TESTING

Microsoft Dynamics® AX 2009

Testing Guidance for Microsoft Dynamics AX 2009

White Paper

Date: September 25, 2008

http://www.microsoft.com/dynamics/ax

Table of Contents

Introduction

This paper outlines some basic testing guidance provided by the Microsoft Dynamics® AX 2009 internal development team in support of Independent Software Vendors (ISV) building product extensions for the product. Much of this guidance also applies to Microsoft Dynamics AX implementation partners and customers who are customizing the application.

Ensuring that an ERP application works after it is deployed at a customer site is an extremely challenging endeavor. The base application, which is both broad and deep, has a myriad of modules, features, software and hardware configurations. ISVs extend or modify the base application to provide functionality for specific market needs and verticals. Finally, implementation partners or customers do point customizations to address specific customer needs. This combination of base application, one or more ISV add-ons, and point customizations that operate in a unique hardware and software configuration is what businesses depend on for mission critical financial data and company operations. A challenge, indeed!

ISVs and implementation partners typically do not have much guidance or tooling available to help in their efforts to guarantee a quality deployment for customers. This paper is a first step in providing some guidance. Over time, this guidance will evolve and tools will come on-line to better support the AX ecosystem.

There are many factors that impact the quality of software. This paper will address only a few factors that are typically associated with the discipline of software testing.

The remainder of the paper has the following sections:

- Don't Test Quality In
- Testing Approaches
- To Automate or Not to Automate?

Don't Test Quality In

A common misconception is that software testing starts after the product is built. If nothing else is taken from this paper, let it be that you cannot 'test quality' into the product. A primary goal of testing is to provide feedback on the product as soon as possible. Identifying the issues in the requirements phase prevents them from being part of the design. Identifying the issues in the design phase prevents them from being coded. Identifying the issues during implementation prevents them from becoming part of the shipping product.

One point to emphasize here is that software test is a role to fulfill on a project team. It does not have to be a specific person. In smaller organizations, it is common for individuals to fulfill multiple roles. In this situation, it is important to identify someone who is wearing the software test or software quality hat during all phases of the project.

Peer Reviews

A primary tool during the requirements, design, and coding phases is peer review. A rigorous approach to peer reviews, software inspections are defined as "Peer review of any work product by trained individuals who look for defects using a well defined process" - (Wikipedia). Alternatively, peer reviews can be informal meetings to discuss the design of a particular new feature. The most important aspect of peer reviews is getting multiple people thinking about, and working on, the same problem – with a focus on identifying defects that can be prevented before the next phase of development.

Code review is perhaps the most important peer review that an organization can perform. As with other peer reviews, the formality can vary. On the formal end of the spectrum is code inspection. This

typically involves a group getting together in a meeting room to discuss the code line by line. Similar in the level of detail, but much less formal, is pair programming. Pair programming is a technique that involves two individuals sitting side by side with one keyboard and monitor. This is essentially real time review as two people are thinking about the code simultaneously.

Informal code reviews facilitated through e-mail can be very effective. The developer of the code sends an e-mail containing the locations and the details of the code changes to a small group of individuals. The individuals, who typically include other developers and testers familiar with the area of code changed, review the changes made by the developer. The reviewers provide feedback in e-mail to the entire code review group. The authoring developer responds to all of the feedback through e-mail. The response for each issue falls into one of the following categories: a) fixed, b) do not want to fix, or c) would like to fix, but this is not the appropriate time. A bug will be written up to address the issue in item c) in the future.

A side effect of any of the peer review approaches is the shared learning that occurs between team members. Junior team members learn from experienced team members. New ideas are brought to the table by anyone who participates. The overall knowledge of the team goes up through these reviews, enabling better discussions over time.

Test Approaches

While focusing on preventing or removing defects early in the development cycle is a critical activity, it is clearly not enough. There are many ways for developers and testers to provide feedback on the application after coding begins. This section will provide an overview of these approaches.

Unit Testing

The practice of unit testing - "a procedure to validate that individual units of source code are working properly" - (Wikipedia) has grown significantly in the past few years. This is a bottom up testing approach where automated tests are written by the developer. In Test Driven Development, the test is written before the production code which will provide the new or changed functionality.

A number of publicly available frameworks that enable unit testing have been written for various languages. An example is nUnit, written for testing applications developed in .NET. nUnit is available from SourceForge.NET or at http://www.nunit.org/. Visual Studio Team System (VSTS) also has capabilities for implementing unit tests for .NET languages.

Because of the popularity of unit testing and the need for a framework supporting the X++ language that is used by AX, the SysTest unit testing framework was developed and shipped as part of Microsoft Dynamics AX 4.0. For more information on SysTest, see http://msdn2.microsoft.com/en-us/library/aa874515.aspx.

