



Microsoft Test Manager Visual Studio 2012

Test Impact Analysis

Wednesday, July 18, 2012

Visual Studio ALM Rangers

Anutthara Bharadwaj, Hassan Fadili, Richard Fennel, Richard Albrecht

Microsoft Corporation

Visual Studio ALM Rangers

This content was created by the Visual Studio ALM Rangers, a special group with members from the Visual Studio Product Team, Microsoft Services, Microsoft Most Valued Professionals (MVPs) and Visual Studio Community Leads.

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

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 document is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Microsoft grants you a license to this document under the terms of the Creative Commons

Attribution 3.0 License. All other rights are reserved.

© 2012 Microsoft Corporation.

Microsoft, Active Directory, Excel, Internet Explorer, SQL Server, Visual Studio, and Windows are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.

Table of Contents

| | |
|---------------------------------------------------------------|----|
| Foreword by Anutthara Bharadwaj | 5 |
| Introduction | 6 |
| Overview | 6 |
| Visual Studio ALM Rangers | 6 |
| Understanding the Epics and Personas | 7 |
| Overview | 7 |
| Personas | 7 |
| Customer types | 7 |
| Scenarios and Guidance Cross Reference | 8 |
| Christine - “Tester” | 8 |
| How to Best Use Manual Testing and Test Impact Analysis | 9 |
| What is Test Impact Analysis | 9 |
| Enabling the Gathering Test Impact Data | 9 |
| Enabling Test Impact in the Team Build | 9 |
| Enabling Test Impact Analysis for Manual Tests | 10 |
| Gathering the Baseline Manual Test Coverage | 15 |
| Running the Build | 15 |
| Running a Suite of Manual Test | 15 |
| Using Test Impact Data | 16 |
| View Impacted Tests in the Build Report | 16 |
| Recommended Test | 17 |
| Summary | 19 |
| References | 20 |

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

Table of Figures

| | |
|---------------------------------------------------------------------------------|----|
| Figure 1 - Setting the Test Impact property of a build definition | 10 |
| Figure 2 - Selecting the Test Settings..... | 11 |
| Figure 3 - Enabling Test Impact data collector | 11 |
| Figure 4 - Selecting which process to monitor for Test Impact | 12 |
| Figure 5 - Selecting which processes to monitor for test Impact | 13 |
| Figure 6 - Enabling Test Impact proxy..... | 14 |
| Figure 7 - Enabling ASP.NET Client proxy for IntelliTrace and Test Impact | 15 |
| Figure 8 - Running a set of tests | 16 |
| Figure 9 - Build Report showing impacted manual tests | 17 |
| Figure 10 - Listing recommended tests in MTM..... | 18 |
| Figure 11 - Tests reset to the active state | 19 |

Table of Tables

| | |
|--------------------------------------------------------|---|
| Table 1 - Christine Scenarios to guidance mapping..... | 8 |
|--------------------------------------------------------|---|

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

Foreword by Anutthara Bharadwaj

Wouldn't it be nice if every time a developer made a checkin, you could automatically figure out exactly the right set of tests to run? The test impact analysis scenario in Visual Studio ALM attempts to provide exactly this value to the end user. Whether you want to run tests as part of your automated build or as part of manual test runs in Microsoft Test Manager, the test impact analysis data diagnostic adapter enables users to identify a subset of tests that are impacted by changes between target builds. As adoption of VS ALM has grown, we have often heard asks from users to provide explicit detailed guidance on how to use the test impact analysis scenario to get to the set of recommended tests.

The Rangers TIA guide seeks to address this ask and provide detailed and comprehensive guidance around the two test impact analysis scenarios available in Visual Studio ALM. The scenarios are described in an easy to follow how-to style with detailed illustrations and snapshots to take you through the workflow. We hope you find this guide useful in your efforts to identify impacted tests automatically.

Anutthara Bharadwaj (Principal Program Manager Lead, Visual Studio ALM Testing)

Introduction

Overview

Test impact analysis with unit testing and manual testing was introduced in Visual Studio 2010 and was accessible from the Visual Studio IDE and Microsoft Test Manager. With Visual Studio 2012 test impact analysis is only accessible from the Microsoft Test Manager. This document will provide information on how to setup and use test impact analysis. Also the document will focus the guidance on how to best use manual testing with test impact analysis.

This guidance should be used in conjunction with documentation that accompanies the product and Microsoft Developer Network (MSDN) at <http://msdn.microsoft.com>.

Visual Studio ALM Rangers

Visual Studio ALM Rangers is a special group with members from the Visual Studio Product group, Microsoft Services, Microsoft Most Valued Professionals (MVP) and Visual Studio Community Leads. Their mission is to provide out of band solutions to missing features and guidance.

This guide is intended for Microsoft “200-300 level” users of the Test Impact Analysis feature in Visual Studio. They are intermediate to advanced users of Microsoft Test Manager Features in Visual Studio and have in-depth understanding of the product features in a real-world environment. Parts of this guide might be useful to novices and experts, but they are not the intended audience for this guide.

Understanding the Epics and Personas



WHAT'S IN THIS CHAPTER?

