**Internal Peer Review Process for Software Artefacts delivered by Scrum Teams in ITX and EST**

Srinivasa Rao Subramanya Rao,

Software Designer/Developer, ITX.

**Abstract**: In this draft paper, we motivate the need for peer reviews to be conducted for all software artefacts delivered by scrum teams in the ITX space and indeed the whole of EST. We argue that internal peer reviews can both improve the quality of software artefacts delivered whilst being cost-effective at the same time.

*Intended Audience:* Software Designers, Developers and Testers, Scrum Masters, Product Owners, Project and Program Managers, Top Management.

**(a) Introduction:**

*Internal Peer Review* is a review of any software artefact finalised by a team member and usually conducted from within the team with an intention of enhancing quality of the artefacts. This is done before the artefact is formally delivered. In many software development companies, this is usually part of the software development life cycle process and is employed, in part, to satisfy the requirements of industry standards such as the SEI-CMM model. Though there is no need for us in the ITX and the broader EST to conform to these standards, defining our own processes and adapting them as and when required will certainly help us in improving the quality and longevity of our software artefacts whilst enhancing our ability to be agile as we argue in this paper.

Within the EST, we do review some of our deliverables externally (and sometimes internally), but we don’t have a well-defined process for internal reviews. This dynamic is, in part, aided by the idea that we may not be able to get the same standard of review comments when the review is done internally as compared to externally. We address this concern in the next section of this paper. Before we do this, we very briefly list the “when” and “what” software artefacts are to be peer reviewed. We then provide a list of reasons for “why” software artefacts should be peer-reviewed. We also dwell upon the important topic of cybersecurity of our systems. Finally, we touch upon the human aspects and psychology of peer reviews briefly.

**(b) When, What and Why of Internal Peer Reviews**

*When do we need a software artefact to be peer reviewed?*

A review may be necessitated when

1. a new software component is constructed
2. an existing software component is modified/changed either due to a new business requirement or a bug fix.

*What are the artefacts that we want reviewed?*

1. Design Document corresponding to the software component being constructed or modified.
2. The new software component (in code)
3. Corresponding Test Cases

*Why do we need the software artefacts to be reviewed?*

1. Design walkthrough or a code walkthrough or a combination of both can help in reducing the number of bugs. (It may not be possible to detect all bugs in the unit testing phase or even higher levels of testing). Design and code walkthroughs will most certainly reduce the cost to the ATO as the number of bugs discovered during Functional Testing and System Testing will be lesser than otherwise. The earlier the bugs are discovered, the lesser the cost of fixing the bugs.

We provide an *abstraction* of a real-world example in our ITX space below using a Design/Code snippet. By *abstraction*, we mean that we only have the bare minimum structure for illustration and all other details not relevant to our discussion have been removed. In this snippet A, B and C are all Boolean.

----------------------------------------------

*If (A and B)*

*Do task-1*

*Elseif (A and B and C)*

*Do task-2*

……..

-------------------------------------------------

In the above snippet, task-2 will never be executed because of the following reason – whenever (A and B and C) is true, (A and B) is also true and so task-1 will be executed whenever we intended to execute task-2. A design/code walkthrough can reveal this and the cost to fix it will probably be a few minutes (easy to do costing us just a few dollars or tens of dollars at most). But once this gets to higher levels of testing, then the cost of debugging and then fixing will be a few hours at the minimum and then the fix must be escalated to higher environments via a new release or via a hotfix (at least a few hundred dollars). Thus, it is cost-effective to fix errors via peer-reviews than fix it later.

Reviews may not eliminate all bugs but can certainly help in reducing the number of bugs.

