level and view four distinct metrics that result in a calculated maintainability index.

**Class Coupling**

The Class Coupling metric provides you with the number of classes that a particular method or class is dependent on (the spaghetti in “spaghetti code”). If metrics indicate that you have a high Class Coupling, it means that a small change to one code class could have a significant ripple effect on numerous other classes. The Class Inheritance metric provides you with a number for the depth of inheritance for a particular class. Similarly to the Class Coupling metric, this Class Inheritance metric alerts you to changes to a base class that can trickle down and affect inherited classes.

**Cyclomatic Complexity**

The Cyclomatic Complexity metric provides you with the number of unique paths through a method. This metric raises a red flag for complex decision and loop logic which could easily break with minor changes as well as a feel for the level of testing required to adequately cover a method. The Lines of Code metric provides the number of executable lines of code within a particular method and alerts you to a place where you might consider breaking the code into smaller components to increase maintainability, readability, and reusability. Using the metrics together, you can easily pinpoint where the most complex code lies and the risky areas for changes that could easily result in broken logic and unhappy users.

**Performance Metrics**

The metrics just described can help you find the risk points, but how urgent is any of this information? Do your users use these methods when they run the application? By using the performance analysis tools of the Visual Studio Team System 2008 Development Edition, you can pinpoint the most called methods, the methods that take the longest to run, as well as numerous other performance metrics. Performance analysis works by running the application in a performance session and gathering all the data behind the scenes as you click around and use the application in the same manner that a typical user would. After you finish running a performance session, you can determine which functions run the most within the application, known as the Hot Path, with the simple click of a button. By cross referencing the performance analysis information with the code metric information, you have a true understanding of the most complex and riskiest points in the application. You now have solid information on the risk points in your code base and the understanding that changes to specific areas will require more resources. Is spending a significant amount of extra development and testing time for every change to a risky area worth it or is there a better solution?

# Quality from the UI to the Database

**Application Code Quality**

Instead of wasting extra resources, the code metrics help you develop a proactive solution to prevent problems in the first place. Since you know where the most complex and risky methods are, you can begin writing unit tests to start checking the core functionality of the application. Unit testing provides a repeatable low-level test of your code base that can run automatically on every check-in with integration into the Team Build functionality of Visual Studio Team System 2008 Team Foundation Server. Starting to write unit tests is a great step but how do you know what sections of code have been covered, and more importantly, which sections have not been covered? You learned in the code metrics section that the Cyclomatic Complexity metric indicates the number of unique paths through a method. This means that your unit tests must also cover every unique path in order to feel fully confident when modifying a method. The Code Coverage feature of the Visual Studio Team System 2008 Development Edition allows you to run your unit tests and walk through your code to see exactly which lines are covered and which are not. By developing and maintaining a high level of code coverage, you can greatly reduce the risk from changes to that code.

**Database Code Quality**

So far this paper has focused on a standalone application and this is of course not true for most enterprise applications today. The reality is that almost every application pulls data from a database. It is very likely that the stored procedures in the database contain a mixture of business logic and pure data pulls and both need to be tested if you want to increase quality and reduce risk. The Visual Studio Team System 2008 Database Edition allows you to write unit tests for stored procedures, functions, or even triggers within the database layer of your application. By testing the core layer of your database, you can make changes and catch issues immediately instead of trying to trace the problem all the way through your code. You also have the freedom to refactor any component of your database while still maintaining a level of comfort because your unit tests will tell you if anything broke. For more information on managing change in your database, see the whitepaper entitled “Managing Database Change with Visual Studio Team System 2008 Database Edition and Visual Studio Team System 2008 Development Edition”.

Now you can pull down a fresh copy of the production database every single time you need to run unit tests to insert unique data as well as make sure that the sensitive production data is scrubbed before it reaches the development environment. The Visual Studio Team System 2008 Database Edition also provides extremely powerful data generation features to produce consistent meaningful test data for all of your unit tests. Data generation also solves the age old problem of how do you test against production-like data in the same quantity as production without exposing sensitive customer data and failing audits.

# Building Quality into a New Application

This paper has discussed tools for maintaining code quality with a focus on maintaining an existing application. Of course, not every developer is focused on maintaining applications, and you can take advantage of the quality tools in Visual Studio Team System 2008 Development Edition and Visual Studio Team System 2008 Database Edition to develop a new application but with a radically different mindset. Instead of pointing out the risks in an application and trying to slowly introduce some quality checks into an existing application, you can use many features described above to develop a new application with quality at the forefront. For example, if you use Visual Studio Team System 2008 Development Edition to write unit tests as you develop new code, you can maintain code coverage levels to ensure that as the application grows, you have tested the core functionality. Your new code will use code metrics and performance analysis results as a preventative measure against spaghetti code and poor performance. And as a great boon to developers, the Visual Studio Team System 2008 Database Edition will generate data to give you a set of data to work with before a production version of the application even exists. Using these tools at the beginning of a development effort makes it easier for the developers who must maintain the application years later.

With new application development, you also may want to enforce specific coding standards such as naming or security standards. The Visual Studio Team System 2008 Development Edition provides you the Static Code Analysis tool, and allows you to check your code against a standard set of rules. By default, these rules are all warnings. You can change the rules that apply to the standards you are trying to enforce to errors, which will prevent compilation of the application if the code violates any of your standards. You can also tie this in with Team Build so that your rules are checked with every automated build and will fail if your code violates the rules. Database developers are in luck here as well because the Visual Studio Team System 2008 Database Edition also comes with static code analysis. By default the database rules are a set of naming, security, and performance rules that can be set up similarly to the Visual Studio Team System 2008 Developer Edition. However, with the Visual Studio Team System 2008 Database Edition, you can also define your own custom rules to match exactly what you need.

# Summary

So what is the big picture? Developers can choose from a lot of various tools that have some similar features, but what sets Visual Studio Team System 2008 apart? Visual Studio Team System 2008 integrates its tools into an IDE that you are already familiar with. On top of this, your development team benefits from powerful collaboration features when you start to use Visual Studio Team System 2008 Development Edition and Visual Studio Team System 2008 Database Editions in correlation with the features of Visual Studio Team System 2008 Team Foundation Server. When using the full blown experience of source control, work items, automated builds, and automated tests, masses of data is collected behind the scenes for powerful reporting. You can start to ask questions like, “What is our code coverage trend?”, “What source code changes and work items were in the last build?”, and “Which file has the most associated bugs?” You can answer these questions and more using real metrics from the use of Visual Studio Team System 2008 Team Foundation Server with Visual Studio Team System 2008. With an integrated and powerful toolset, you can start increasing code quality and customer satisfaction as soon as you finish the installation process.

For more information, visit the Web site at <http://www.microsoft.com/defyallchallenges/teamsystem>.