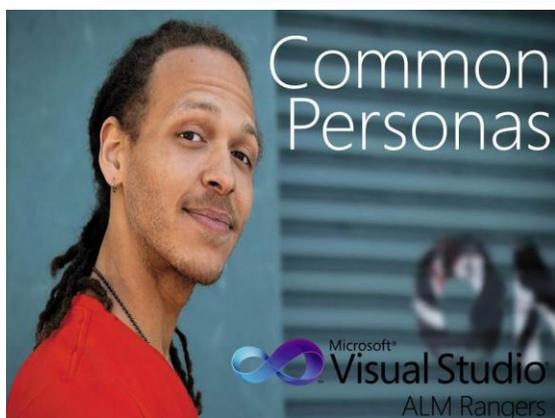
- Purpose of Epics and Personas in this document
- The Tester Persona
- Microsoft Test Manager Visual Studio 2012

Overview

This guidance is based on hypothetical customer profiles, personas, and scenarios (user stories). The intention is to demonstrate, in a realistic and convincing way, how personas leverage technology, along with this guidance, to perform a task and create a quantifiable outcome within their environment.

Personas

Refer to [Visual Studio ALM Rangers Personas and Scenarios](#)¹ for more information on the personas.



Personas at a glance

| | | | | | |
|--------------------------------------|-----------------------------------|-------------------------------|-----------------------------------|----------------------------|---------------------|
| Oscar Ops Lead | Paul DBVS Admin | Sam Release Manager | Sofia Subject Matter Expert | Mike Program Manager | Christine Tester |
| Jane Infrastructure Specialist | | Abu Build Master | Bill ALM Consultant | Akira Product Owner | Doris Developer |
| Dave TFS Administrator | Stephen Security Specialist | Dinesh Business Analyst | Alex Technology Consultant | | Garry Dev Lead |
| Shared Expertise Personas | | | | Team Personas | |

Customer types

Refer to [Visual Studio ALM Rangers Personas and Scenarios](#) for more information on the customer profiles.



Customer Profiles at a glance

| | | |
|------------------------|---------------------------|---------------|
| | | |
| Humongous Insurance | Consolidated Messenger | Troy Research |
| Large | Medium - Large | Small |

¹ <http://go.microsoft.com/fwlink/?LinkID=230942>

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

Scenarios and Guidance Cross Reference

Christine - "Tester"

Christine - Tester

- Works in small – large teams

- Experience

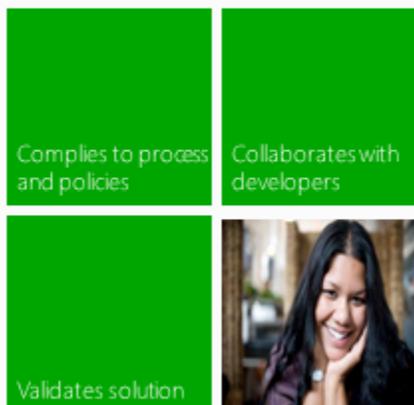
- Has Team Foundation Server experience
- Has Virtualization experience
- Has Visual Studio experience

- Mindset

- Complies to process and policies
- Collaborates with developers

- Importance

- Validates solution quality



| Scenario | Refer to page |
|---------------------------------------------------------|---------------|
| How to best use manual testing and test impact analysis | 9 |

Table 1 - Christine Scenarios to guidance mapping

How to Best Use Manual Testing and Test Impact Analysis



WHAT'S IN THIS CHAPTER?

- What is Test Impact Analysis?
 - Enabling Test Impact Data Collection
 - Gather baseline Manual Test coverage data
 - Using test Impact data to advise on which Manual Test to run
-

What is Test Impact Analysis

Though it is preferable to run all test available for a new build of a solution, running a full suite of tests, whether they be manual or automated, can be time consuming.

To provide an alternative to running the entire test suite or having to decide which tests to run yourself, since Visual Studio 2010, Test Impact Analysis is part of the Visual Studio [Premium and Ultimate SKU](#)²

Test Impact Analysis is a related concept to test code coverage. Test coverage tools tell you what percentage of the code base is covered by tests. By knowing which tests cover which blocks of production code, it is possible to say which tests need to be run if a given block of production code is edited. This technology can be applied to manual tests, as well as to unit tests, as long as suitable diagnostic data is gathered when a manual test is first run.



WARNING

A really common mistake is to try to make use of test impact analysis before a test has been successfully run at least one. Visual Studio/TFS cannot advise of test impact without the data stored from a successful test run. This applies equally to both manual and automated unit tests.

Remember also that test impact data is not collected when you file a bug, irrespective of whether the test is marked as passed or failed.

Enabling the Gathering Test Impact Data

Enabling Test Impact in the Team Build

Team Build is the key to the way Test Impact works with manual testing. When a build is run, as well as compiling the code and running the unit tests it can also perform Test Impact Analysis. It uses the data from previous manual test runs on past builds (of the same build definition) to see which manual tests need to be re-run due to changes in the production code.

Test Impact Analysis for a build is enabled by default. However, the setting can be checked in the build definition. The property **Analyse Test Impact** is in the **Advanced** section of the build definition.