The justification for developing unit tests in parallel with product code can be challenging. One argument against unit testing is that it takes highly paid developers away from writing the production code. While this is a reasonable argument when taking a short term view, the benefits of writing unit tests are significant. These benefits include finding bugs earlier, providing a safety net of tests for making changes later, and improving design. In the long run, unit testing will improve customer satisfaction and developer productivity.

Development of SysTest based unit tests in AX can be an important quality practice for ISVs and implementation partners. Combined with the X++ Code Coverage capability in the application, developers and testers are able to make informed decisions on the gaps in their testing.

Functional Testing

Functional testing, as used in this paper, is any testing that is focused on verifying the functionality of the product from the user's perspective. In contrast, unit testing is focused on verifying the design intent of the code from the developer's perspective. The scope of functional tests could vary from a particular feature in the system to end-to-end scenarios that involve multiple modules in the product.

While testing at different levels (unit, feature, module, and system) is a good idea, building scenario focused tests is a good approach in addition to unit tests. Scenario testing is "a software testing activity that uses scenarios based on a hypothetical story to help a person think through a complex problem or system" (Wikipedia). Personas are also frequently used with scenarios tests. Personas are defined as "fictitious characters that are created to represent the different user types within a targeted demographic that might use a site or product" (Wikipedia)

ISVs frequently use business domain experts for their functional testing. This is a good strategy because these individuals understand the user needs and can well represent the user while testing. However, this staffing strategy needs to be coupled with good guidance for the management of functional testing activities. While business domain experts know user needs, they do not necessarily understand the goals of software testing.

The Visual Studio Team System group has recognized this need and has made test case management and support for the manual tester a key tenet of the upcoming 'Rosario' release of the product. Rosario is available as a Community Technology Preview (CTP) at the time of writing. For more information on the product, see MSDN Overview of Rosario.

Until Rosario becomes available, and for ISVs that choose not to purchase VSTS, Microsoft has some tools and guidance available now:

Tool	Description
Test Case Management in Excel®	One challenge of coordinating a testing effort is identifying what tests can be run, should be run, and have been run. To help organize test cases, an Excel template has been created. Test cases managed in this Excel template should import into Rosario with minimal rework when that tool becomes available. This template is available in the same MSDN download as this paper.
Task Recorder	Available directly in the product for Microsoft Dynamics™ AX 2009, the Task Recorder is a tool that can log and create documentation and workflows of user activities in the application. The tool can output this transcript of activity in several Microsoft® Office system formats. The formats include training-oriented documentation in Office Word, presentation content in Office PowerPoint®, and process-oriented documentation in Office Visio®. Task Recorder can be used as a potent tool to help manual testing. Since it records screenshots and user actions into Word, a tester creating a test script could easily augment the document with expected results and other notes to fully document the script. A sample test script created using Task Recorder is available in the same MSDN download as this paper.
Microsoft Dynamics AX Demo Data	A challenging part of functional testing is setting up the system with base data to enable testing in various parts of the application. Microsoft Dynamics AX 2009 Demo Data is a good starting point for testing efforts. The links to Demo Data on PartnerSource and CustomerSource can be found in the References section at the end of this document.
Microsoft Dynamics AX Demonstration Scripts	In addition to Demo Data, Microsoft Dynamics AX2009 provides some Demonstration Scripts. While targeted for demonstrating the product in pre-sales activities, these scripts are also useful as a starting point for end-to-end test script for use as a regression tool and for creating your own test scripts for Dynamics AX. These test scripts use the Demo Data and many also use the Task Recorder. The link to the Demonstration Scripts on PartnerSource can be found in the References section at the end of the document.

In addition to creating scripted tests for functional testing, business domain experts are an excellent choice for performing exploratory testing. An individual performing exploratory testing will use the application with a particular goal targeted. Typically time-boxed into sessions, the tester will explore the application without a fixed plan. This approach enables the tester to use their knowledge of the targeted goal and the application to quickly identify issues in the application.

Functional testing is the core of what a software tester does to provide feedback on the product and help ensure a quality release. Supporting business domain experts with tools to manage their work will go a long way towards a successful testing effort.

Other Testing Approaches

Non-functional requirements are "requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors" (Wikipedia). This should be contrasted with functional requirements that specify behavior or functions. Non-functional requirements are often called qualities of a system.

Performance is one example of a non-functional requirement. There are several other important examples that include security, reliability, supportability, accessibility, usability and others.

For Performance testing, the best tool available is the load testing functionality within VSTS today. For more information about load test authoring and debugging, see the VSTS Software Tester Team Center. Within a short time, a tester will be able to use Web test to automate an Enterprise Portal feature to be tested, run the Web test in the load test framework and have results on performance. A tester can also use the Microsoft Dynamics AX Business Connector or Web Service to create a unit test in the VSTS framework and then also run these tests within load test.

