Implementing a Standard Software Testing Process at MSU

MSU's University Systems teams develop and support software and services for MSU campus users (faculty, staff and students). Currently there is no standard testing methodology in use by these teams [including the one I work for], which hinders the ability to consistently produce high quality deliverables. My topic of focus this semester is to produce a detailed recommendation to my Director, seeking to implement a high level testing approach that these University Systems teams will follow before a product or service is released.

Even before beginning work on a standard testing approach for MSU, I was aware that this would be a complex task for a number of reasons:

- MSU teams and individuals have not collectively worked under departmental-wide standard testing processes before; many have not operated under any form of structured approach (e.g., one that is based on and follows industry-recognized procedures and guidelines).

- Users have differing experiences related to testing; from those with no testing background or experiences, to those with broad and extensive knowledge of project-based, structured testing.

- Testing by its very nature has wide-ranging testing concepts and vocabularies and the understanding of these also vary widely between MSU IT teams and individuals.

- Reluctance for individuals and teams to the process of "change," which is the degree and ability to adopt new (and usually fundamentally different) practices due to the modification or elimination of long established processes. While there has always been an MSU goal of creating high quality, useful software and services, there has never been a concerted effort to follow consistent, similar processes or reviews across development teams. Teams currently have their own individual testing processes and reasoning for following them; although most do so because "that's the way they've always done it."

- Ongoing training sessions and reference materials on testing are needed to assist and direct both the management and development/individual sides through any new testing process until it becomes well known and followed.

These barriers were initially perceived when I first came to MSU in late 2009 to act as the Testing Coordinator for the Enterprise Business Systems (EBS) Project, which was automating HR/Payroll, Finance, Business Intelligence and Data Warehousing processes. While the teams did have a focus on developing and testing useful solutions for their end users (MSU business employees), they seemed even more concerned with being the "best" MSU team in terms of getting their part of the project done the quickest, and having the least amount of software defects. As I've moved around to different development areas, the overriding goal always seems to be to have a great reputation with other areas of campus. But there is a better opportunity to be seen in a positive way by creating and delivering high quality software and services; and a standard approach to testing can provide immediate, noticeable results.

As I began work on a testing proposal I examined the main processes and common, multiple Software Development Lifecycle (SDLC) patterns that are currently in use by our teams. I found that there was no shared common pattern across our departmental teams in regards to either a definition or adherence to any SDLC model. SDLC methodology outlines repeatable steps that all development teams will follow; software testing is a process that occurs late in the SDLC, and its direction, goals, and methodology are based on the selection and continued use/adherence to a SDLC method. Because the development methodology drives the types and amount of testing that are needed, it did not make sense for me to continue with a software testing approach until an SDLC was established. So I asked to be placed on a small committee to draft something up.

I began this task by questioning developers to identify their understanding of SDLC methods, and found that the experiences of these teams range from little experience, to those that are familiar with most SDLC methods. I ended up creating an initial, hybrid SDLC model that has a linear pattern, with spiral iterations to it. The spiral patterns occur only between a few key [and adjacent] steps in the SDLC, and act as a check and balance process to ensure the prior process(es) have been accurately and fully completed before the process proceeds to the next steps.

There are also several milestones where the progress will be reviewed and signed off on, before it can proceed to the next step in the model/pattern. My SDLC proposal has been handed off to the two other leads (security and development) to polish off and present. The plan is to preview it to all development team leads to explain it and get a final approval/buy-in from all. With testing being a major step in the SDLC process, my work should also get me recognized as a contributor and knowledgeable resource, which aids both my credibility, and abilities to get a standard testing process established.

After considering many potential ideas related to proposing a business testing strategy [e.g., Develop, Test, Strategy, Model, Method, Hybrid, Document, Tester, Plan, Standard, Process], I selected "Vision" as my abstracting idea to concentrate on. This was for a few of reasons:

1. "Vision" is a word that is critical to the understanding and ultimate approval of my proposal by both the management team, and my fellow co-workers.

2. A clearly understood, shared vision promotes a higher level of buy-in by all affected parties, and improves the chance of implementing a successful process.

3. It allows for multiple forms of representation (analogies) for me to consider and select, to deliver a common message, especially as I approach the Training aspect of informing the teams and providing them with the information that they will need to successfully set up and follow a standard testing approach.

Today our departmental teams strive to create effective, high quality products and services; to various levels of success. There are no standard processes that they follow, no required "passing" conditions or criteria to check for, and spotty reviews and approvals along the way. In my attempt to provide a creative, encompassing solution, it is critical to first get everyone to have a shared vision on how an SDLC and testing solution will work, and how it benefits everyone: management, the development side and the end users. Included is an examination of the potential risks that may be encountered, and how my proposals work to minimize or eliminate them.