² <http://msdn.microsoft.com/en-us/library/dd264992.aspx>

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

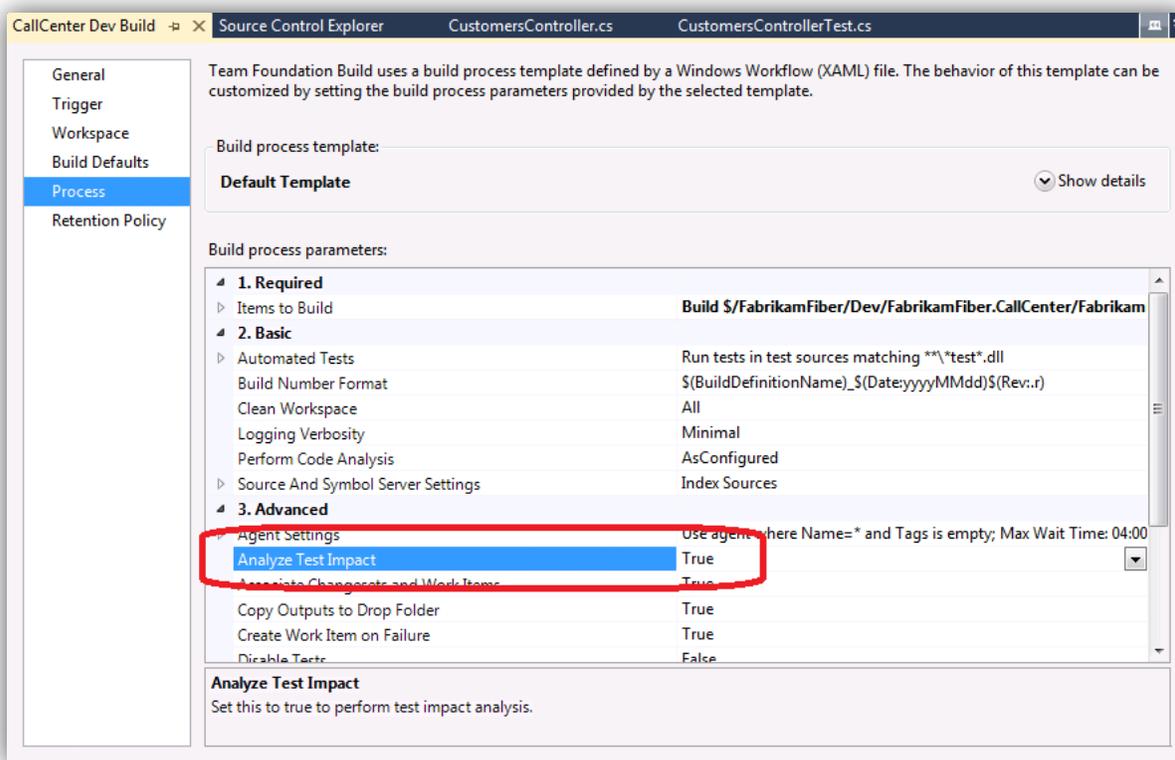


Figure 1 - Setting the Test Impact property of a build definition

Enabling Test Impact Analysis for Manual Tests

Once a build has been defined and run then it is possible to perform manual tests against it. However to gather the test Impact data the test impact data collector must be enabled within Microsoft Test Manager. The process to enable this is as follows

- Load Microsoft Test Manager
- Select your Team Project and Test Plan
- Select the **Plan** Tab
- Select the **Properties** Tab

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

- Select the **Manual runs: Test Setting** that you wish to use from the combo box (or create a new one)

