



Audits... the Overlooked Dimension of Software Evaluation

by Jerry E. Durant, Managing Director

What does the word “audit” conjure up in your mind? Fear, reactive, issue raising, too late... I’m sure that each of you can come up with other superlatives that aren’t even fit for print. Whatever the case, audit is the forgotten dimension and lies in the shadows of it’s more prominent siblings, software testing and software reviews. Those that have been immersed in this discipline have seen not only a thousand different ways to test and review but may have even tried some of the methods. What is interesting is that seldom is software audit mentioned, and if so it gets discounted as something done after the fact. It wasn’t all that long ago, and to some extent it is still practiced today, that software testing and software reviews were considered in much the same light.

In order to have a complete and comprehensive assessment of a software project you must consider using all three evaluation methods (testing, reviews and audits). Opting for one method over the other will allow for the potential of latent issues to go undetected. What one doesn’t find, another may. Even though the functionality of computers and languages is quite pragmatic, the implementation of these technological elements is left to a much more fluid application by humans. We seldom are so rigid as to remove those brief moments of creativity which opens the door to defects. Even if we were, the world around us is a melting pot of change and as a result the shifting of direction introduce defect risk potential.

The software audit process is not something that we impose at the end of a project but is a part of the development process. Anyone who has read articles or attended classes on systematic testing or structured reviews know that the activity is a parallel process that runs along side the project. It makes no difference where you are using a very simple development process that is unencumbered by tools or whether you are using a very structured processed steeped in technology, parallelism is essential. Without parallelism, opportunity for change is lost, cost to correct escalates, and playing catch-up is an impossibility. If you don’t do it now, you won’t be able to do it later. Although this statement is partially true (eg. Once you have a problem you must fix it) late stage repair and modifications have long been known to be the casual point in system degeneration.

It may take a bit of flexibility for you to start thinking of the audit software process as something that is done in parallel with the evolution of a system. To make this happen we need to understand the concept of audit objectives. Although there are multiple forms of projects that range from large scale development projects, to rapid application development (RAD), joint application developments (JAD), rapid prototyping, and the most common “the maintenance cycle”, audits a single operational objective. Audits are design to confirm that what should take place has happened. Sounds simple and some of you may already be thinking that all we need to do is to use a simple checklist. However, having something take place is only the beginning, we need to also confirm that it was done to an appropriate level of quality. As an illustration, a software evaluation audit of a requirement specification. The mere existence of the document is not sufficient to constitute acceptability, the audit must also address such elements as completeness, accuracy, ambiguity, and consistency. In order to carry this out the audit must understand the collaborative mechanisms that will work towards a reduction in these areas. This might include the development of test cases, use of



engineering prototypes, structured walkthroughs, and even brainstorming sessions. This illustrates that you need to not only examine work products but also work process. You must also understand, observe, and measure the processes that are in play. In addition, the person(s) carrying out the audit must also understand and appreciate not only the framework by which the application is being subjected to the software engineering process but also the full cadre of approaches to both software engineering and software evaluation. This will require education whether by reading, classroom, research and/or dialog.

Who should do the audit? Probably most would jump at the notion of using the auditors (internal/external or both) and this may very well be the right choice. However, I would also argue that it isn't function driven as much as role driven. By this I mean that it's a task in the project initiative that needs to be filled. The person must be dedicated to the task but be independent of the phase in which they are participating. If there is a stake or there is an involvement, objectivity is severely compromised. To get one started with this I offer the following suggestion;

- Requirement Phase – Software programmers/engineers
- Design Phase – Users
- Construction Phase (coding) – Project Managers, testers
- Implementation Phase – Users

This propose approached contends with to issues; vested interest and technical ability. As an example, user(s) are lest prone to have a technical orientation and as a result they are inadequately equipped to assess code related audits (and code related activities). However those same users are apt vested in seeing to it that the requirements that they have developed (which were converted into technical specifications) are carried out in the design of the system. Likewise they have a significant interest in seeing that the implementation, and any associated issues that might arise are properly dealt with. Some organization have used quality assurance to fulfill the role as auditors. Usually when an organization assigns a permanent assignment to a group to carryout audits it is because they believe that establishing a permanency will insure repeated continuance. Unfortunately, even continuance does not insure that the audits won't fall prey to a lack of quality. Therefore to counter this risk periodic post mortem project assessments will shed light on the overall evaluation process. We sometimes forget that in order for any process to exist it must be effective and cost efficient. Today's business cannot afford to perform activities simply for the sake of carrying them out. Over the last twenty years we have seen repeated examples of world class evaluation enterprises that have been disbanded because they failed to deliver results or to do so in an economic fashion. If it cost \$20 to save \$1, is that good business? If you spent \$20 but found no problems, and later we found 1 problem that cost \$1,000,000 to remediate the impact and repair, was the \$20 well spent? Sometimes self-assessment is painful, but it is done in order to contribute to the success of our business. I have said more than once, "our objective should be to work ourselves out of a job by way of change in software engineering culture". Thus far we have come close... but we aren't there yet!

For more information on software evaluation (systematic testing, structured reviews and software audits) contact Certellus LLC at certellusllc@earthlink.net .