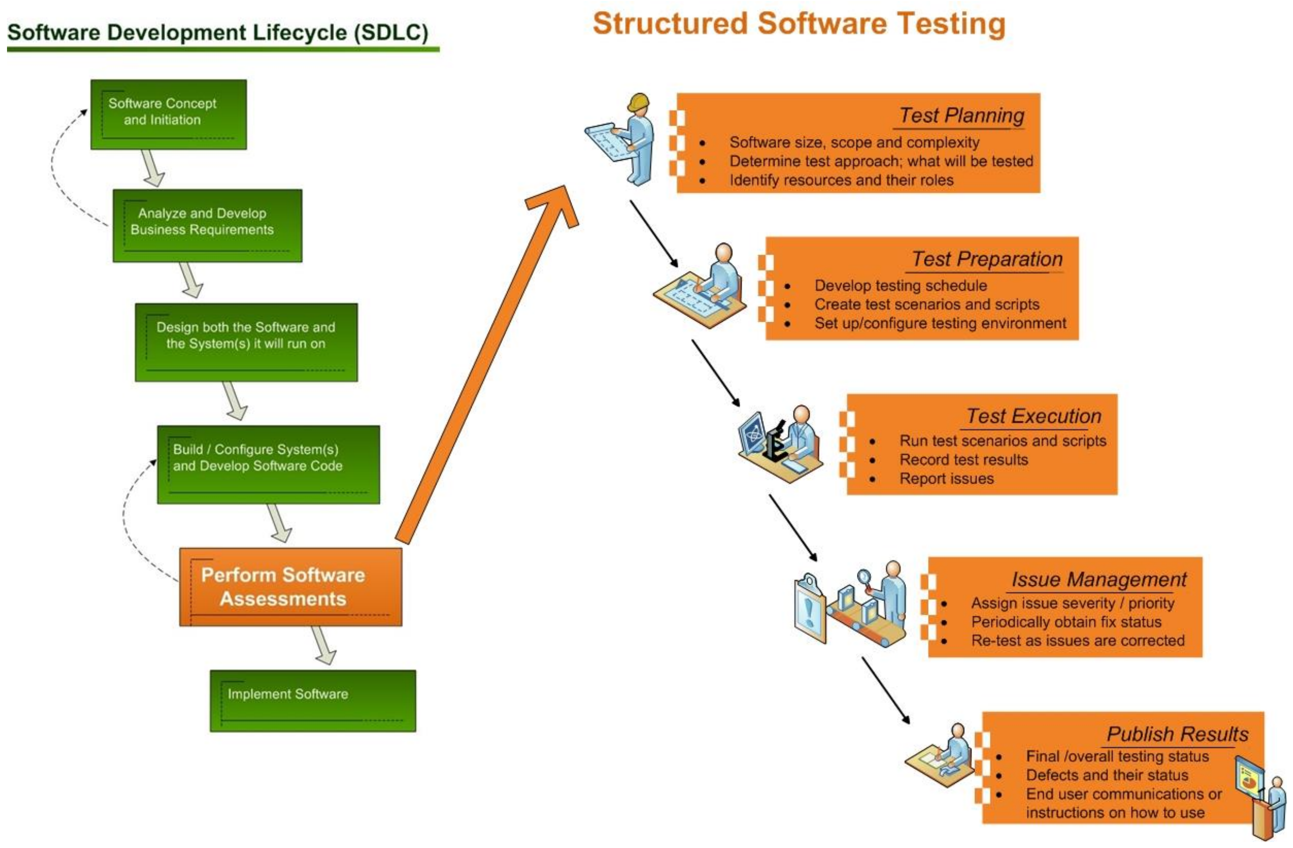
This "vision" encourages involvement from all teams to share their best practices and evaluate proposed ideas and processes that benefit the most from reviews and approvals. My testing approach then provides a standard blueprint with directions for achieving a widely-agreed upon solution, while allowing team flexibility on specific steps that are taken. It provides templates and checklists they can utilize, and requires the long term retention of documentation that aids with future development and auditing purposes, by maintaining a history of what has been done.

With twenty years of testing experiences I regularly empathize with testers on many of their ideals and concerns, as I have generally experienced similar situations and resulting emotions from testing software. I considered these experiences as I developed both the SDLC model and software testing approach.