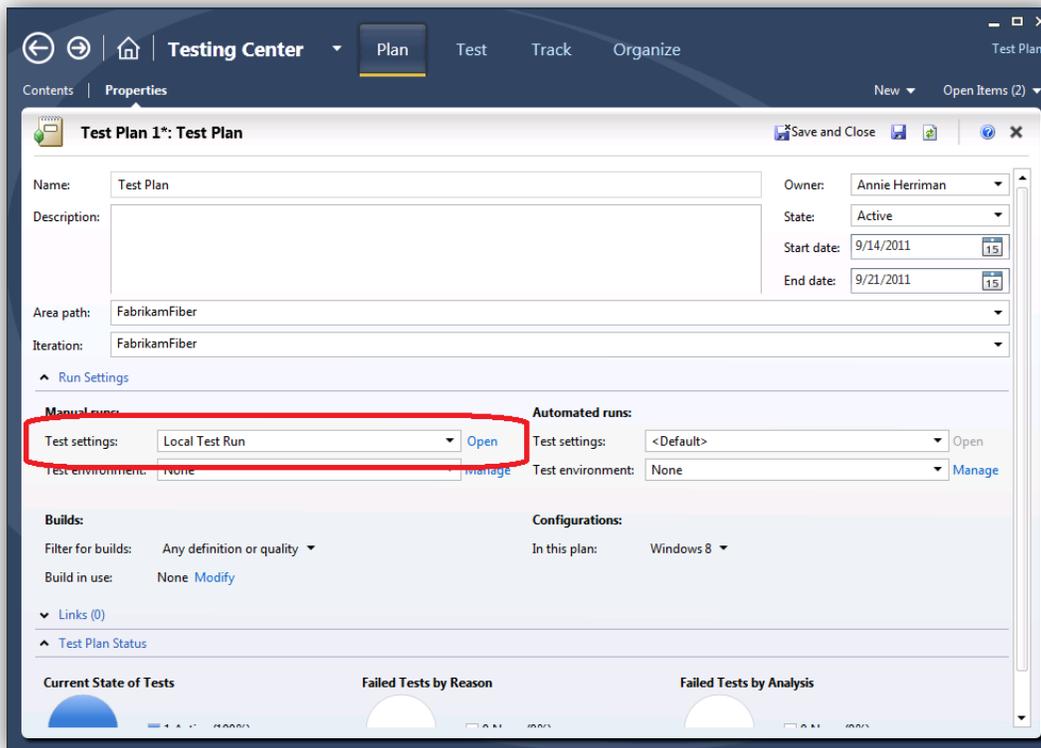


Figure 2 - Selecting the Test Settings

- Make sure that the **Test Impact** diagnostic data collector is enabled

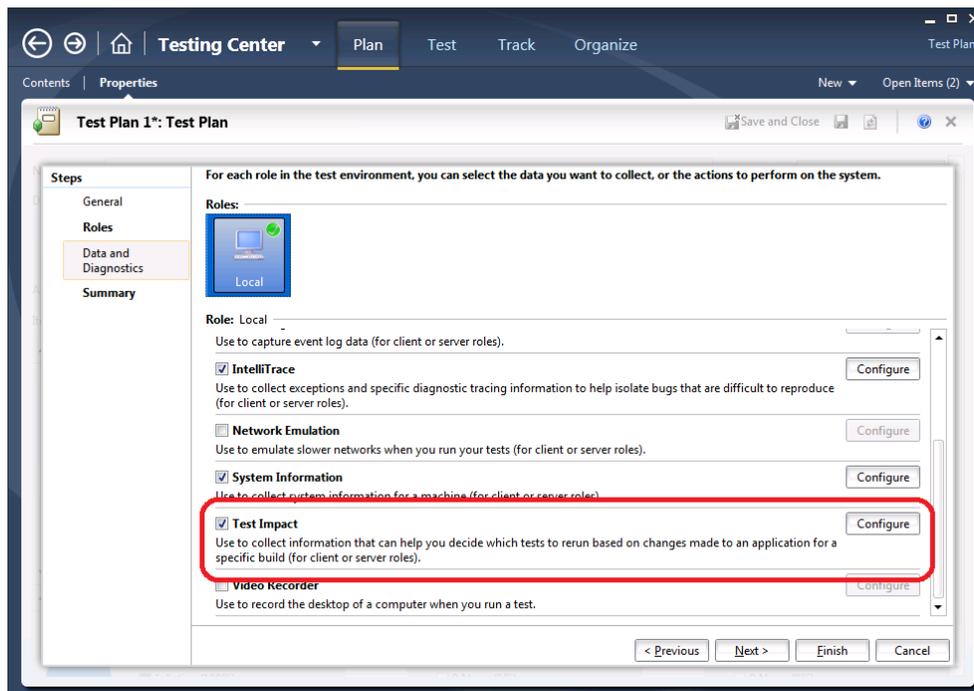


Figure 3 - Enabling Test Impact data collector

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

- You now have the option to configure what will be monitored by the collector i.e. select which assemblies and/or processes to monitor or ignore