1. We can quantify and increase the quality of the software artefacts delivered by us in ITX (Quantification can be done using data in the review documents).
2. In addition to correctness, comprehensibility of Design Documents and the corresponding programs are important. The Design Documents and the algorithms in these design documents should be written and structured in such a way that these should be readable by any person whose only pre-requisites are that they have some programming experience (usually well-versed with structured programming primitives such as decisions, loop structures along with an ability to play with simple data structures such as arrays). One of the primary goals of a design document is to reduce the cost of ongoing maintenance of the software system and a prerequisite in achieving this goal is to have design and development that is comprehensible. If comprehension is not a goal and is side-lined, then it is not necessary to have a design document in the first place.
3. Security is of paramount importance in today’s world. Given that we have non-state and state actors in the Cybersecurity adversary space, it is more important than ever to do everything to reduce the risk of application software vulnerabilities. Code walkthroughs can help us in doing so. Since this is a very important topic for all of us at the ATO, we devote a separate section (Section (b)) in this paper to elaborate on this.
4. Internal peer reviews help us in reducing our dependency on external reviews and can help us in doing away with the associated bottlenecks in terms of the turn-around wait time for the external review. When we depend on external reviews alone, there can be a long turn-around wait time for the reviews to be completed as the number of external reviewers are small. Internal peer reviews facilitate agility and in synch with the agile philosophy.
5. Reduce dependencies on one or two “rare” resources who conduct all the reviews and reduce the workload on the Pos doing internal reviews. When we rely on “rare” external resources, it is possible that these resources get stressed and this can result in missing out on crucial errors in the system. This is especially important in an environment where there is extensive dependency on “rare” resources. Reducing this dependency may help us become self-reliant at the ATO and will also reduce cost to the ATO.
6. Facilitate “person-independent” teams (for instance, if a particular team member is away, then somebody else can quickly step in and take the task to completion). We don’t want to be in a scenario where we think that resources indulging in day-to-day design or development can’t indulge in internal reviews of these very artefacts that they develop in the first place.
7. Facilitate “pairwise” programming, designing, and testing.
8. Data from the review documents can be used as a learning and an informing tool for the entire train. It may help us determine, for instance, if there is a need for training in some areas or if there is a change required in the way we do things.
9. Create more cohesion in the scrum teams, enhance participation and increase collective productivity.
10. Make the business areas and clients (both internal and external) happy due to the enhanced quality of our software products.
11. Facilitate uniform standards across divisions/work-areas (for example, sticking to a single Cobol coding standard across the ITX and indeed across the ATO - Naming conventions of data variables in our programs can impact comprehensibility, for instance), then comprehension of software objects becomes easier across Business areas.
12. There are occasions when the Design and Code are not in synch in our space. In these circumstances, it is hard to determine later as to whether Design or Code is the source of *truth*. An internal peer review process can help in the reduction of the number of instances where this occurs.

**(b) Cybersecurity:**

The importance of the security of our applications cannot be overemphasized in a world of ever-increasing connectivity. In fact, one can argue that application security of our tax systems is as important as correct functionality.

Currently we have a “Security Walkthrough” document for our backend services, but this may not be sufficient in the current world. Security should be an integral part of the Design and coding effort with a goal of reducing software vulnerabilities. Peer reviews can help in reducing software vulnerabilities. Not all vulnerabilities can be immediately exploited by adversaries. But if we design and code for security, then we can reduce the chances of the success of the adversaries.

To further motivate this, we provide a detailed context below.

A story of a data-breach that apparently occurred in one of the Federal Govt agencies in America is documented and available publicly at (<https://fedtechmagazine.com/article/2015/06/cobol-and-outdated-technology-cited-factors-opm-hack>)

and

(<https://www.nextgov.com/cybersecurity/2015/06/heated-house-hearing-offers-new-clues-how-hackers-broke-opm-networks/115474/>).

A significant and surprising part of this story is that legacy COBOL was one of the factors that contributed to the data-breach. Though this data-breach story is from 2015 and thus an 8-year-old story, it may be relevant to us in the ITX and the ATO as we use legacy technology including COBOL. In this context, there is a paper by Pang and Tanriverdi with the following title:

*Strategic Roles of IT Modernization and Cloud Migration in Reducing Cybersecurity Risks of Organizations: The Case of U.S. Federal Government*.

This paper is available at <https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2933577>.

 In this paper, the authors hypothesize and write "*Organizations that have a larger stock of legacy systems are likely to experience more security incidents*".

While the actual technical details of the breach due to COBOL in the above story are not available publicly, it is feasible that SQL injection or inadequate data validation may have contributed to the breach. It is thus important to fortify our COBOL services with robust input data validation. This not only reduces the number of possible bugs but also reduces the risks of a security breach. This is especially important as our COBOL programs routinely deal with sensitive input data procured from our DB2 tables that is seldom encrypted (both when the data is at rest in the DB2 tables or when the data flows through various COBOL programs).

Thus, to reduce the risk of software vulnerabilities due to inadequate data validation, it is important to perform appropriate Design and code reviews with security as a focus.

**(c). Human aspect and Psychology of Peer Reviews:**

Sometimes, peer reviews can be viewed negatively and may be interpreted as an exercise in fault finding. Thus, there may a need for top management to provide an assurance that no comments in formal or informal peer reviews will be held against any employee in any circumstances. It should be emphasized that the sole intention of peer reviews is to improve product quality and can be conducted with minimal or zero disruption. This can facilitate a culture of innovation and human egos will not be hurt.

**(d). Cost to ITX and EST:**

Negligible cost. In fact, there will be a reduction in costs, as argued above. So, it’s a win-win situation for all stake holders.

**(e). Further Work:**

After we develop consensus across all scrum teams in the train on the need for internal peer reviews, we can then come up with an answer to “How” to do peer reviews for the software artefacts delivered by us.