Testing non-functional requirements often takes specialized skills. For example, knowledge in running user experience labs is required to objectively test usability. Security Engineering has become a field in its own right today.

Additional guidance on non-functional testing is outside the scope of this document.

To Automate or Not to Automate?

A typical request from ISVs, partners, and customers is to provide a simple record and playback tool that non-technical testers can use to create automated regression tests.

On the surface, this sounds very appealing. Using a business domain expert to create automated tests that can be repeated hundreds of times would be a big saving. However, the overall experience of the software industry with this approach is not very positive. These tools have proven to be very unstable. They produce lots of code (often with much of it duplicated in many scripts) which is very challenging to maintain. The maintenance effort cannot keep up with the product changes and soon the maintenance cost outweighs the benefit of rapid test creation.

Companies that successfully automate regression suites that use UI tools do so with a strong technical presence on their team. For example, the title for the 'test engineer' at Microsoft, Software Design Engineer in Test, acknowledges that individuals in testing must have very strong development skills. It is not uncommon for test engineers to end up writing more test code than the code that is contained in the product.

Once again, the VSTS Rosario release is targeting tools to help the manual tester. Currently called Automation for Navigation, the feature "helps the tester to automate the repetitive and time consuming parts of the manual test case so that more focus can be laid on testing the functionality of the application, thus increasing efficiency" (taken from http://blogs.msdn.com/vstsqualitytools/archive/2007/12/05/manual-test-runner.aspx, 2nd paragraph).

The best automation tool for the Microsoft Dynamics AX rich client today is the SysTest framework. Besides using it for unit testing, as described earlier, it can also be used for functional testing. Much of

the core business logic can be tested by using this framework providing the best practice of not placing business logic on the forms is followed.

For automation of test scenarios that are focused within Enterprise Portal, the best option is the web test functionality within VSTS today. Refer to the VSTS Software Tester Team Center for information about web test authoring and debugging. Consideration needs to be given to making tests robust to run in different test environments, and to providing in-depth data validation. Future guidance will offer examples of these types of tests.

For testing Microsoft Dynamics AX Web Services developed in the Application Integration Framework, a tester should first consider a set of unit tests using the SysTest framework. For end user scenario testing, a tester can use any readily available test tool, such as VSTS or nUnit. Automating the setup of the AX Services can be a complex task and is best performed as a manual setup that you do on the test environment before running the automated test.

Conclusion

Testing of ERP applications and customizations of those applications is a difficult problem.

The recommendation today is to focus automation efforts on unit testing and to provide business domain experts with the tools for test case management along with documentation. All of which helps them test effectively and efficiently. Look to use VSTS Rosario when it becomes broadly available for additional automation efforts.

Software Testing Resources

The Software Test discipline is evolving and the resources available for software testers are becoming more prevalent. While this paper is a cursory overview for testing Microsoft Dynamics AX 2009, a starting point for some good web sites and books is as follows:

- MSDN Tester Center – This web site is the starting point for guidance and tools from Microsoft. From the home page; *"The Microsoft Tester Center showcases the test discipline as an integral part of the application lifecycle, describes test roles and responsibilities, and promotes the test investments required to deliver high-quality software."*
- Software Tester Team Center – This website covers multiple topics all related to the testing discipline and to using Visual Studio Team System.
- Software testing at Wikipedia – This gives an overview of the history and challenges of software testing.
- Lessons Learned in Software Testing, by Cem Kaner, James Bach, and Bret Pettichord – From the book description, "*Decades of software testing experience condensed into the most important lessons learned.*"
- Testing Computer Software, 2nd Edition, by Cem Kaner, Jack Falk, and Hung O. Nguyen – From the book description; "*This book will teach you how to test computer software under real-world conditions*."
- A Practioner's Guide to Software Test Design, by Lee Copeland – From the book description; *"Here's a comprehensive, up-to-date and practical introduction to software test design. This invaluable book presents all the important test design techniques in a single place and in a consistent, and easy-to-digest format."*
- xUnit Test Patterns: Refactoring Test Code, by Gerard Meszaros – From the book description; *"Automated testing is a cornerstone of agile development. An effective testing strategy will deliver new functionality more aggressively, accelerate user feedback, and improve quality. However, for many developers, creating effective automated tests is a unique and unfamiliar challenge."*