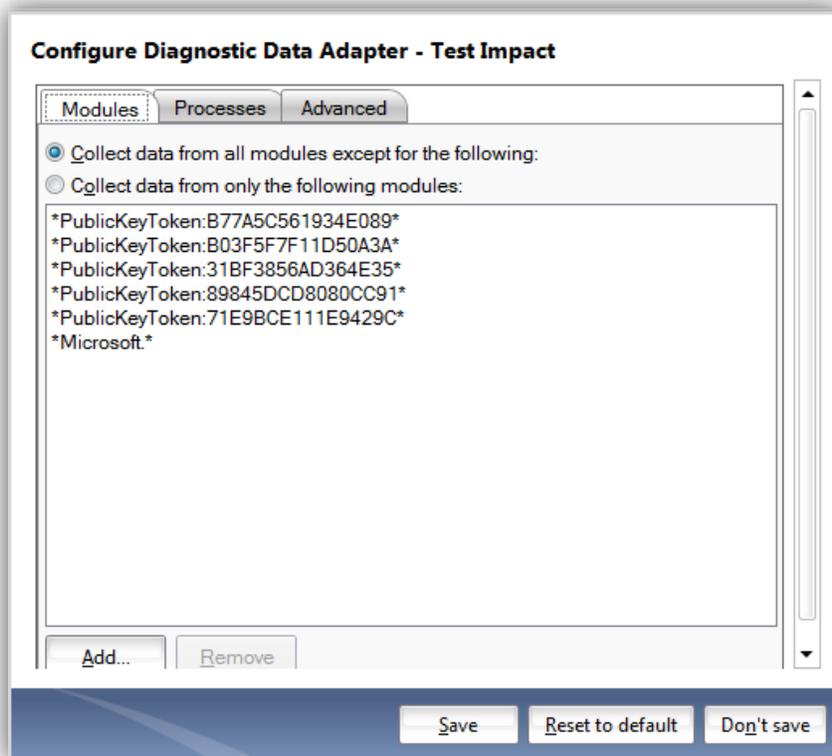


Figure 4 - Selecting which process to monitor for Test Impact

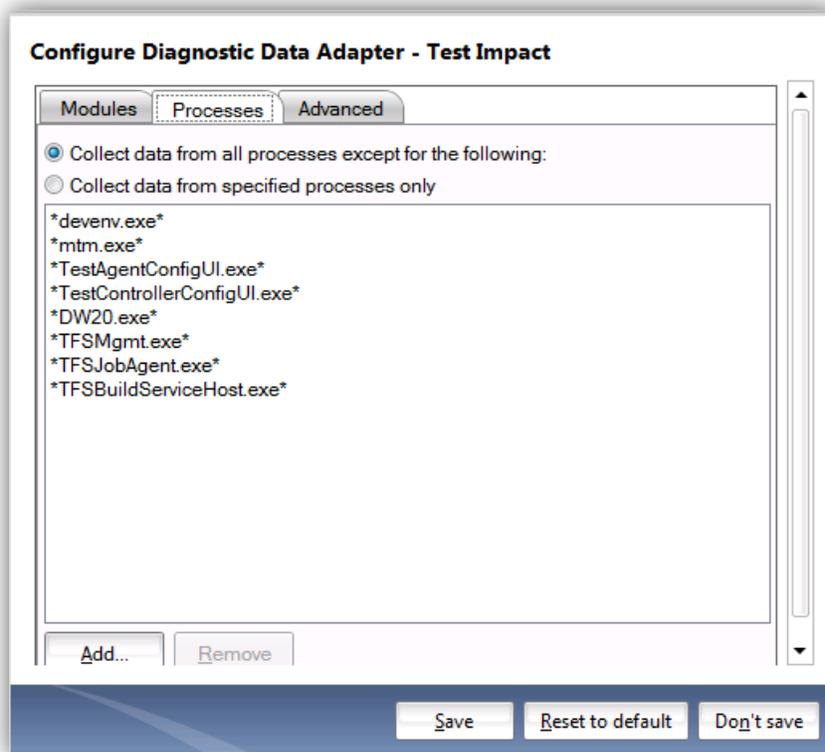


Figure 5 - Selecting which processes to monitor for test Impact

- On the **Advanced** tab of this dialog you are warned that if you wish to use Test Impact with an ASP.NET application on IIS then you must, as well as checking the checkbox in this dialog, also enable the **ASP.NET client proxy for IntelliTrace and test impact** collector.

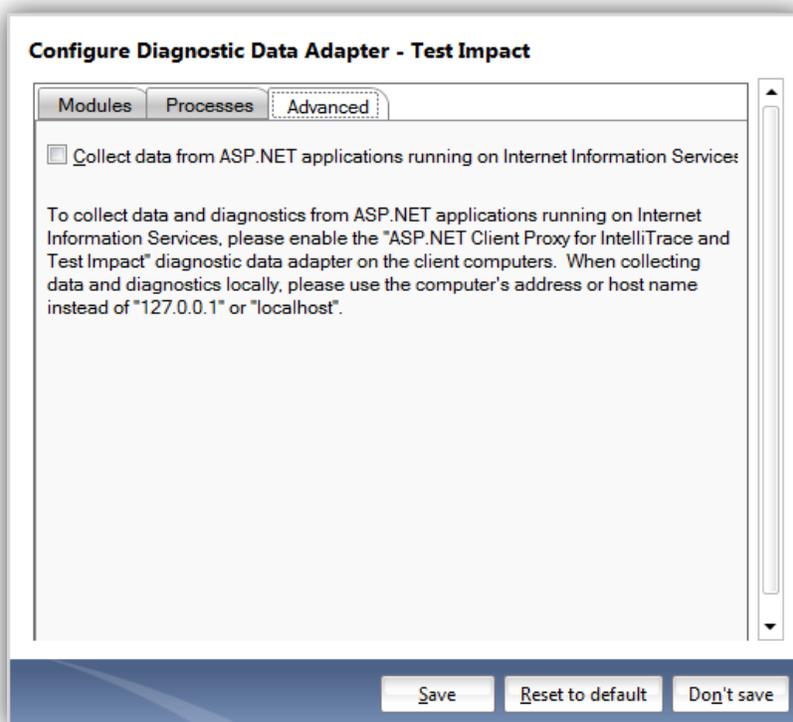


Figure 6 - Enabling Test Impact proxy

- The **ASP.NET client proxy for IntelliTrace and test impact** can be found in the same list as the **Test Impact** collector. Though there is a configuration button for the client proxy data collector, no configuration is available or required.

