Engineer Your Review Process:

Some guidelines for engineering your engineering review processes for maximum efficiency

Tom Gilb
Result Planning Limited
Tom.Gilb@INCOSE.org

Abstract.

You can tailor your various review processes so as to maximize effectiveness for purpose, at minimum costs. I call this review efficiency. This presumes you are willing and able to state a set of clear objectives for your various reviews. Then we can design review strategies to try to meet those objectives. In addition to corporate level review design, you need to design-in the freedom locally, at the project level, and the individual review level, to dynamically adjust or optimize the review process for local conditions and local requirements.

Review Process Tailoring

Review Principles:

1. the right way to tailor a review process depends entirely on
    a. your objectives in doing it
    b. your objectives which can be multiple
        (example: train and detect defects, and measure and motivate)
    c. the economics of your project and your organization
    d. the complimentary set of review sub-processes you have chosen
    e. and many other factors (example your maturity, special circumstances immediate deadline, already released software, non-software reviews)

A Comment on the IEEE Software Review ('Inspection') Standard in Process:

This paper is triggered by questions to me by participants in some IEEE Standards work.

I was not happy with the approach. It was both vague and bureaucratic. The core of this paper was my reply. This paper goes into far more detail than that reply (April 2007, Blog at gilb.com). The IEEE Standard was for software reviews. My concern is *any* kind of review, management, systems engineering, or software. I believe the principles of tailoring apply to all of them.

"The IEEE 1028 standard (Software Inspections) is presently being updated. There is a paragraph describing the responsibilities of a software specification reader as follows:

> *"The reader shall lead the inspection team through the software product in a comprehensive and logical fashion, interpreting sections of the work (for example, generally paraphrasing groups of 1–3 lines), and highlighting important aspects."*

This suggested review process has the following negative characteristics: (Gilb opinion)

1. It prevents to allocation of special defect-searching *roles*. These can maximize the major defect find for the team.

2. It forces *individuals* to go at the same *pace*, and in the same *sequence*. Some individuals can and should go much faster than others, and both slow and fast people working at the optimum rate for them as an individual can produce better major defect-finding for the team. They need to work in different sequences, especially to fulfill complimentary checking roles.

3. The reader, with their oral in group interpretation creates a disturbing noise that may distract and prevent individuals from finding defects.

4. The 'reader function', with interpretation presumes that interpretation is a relevant and effective way of discovering defects. This may be true for certain types of material, but is clearly untrue for many - even most - other types of material. For example in most reviews it is necessary to compare the primary document against many other documents, checklists, rules, source documents, sister documents (example logic compared to detailed test cases) and derived documents (for example user documentation and sales brochures for a product). The narrow point of view that review is 'some kind of check' on a 'fairly sequential logic process' is dangerously narrow, at best a special case.
Even the earlier 1970's software reviews at IBM were used for about nine different classes of documents, source code being only one. How would such a sequential reader interpretation process apply to requirements or architecture or test cases for example?

5. It is an immature idea to *proscribe* such things in a 'standard' without considering the above and more. There are so many exceptions to the 'rule' that people will quickly disrespect the standard!

6. An engineering (did I see IEEE there?) standard can describe potential elements of a large Review process, such as this. But it should not mandate or require them. "

A Mature Alternative Way To Present Software Engineering Process Standards

“Instead a useful engineering standard would (as good engineering handbooks traditionally do) describe the *multiple characteristics* of this sub-process. This would include information about

a. the effect on *learning*

b. the proper (and improper *areas of application* (example which types of specifications).

c. the effect on defect *detection* (% found of those present, aka 'effectiveness'

d. the effect on *efficiency* (defects found per work hour applied by a team)

e. the *costs* of using the method

f. the prerequisite *conditions for success* (how knowledgeable does the reader or checker or reviewer have to be in the domain for example, and many other things)

g. any other factors, effects, side effects, synergy and thrashing (with other sub-processes) expected, or experienced, from the process.

f. useful alternative processes should be named, to achieve some of the effect (for example the learning effect) without the disadvantages or inapplicability for certain spec types.

The inability or unwillingness to deal with all the items in the proposed standard constitutes professional failing in the review field, and such people should surely not be competent to design best practice standards?”

Well this was my reply to my friend in the standards committee. Hopefully it gives the reader a perspective on reviews. Here is a more systematic exploration of elements of engineering your review processes.

Let me start by trying to formulate some basic principles of reviews.

1**. The Variation Principle**: Reviews have varied purposes, and should be designed and tuned to those purposes.

2. **The Efficiency Principle:** The efficiency of a review process is the *effectiveness* in achieving its purposes, in relation to the *costs* of doing so.

3. **The Payoff Principle:** The payback for doing a review is a function of the *losses* that *might* be incurred if the review were *not* held, and consequently could not cause improved action to be taken.

4. **The Many-Purpose Principle**: A given review may itself have multiple simultaneous purposes, and each objective needs to be addressed in designing the review at hand.