- Code Complete: A Practical Handbook of Software Construction, by Steve McConnell – From the book description; "*Take a strategic approach to software construction and produce superior products with this fully updated edition of Steve McConnells critically praised and award-winning guide to software development best practices.*"
- Inside Microsoft Dynamics AX 4.0, by Arthur Greef et. Al. 2006. – This includes a chapter on Unit Testing by using SysTest, the test framework that is included in Microsoft Dynamics Ax. This book is also available as a free e-book at http://download.microsoft.com/download/2/5/8/258C8894-B94A-4A87-81EA-4DBB9776F8F2/622579eBook.pdf.
- Unit Test Framework at the Microsoft Dynamics AX Developer Center – Provides an overview of SysTest with links to How To examples.
- Microsoft Dynamics AX 2009 Demo Data – Demo Data for Microsoft Dynamics AX 2009 is available on PartnerSource (https://mbs.microsoft.com/partnersource/support/selfsupport/productreleases/AX2009DemoData.htm) and CustomerSource (https://mbs.microsoft.com/customersource/downloads/servicepacks/AX2009DemoData.htm).
- Microsoft Dynamics AX 2009 Demonstration Scripts – Demonstration Scripts for Microsoft Dynamics AX 2009 are available on PartnerSource (https://mbs.microsoft.com/partnersource/deployment/documentation/howtoarticles/presalesdemokitmdax2009).

Frequently Asked Questions (FAQs)

1. Does Microsoft intend to provide a Record and Playback UI automation tool for Microsoft Dynamics AX2009?

   ***Answer**:* Microsoft Dynamics AX2009 provides a Recording tool named Task Recorder for recording functional scenarios and outputting the actions to Microsoft Office tools. However, currently in AX2009, there is no Record & Playback UI automation tool. Microsoft's next Visual Studio release, code named 'Rosario', has support for Record and Playback UI tool which is targeted to help manual testers. Details are available at MSDN Overview of Rosario.

2. Are there commercially available tools from companies other than Microsoft available for automating Microsoft Dynamics AX 2009?

   Answer: There are no commercially available third party tools that Microsoft Dynamics AX 2009 supports at this time.

3. Are there improvements planned in the next major release after Microsoft Dynamics AX 2009 that will improve testability of the product?

   Answer: Microsoft Dynamics AX 2009 provides an out of the box unit test framework tool (the SysTest unit test framework) for automating unit tests for X++ classes. The Microsoft Dynamics AX 2009 team plans for continual improvements in the testability of the product in coming releases. At this time, it is too early to say what exact types of improvements there will be, but the plan is to communicate any future updates through CTPs and/or PartnerSource.

4. What's the best way to get started using the SysTest unit test framework?

   Answer: There are number of ways to get started with SysTest, including the MSDN site, http://msdn2.microsoft.com/en-us/library/aa874515.aspx, and the Inside Microsoft Dynamics AX 4.0, Arthur Greef et. al. book which has a chapter on Unit Testing using SysTest.

5. Is there any documentation with real world examples for using SysTest unit test framework?

   Answer: The Microsoft Dynamics AX 2009 team plans to provide more guidance around real world examples for unit tests by using SysTest shortly after the release of Microsoft Dynamics AX 2009. Please see the Microsoft Dynamics Developer Center for future updates.

6. Does Microsoft intend to provide any of its internal tests for usage for Microsoft Dynamics AX 2009 development done at ISVs, partners, or customers?

   Answer: At this point, there is no plan to provide any of the internal tests because of tighter dependency on internal tools, but this may change in the future releases.

7. Does Microsoft intend to provide any tools for Source Code Configuration management for AX development at ISVs, partners or customers?

   Answer: Microsoft Dynamics AX 2009 has a built-in source code configuration management tool which is part of the MorphX® development environment. Microsoft Dynamics AX 2009 extends this functionality with integration to Visual Studio Team Foundation Server(TFS) version control system.

8. Does Microsoft intend to provide a test automation tool for Performance, Load, and Stress testing for Microsoft Dynamics AX 2009?

   Answer: Microsoft Dynamics AX 2009 team uses Web Test and Load Test capabilities of VSTS tool to conduct Performance, Load and Stress testing. VSTS Web Test and Load Test are available products from Microsoft. Details can be found at the following web site: http://msdn.microsoft.com/en-us/teamsystem/aa718823.aspx.

9. Is there any documentation with real world examples for using VSTS for testing Enterprise Portal or Web Services?

   Answer: The Microsoft Dynamics AX 2009 team plans to provide more guidance around real world examples for using VSTS for testing web applications and web services. Please see the Microsoft Dynamics Developer Center for future updates.

Microsoft Dynamics is a line of integrated, adaptable business management solutions that enables you and your people to make business decisions with greater confidence. Microsoft Dynamics works like and with familiar Microsoft software, automating and streamlining financial, customer relationship and supply chain processes in a way that helps you drive business success.

U.S. and Canada Toll Free 1-888-477-7989

Worldwide +1-701-281-6500

www.microsoft.com/dynamics