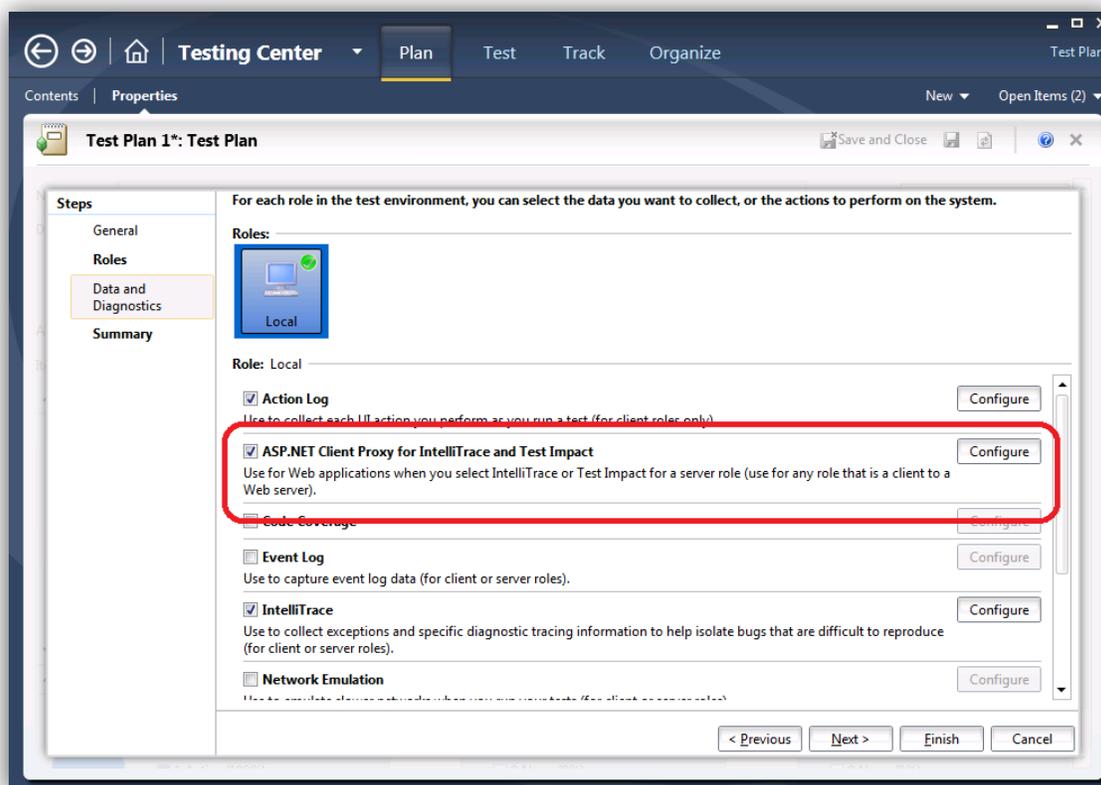


Figure 7 - Enabling ASP.NET Client proxy for IntelliTrace and Test Impact

- Once these changes are saved then tests can be run and Test Impact data will be gathered.

Gathering the Baseline Manual Test Coverage

Running the Build

Gathering the Test Impact data requires that the defined manual tests are run against a build produced by Team Build. Hence the first step in the process is to queue a Team Build that will act as the build for the baseline.

Running a Suite of Manual Test

Once a build has completed, and has been deployed to a suitable test system, manual tests can be run from within Microsoft Test Manager. The process is as follows

- Load Microsoft Test Manager
- Select your Team Project and Test Plan
- Select the **Test** tab

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

- Select the tests you wish to run and using the Run drop down select **Run with Options**³