5. **The Manifold Principle**: A given review might want to deploy a large number of phases, and a large number of tactics, in order to delivery best efficiency. [SI]

6. **The Prevent – don't Clean Principle**: Reviews should not normally be used to clean up bad work, but they should be used to *measure the work quality delivered*, and to *motivate* professionals to deliver work up to standard.

7. **The Teaching Principle**: Reviews can be an effective instrument for teaching people the current corporate engineering standards, usually more effectively than formal training can teach them.

8. **The Stitch In Time Principle**: The Reviews need to contain a number of built-in devices to allow the review management to stop the process or curtail it, when it clearly will be a waste of time to continue.

9. **The Entry-Exit Principle**: A Review process should be formally governed, like most other engineering processes, by formal process entry and exit conditions – one condition needs to be *defect density tolerance*, to measure and use the numeric defect level of entry and exit products as a condition of stop and go.

10. **The Clean to be Mean Principle**: Specifications need to be clean *before* they can be mean – meaningful to project objectives. You cannot really judge if specs are fit for purpose, if they are not clearly written yet. You probably need a two phase review: clear?, then fit?

Purposes of Reviews

Reviews can serve a large number of different purposes, many of them simultaneously. It is critical that the organization design standard review processes to suit several well-understood sets of purposes. These purposes can be thought of as Review Objectives. They can generally be stated quantitatively, tracked and measured. This will enable us to systematically engineer reviews, to improve their power for purpose, and to maintain their level of efficiency.

Here are some of the known purposes of reviews of specifications:

1. measure quality
2. determine entry to a process
3. determine exit from a process
4. get management to take responsibility for resource approval decisions
5. get a technological group to take responsibility for technical decisions

6. teach spec writers how to follow technical standards
7. motivate spec writers to follow technical standards
8. identify defects for removal
9. to estimate costs of continuing with current plans (delays, rework, maintenance and defects removal costs at later stages)
10. to estimate the remaining major defects, with or without removal of the ones found in the review.
11. to stimulate the creation and contribution of better ideas, in the review or later
12. to measure the effectiveness of processes
13. to measure the productivity and quality or organizations, teams and individuals: people.
14. to reduce defect content by removal of defects identified
15. to reduce the costs of testing
16. to reduce the time to market
17. to give early feedback, before too much is invested in some work
18. more, but this covers the main ones.

Measures of Review Performance

PRIMARY REVIEW PROCESS MEASURES

Primary review process measures can be used to state, and discuss, primary objectives with the review process. It is a widespread misunderstanding that the number of defects found in a review is a *primary* measure. It is *not*. 'Defects found' could simply reflect a ridiculous quantity of defects *available*, because of a failed engineering process or 'bad' engineer. Primary measures reflect what we primarily are *really* trying to achieve with review processes: which is to motivate a stable long-term low level of defects injected, and consequent acceptably low level of defects actually in the specification before, during and after the review. More simply, high quality work.

Defect Density:

Gist: how good or bad the specification is in relation to standards, at a given point on the development process.

Scale: the Average Major Defects per Logical Page remaining at a defined Stage

Major: a defect that can potentially cause economic or quality damage, and must be corrected if known.

Defect: a deviation from an official, written, valid standard (Rule) of writing the specification.

Logical Page: 300 non-commentary words of specification text.

Usage: Primary use is as a primary process entry or process exit condition. It can also be used to analyze process change

Defect Injection Rate:

Gist: the degree to which engineers inject major defects in a process.

Scale: average Major Defects per Logical Page Injected at a defined Stage, by defined Staff, under defined Conditions.

Stage: a development, or maintenance stage where people write or modify engineering specifications.

SECONDARY REVIEW PROCESS MEASURES

Secondary review process measures, are an important means of analyzing, understanding, managing and improving a review process, but they are not themselves the primary reasons for doing the review. They are measures of how good we are at doing reviews.

Rate of Review:

Scale: the planned, actual, individual, optimum or average speed of checking a defined specification (in words, or lines) of a given [Type] using given [Support Documentation] with given [Tools] and given [People Qualifications].

Use: the review rate is a critical determinant of the review effectiveness, for an individual. If the reviewer goes too fast the effectiveness will drop by one, and then two orders of magnitude. Too slow is rare, but has been measured to produce low effectiveness too.

Review Effectiveness:

Gist: how good a review process is a detecting defects that are actually present in the spec.

Scale: % of Major Defects Detected of All Defects Present;

Detected: identified by the review process, and logged for processing.

All Defects Present: all Major Defects that are currently present in a given spec area, and could theoretically be discovered now or later, directly or indirectly by any discovery process including later reviews, testing, and actual stakeholder use.

Use: effectiveness can be used to make sure you are finding the degree of defects you should be experiencing from a review process. It can be used to manage the review process in an organization. It can be used in real time, for single reviews, to accept or reject the review as valid (entry and exit conditions can be used). It can be used to estimate remaining defects. If we find 30%, then 70% remain undetected.

Review Efficiency:

Gist: how cost-effective a review process is.

Scale: The Review Effectiveness in relation to the Review Cost.

Review Cost: all costs associated with designing, maintaining, teaching setting up, managing, operating, auditing, evaluating results of, deciding to do – a review. Review Cost is our best estimate of the organizational cost of conducting a given review, for a given spec area, to a given standard, with given people type and number.