My goal is to introduce a testing approach that provides MSU teams with both scale and dimensional mapping / thinking ideals, so testing can be tied in with their more well-known software knowledge and experiences. A Top Down flowchart was developed with the top five major testing processes in the order they need to be performed.

Graphic clip art has been included with each process that visually represents and supports it. For additional reference examples of sub task that are commonly performed as part of each process were provided. While these sub tasks could also be further scaled into more specific, highly-defined instructions and directions, I did not wish to go to that level of detail. I didn't want it to appear that we were forcing a direction with little input from the teams, or the inability for them to have flexibility in how they carry out their specific development and testing tasks. This approach will be more successful by providing a high level blueprint with a minimum number of requirements, rather than a perceived inflexible set of standards to follow.

In a similar way that the "Powers of Ten" video by Charles & Ray Eames provide visual scaling, I wanted to tie Software Testing into the larger six-step Software/System Development Lifecycle (SDLC) model; so viewers can understand where in the SDLC process that software assessment occurs. I've listed it in the same Top Down format and clearly shows that testing (and other assessments) occur near the end of software development (see diagram below). Upon close inspection there are some similar steps between the SDLC and Testing where the process and order are identical, but the scope of what is being covered is different. It provides symmetry and relationship between the two, which could help individuals to more deeply understand the process by tying the two together.


Software Development Lifecycle (SDLC)
Software Concept and Initiation
Analyze and Develop Business Requirements
Design both the Software and the System(s) it will run on
Build / Configure System(s) and Develop Software Code
Perform Software Assessments
Implement Software
Structured Software Testing
Test Planning
Software size, scope and complexity
Determine test approach; what will be tested
Identify resources and their roles
Test Preparation
Develop testing schedule
Create test scenarios and scripts
Set up/configure testing environment
Test Execution
Run test scenarios and scripts
Record test results
Report issues
Issue Management
Assign issue severity / priority
Periodically obtain fix status
Re-test as issues are corrected
Publish Results
Final /overall testing status
Defects and their status
End user communications or instructions on how to use

Long term interactive information will be generated off of this model; likely in the form of a website with clickable links that provide more in-depth descriptions and supporting examples of software testing steps, along with document templates that demonstrate what type of information is collected, created and displayed. In this manner there will exist a quick model for those with more knowledge on software testing, and clickable information for those that are new or learning about testing. I also have some other graphic and interactive models from previous MAET classes that I can use as supporting tools, including a robust library of testing definitions to provide standard terms and address gaps in team's vocabulary.

Once a standard testing approach is in place, there will be a need to introduce the new concepts and processes to the development teams, and I created a collaborative group project that could be performed as part of a training class. This modeling effort requires observation and deduction skills common to software testers, and provides reinforcing concepts and visions that are an important part of everyone's knowledge on our development teams.

The implemented SDLC model and Testing Approach should be reviewed periodically for ways to improve on each, using the results from each team's development efforts, as well as all involved stakeholder feedback (management, developers, testers, end users of the software).

In summary, successfully assessing and solving learning and development issues can be accomplished through the use of the creative methods that were learned in this class. With my topic I had to initially observe and identify a missing component [implementing a standard testing approach] that should lead to better quality products from the MSU teams I work for (and with). I met with several developers and spoke to a couple of managers to get their perception of the current software development situation, and to obtain initial feedback on how introducing some guidelines around testing would be received by all. Because no department-wide processes exist, I planned to implement a process that followed a pattern with minimal steps to it, and that allowed the teams to retain a majority of their existing processes and decision-making tasks. I directly empathized with the testing teams, but widened this to also consider best practices and easily-repeated procedures that would meet the University's management needs, and didn't stray too far from existing practices. Where there are information gaps, I look to answer them with supporting details on the benefits that this change will bring throughout the University; from managers and developers to the testers and (especially) to the customers we service. Supporting, interactive information models have been created to get this process up and moving towards its goal; and there are initial training games to identify and drive home the focus of standardized testing, and how it is an important component to delivering high quality software and products.

Sir Ken Robinson noted that creativity was "…original ideas that have value." While my ideas for an SDLC and Testing Approach are not generally original onto themselves, they are new to MSU development teams. Throughout my process I have modified many of the features, and my standard approach is truly a hybrid collection of processes that I have examined from many levels and perspectives, selecting the options that will be the best received and capable of reproducing time and again going forward by all teams. It is through these creative “twists” that I strive to bring heightened value and quality not only to this endeavor, but to all tasks that I work with going forward.

Brian Jenks

CEP 818

December, 2013