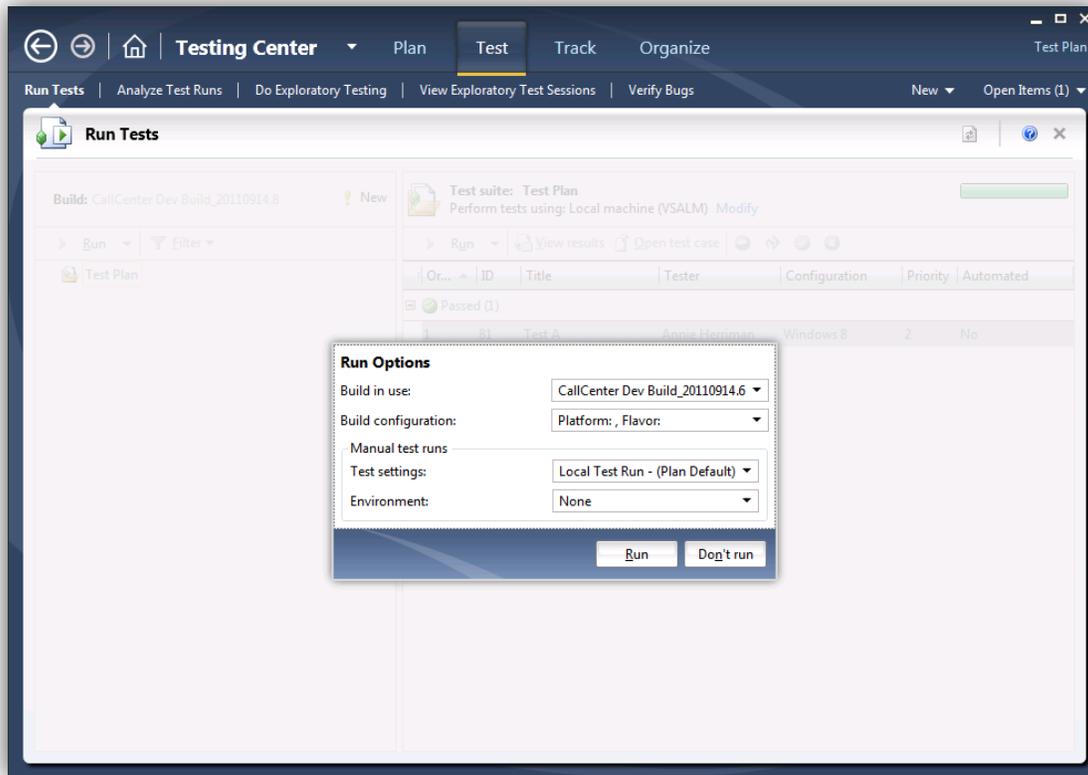


Figure 8 - Running a set of tests

- Make sure that the correct build and the test settings, configured previously, are selected. The selection of the build is key here; the manual test coverage will be associated with this build to provide the baseline
- Press the **Run** button and complete the manual tests. Remember the test must be completed without any failures, and no bugs must be logged for Test Impact data to be recorded.

Using Test Impact Data

Once the baseline has been taken, any future builds of the same type, that are configured to perform Test Impact Analysis, will highlight any tests that are impacted by a change to production code.

View Impacted Tests in the Build Report

A list of impacted tests, due to changes in the production code, can be seen at the foot of the build report.

³ We are using the **Run with Options** to make it obvious which build/configuration is in use, normally selecting **Run** would be adequate as the correct defaults would be selected

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

CallCenter Dev Build_20110914.9 - Build succeeded

View Summary | [View Log](#) | [Open Drop Folder](#) | [Retain Indefinitely](#) | [Delete Build](#) | [Requeue Build](#) | [<No Quality Assigned>](#)

Brian Keller triggered CallCenter Dev Build (FabrikamFiber) for changeset 22
Ran for 30 seconds (VSALM - Controller), completed 4.2 hours ago

Latest Activity

Build last modified by LOCAL SERVICE 4.2 hours ago.

Request Summary

[Request 1](#), requested by Brian Keller 4.2 hours ago, Completed

Summary

Default Configuration and Platform

- 0 error(s), 2 warning(s)
- \$/FabrikamFiber/Dev/FabrikamFiber.CallCenter/FabrikamFiber.Web/FabrikamFiber.Web.csproj compiled
- No Test Results
- No Code Coverage Results

Debug | Any CPU

- 0 error(s), 0 warning(s)
- \$/FabrikamFiber/Dev/FabrikamFiber.CallCenter/FabrikamFiber.CallCenter.sln compiled
- 1 test run(s) completed - 100% average pass rate (100% total pass rate)
- No Code Coverage Results

Associated Changesets

[Changeset 22](#), Checked in by Brian Keller

Impacted Tests

- 1 impacted test(s) of type "Manual Test Case"
"Test A" - impacted by: 1 code change(s)

Figure 9 - Build Report showing impacted manual tests

Recommended Test

It is also possible for a tester to ascertain the impacted tests from within Microsoft Test Manager. This is achieved as follows

- Load Microsoft Test Manager
- Select your Team Project and Test Plan
- On the **Track** tab, select the **Recommended Tests**
- Make sure the **Build to use** option is set to the build that will be under test. If it is not set correctly, use the **Modify** link to change it.
- Select the **Previous build to compare**. This should be the build for which there is Test Impact data available
- Press the **Recommended tests** button to see a list of the impacted manual tests.

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

- A list of impacted tests will be shown.

