

Requirements and Bridging the Silos

By: Andrew Hayward

Part 2 of a 3-Part Series

Introduction

The first part of this three part series on *Requirements and Bridging the Silos* described how, in larger development efforts, the division of resources into roles often reduces the effectiveness of communication and hinders understanding of requirements. Collaborative analysis and design methods, commonly referred to as JAD sessions, and the use of a requirements management tool that is accessible to analysts, designers, developers and customers, were recommended remedies to these communication barriers.

This second installment will continue with two other common silos, testing and implementation, and will describe how to increase understanding of project requirements within and between these groups and the benefits of improving communication with these silos.

Testing

Quality assurance teams are responsible for testing at several different levels. They may be involved in requirements testing, unit testing, system testing, quality attribute testing (such as performance or usability), integration testing, user acceptance testing and more. The question quality assurance managers are often faced with is: 'what can be tested in the time available?' In complex systems with several different functions for different user groups, it is not realistic or feasible to test all of the different possible scenarios in the time allotted for a given project. Throwing more cooks in the kitchen is often not an answer even though it is the one that first comes to mind. Quality assurance managers need to fully understand the business requirements to be able to prioritize the areas and scenarios that pose the highest risk and value to the project and the customer.

Open communication between the customer, quality assurance and the development team is essential in these priority decisions. As mentioned in the first installment in this series, no requirement specification is complete. What is needed is a high level understanding of the business requirements or objectives (such as 'reduce check-in times by 25 percent') to make prioritization decisions on which areas to focus testing efforts. Further, the managers need to ensure that their understanding of these priorities is correct with respect to the current business climate. Involving the customer in these decisions aids in this process and also builds trust and ownership in the end product. The customer needs to understand what decisions were made regarding testing priority and why they were made in the same way they need to understand how their requirements were communicated and interpreted by the development group.

Once the decisions are made and agreed upon, traceability from tests and test scripts that result from those decisions to the business requirements they attempt to validate comes into play. If a requirement changes, the people developing the tests that validate the requirement need to understand what scripts they've developed that may now need to be adjusted. It is often the case that a single test script may verify many different requirements at different levels of abstraction and likewise a single requirement may be verified by multiple test scripts. Managing

these complex trace relationships and interdependencies quickly becomes very difficult if you are relying on spreadsheets or documents.

Requirements management tools will help here but be sure to look beyond the sizzle of demonstrations and find a solution that addresses the problem in a practical, useable way. Traceability matrices are an example of sizzle without real utility. They look good in a demonstration, but any project of significant size quickly renders these large grids unusable. Automated test script generators, which take use cases and generate test cases, are also great in demonstrations. However, they still rely on the quality of the use cases or scenarios entered as well as the test resources' understanding of them. If quality assurance people were not involved in the analysis and design work, they will have no basis on which to prioritize the testing.

Again, once the artifacts (test cases, requirements, designs and specifications) are related via these trace relationships, the tool employed should also be able to aid in communication and visibility across the silos of the organization. Resources in quality assurance should be aware of requirements and designs which have changed, and be able to adjust and react accordingly.

Implementation, Operation and Maintenance

Silos, and the divisions between them, become especially noticeable when it comes to the operation and maintenance of the system. In my experience, there is rarely the interest or staff available to maintain requirements, test and design documentation once a system has gone to production. The people who know the documentation are moved to other projects, and the people who maintain and operate the system, such as support staff, don't have the time, knowledge of the documentation, the tools or the expertise to continue to manage those artifacts. This results in the knowledge of how the system was designed, and why it was designed that way, being lost as it is not sufficiently transferred to the maintenance team.

This is a tremendous loss of value to the organization for many reasons. First of all, impact analysis of changes to the system become increasingly challenging. As time passes the system will gradually change to a point where the original documentation is outdated and can not be relied on as a representation of the system and its behaviour. All of the system knowledge will be fragmented across the people who originally designed the system, use the system and maintain the system. The natural evolution of an organization, accounting for staff changes for example, and the simple passing of time contributes to large pieces of this understanding being lost. Further complicating this is the fact that change continues to happen. When change requests are made against a system, whether they represent bug fixes or changes to the original business requirements and hence the functionality of the system, they are often captured in disparate systems from those original requirements and development activities. This makes the determination of all of the potential impacts of the changes nearly impossible. As a result, a single apparently simple change can lead to unexpected behaviours within other areas of the system thereby propagating more change requests.

The second opportunity that is lost due to the silo effect between operations and maintenance and the rest of the organization is the ability to determine where a problem with the system originated. Without traceability between the implementation artifacts and operation artifacts, and without accurate and up-to-date documentation it becomes very difficult to determine whether a problem is "as designed," meaning how the requirements specified the behaviour, or with how the requirement was satisfied, meaning how the code was written. Being able to gather statistics

on where problems most often originate is a very powerful input for evaluating and improving upon the system development processes.

So again, with the operations and maintenance group we have several challenges to overcome. How do we address knowledge transfer, traceability and impact verification?

The first step to addressing these problems is to provide appropriate levels of visibility and communication of the requirements for a system to the staff that provide system maintenance and support. Too often support people are expected to handle service requests for systems without understanding or context on what the end users need from the system and how it is supposed to operate. Ensure that any training for support staff includes an overview of the value the system provides for the business and users and an understanding of how users work/interact with the system not only today but when the system was originally designed.

Second, break down the barriers between the groups and set up a process where representatives for the customer, IT business analysts, development teams, operations and maintenance and quality assurance review all change requests and service requests, and record them against the specific item or items they affect. In most organizations, change requests and services requests are only recorded against systems or documents as a whole. If it is a change to a requirement, it should be recorded as a change request for the requirement. If it is a change request as a result of an error in the code, meaning the requirement was right, the design was right, but the execution failed, the change request should be recorded against the piece or pieces of code.

The third way to solve these problems is to maintain traceability -- more specifically -- application lifecycle traceability. Leverage traceability diagrams, trace hierarchies or preferably CASE tool functionality when conducting impact analysis, so that when a change to a requirement is made the relationships previously set up can be traced to find all of the potential effects of that change. When a change to a piece of code is requested, the relationships can be traced back to the requirements to see the other design and development elements that are connected.

The challenge is that often the systems used to manage change requests, service requests, design, requirements, source code, testing, and deployment are also in their respective silos, with limited or no integration. Systems that store all of these artifacts within a single repository are ideal in addressing these challenges, with capabilities that act on artifacts within that repository. Using a single data model across the organization is the next evolution, but at the very least a system that integrates knowledge of these artifacts is necessary to even begin to address the problems.

Concluding Remarks

This installment of our three part series on *Requirements and Bridging the Silos* continued to present ways to improve communication and understanding across the development groups. Involving representatives from the testing group earlier in the development effort better enables the quality assurance team to prioritize their work according to what is the most crucial functionality to the customer.

In my experience, a large amount of valuable information is often lost once a system has been released to end users. Change and service requests are often recorded in separate repositories from the original system development documentation and source code. Traceability across all of

these silos will provide valuable information on where development processes need improvement, as well as inform resources in the development effort when their work needs to be reevaluated.

The third and final installment of this series will describe some examples of how silos effect communication and what remedies were applied to solve the problem. Finally, the third installment will provide a list of eight best practices for improving communication of requirements within and across silos.

Andrew Hayward has five years of experience working as an independent consultant, gathering and managing requirements and providing expertise in Requirements Management best practices for organizations across North America and Europe. Currently, as a National Application Engineer with MKS Inc., he works primarily in implementing MKS Requirement Management processes and solutions.