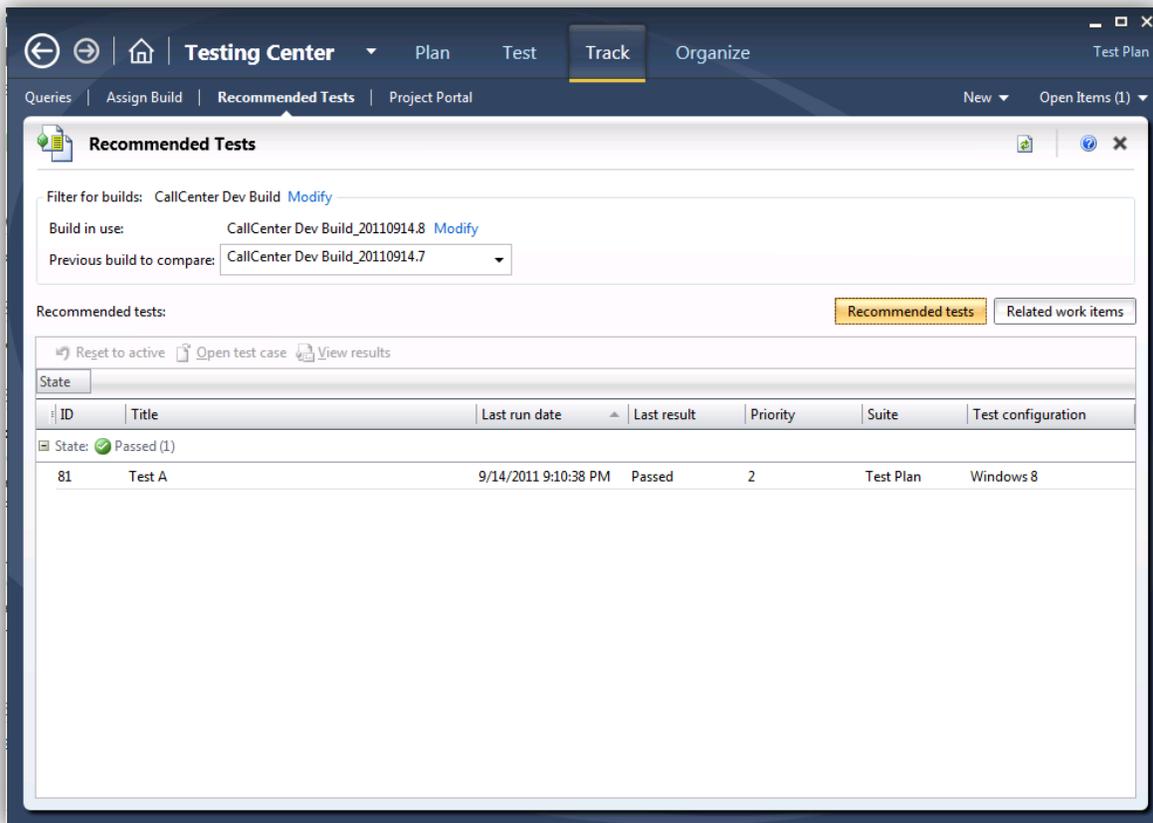


Figure 10 - Listing recommended tests in MTM

- If the tests within the list are selected, you can right click and use the **Reset to active** option. By doing this the tests will be set to the active state. They will be then shown the **Run** tab and can hence be re-run.

Microsoft Test Manager Visual Studio 2012 - Test Impact Analysis

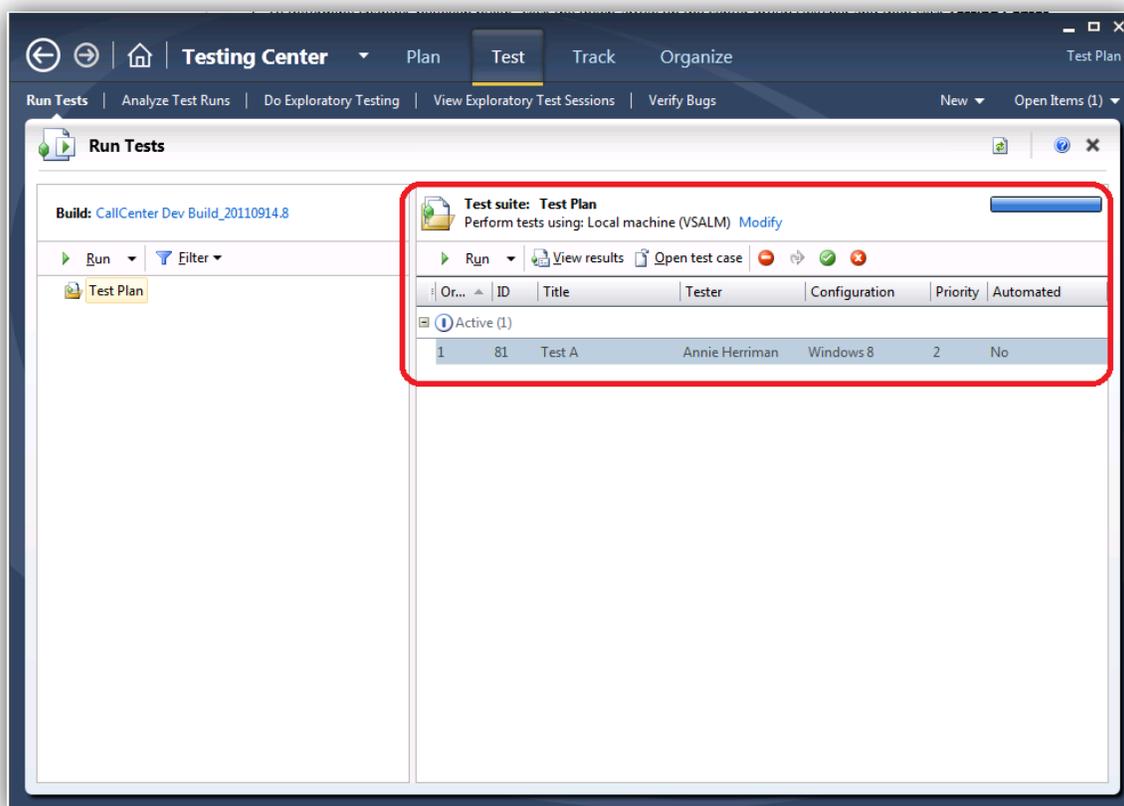


Figure 11 - Tests reset to the active state

Summary

In this section you have seen that Test Impact Analysis is not limited to unit tests, it can also be used for manual testing. Hence can be a great way to focus the efforts of a manual test team.

References

[http://msdn.microsoft.com/en-us/library/dd286743\(v=vs.110\)](http://msdn.microsoft.com/en-us/library/dd286743(v=vs.110))

[http://msdn.microsoft.com/en-us/library/dd286586\(v=vs.110\)](http://msdn.microsoft.com/en-us/library/dd286586(v=vs.110))

<http://www.tinyurl.com/almrangers>