Use: *efficiency* helps us to manage human resources effectively. For example by not putting too many people into a review, with diminishing returns on the investment.

Close Cousin: Process return on investment. ROI in Review processes has been measured in the 10±2 to 1 rate of return [IfM, Raytheon and others].

Individual Learning Rate:

Gist: how fast a person learns to conform to a standard, and avoid injecting defects.

Scale: % reduction in Defect Injection Rate per cycle of learning a defined Specification Standard, by a defined {Individual].

Use: recognition that individuals have a predictable 50% reduction per cycle learning rate for a new spec standard, and they are probably not qualified (are 'learners') until they go through about 5 to 9 personal learning cycles (100 majors/page, 50, 25, 12, 6, 3, 1), and achieve the normal and exit acceptable exit level on first try, each time.

Review Components

Here are some review design components. We need to know, estimate, or measure the effect and the cost of using each one, in combination with a total set of components for one review process, in order to understand if we have an appropriate design for our reviews. Most of these, except digital tools, are found in abundance in Competitive Engineering [CE]

Specification Standards:

Spec Rules**:**

Rules for the specification *itself*

Rules for *inputs-to-process that produced the spec being checked.*

Note: one ossible entry condition to a review, can be a sampling check on the input spec (example the requirements for a design or test plan). If a quick sampling shows a high defect rate, we may fail entry to the review process because we do not have good enough quality input specs to judge the main spec we want to QC.

2.4 Rules: Requirement Specification

Tag: Rules.RS.

Version: October 7, 2004.

Owner: TG.

Status: Draft.

Base: The rules for generic specification, Rules.GS apply. For the different types of requirement use also the relevant rules (that is, Rules.FR, Rules.SR and Rules.SD).

R1: **Stakeholders**: There must be a list of the defined stakeholders and it must span the entire product lifecycle and system space.

For any specific specification, the specific stakeholders can be stated or defined explicitly. For example, Stakeholders: {A, B, C}.

R2: **Scope**: The scope or 'system space' of the requirements must be defined. All specified qualifiers for requirements must be relevant to the system space.

Note: Scope states the 'overall system boundaries'. The scope for specific requirements is generally specified using {qualifiers}. See Section 2.7 for discussion of qualifiers. Use a Scope parameter if you want an explicit definition.

R3: **Qualifier Conditions**: Using qualifiers, requirement specifications must adequately cover the time period (When: long term and short term) and the physical scope (Where) for the system and, must state any known dependency on conditional states or events (If).

R4: **Rationale**: The rationale or justification for a requirement and for specific aspects of it should be given. *Use the Rationale parameter.*

R5: **Dependencies**: Any conditions or circumstances, which a requirement depends on for relevance or authority, must be specified. (*Use the 'Dependency' parameter, or any other relevant means.*)

R6: **Internal Links**: All specified requirements can be grouped into relevant hierarchical levels of requirements. Linkage to related requirements should be explicit and complete.

For example, use Planguage specifications such as:

- *Hierarchical tags (for example, 'System.Subsystem.Component').*
- *'Consists Of' or 'Includes' to link to lower hierarchical levels.*
- *'Is Part Of' to link to higher hierarchical levels.*
- *'Supports' and 'Is Supported By' to explicitly specify any intended direct links.*
- *'Impacts' and 'Is Impacted By' to explicitly specify impacts including any side effects (Impact Estimation table linkage).*

Illustration: Rules for reviewing requirements. Violation of rules determines the presence of a defect. Source:CE.

Spec Templates:

Have all required parameters been described, and have all parameters described been described correctly, and according to the template hints?

Note: See Scalar Requirement Template.

========== **Resource Requirements** ==========

Resource Requirement:
<**{Financial Resource, Time Resource, Headcount Resource, others}**>: <State tags of the resource requirements>.
Note: See Scalar Requirement Template.

========== **Design Constraints** ==========

Design Constraint: <State tags of any relevant design constraints>.
Note: See Design Specification Template.

========== **Condition Constraints** ==========

Condition Constraint: <State tags of any relevant condition constraints or specify a list of condition constraints>.

========== **Priority and Risk Management** ==========

Rationale: <What are the reasons supporting these requirements? >.
Value: <State the overall stakeholder value associated with these requirements>.
Assumptions: <Any assumptions that have been made>.
Dependencies: <Using text or tags, name any major system dependencies>.
Risks: <List or refer to tags of any major risks that could cause delay or negative impacts to the achieving the requirements>.
Priority: <Are there any known overall priority requirements? >.
Issues: <Unresolved concerns or problems in the specification or the system>.

========== **Evolutionary Project Management Plan** ==========

Evo Plan: <State the tag of the Evo Plan>.

========== **Potential Design Ideas** ==========

Design Ideas: <State tags of any suggested design ideas for this system, which are not in the Evo Plan>.

Figure 2.9
Requirement specification template. This is a summary template giving an overview of the requirements.

Illustration: lower part of a 'template with <hints'. For requirements. This framework helps reviewers learn, remember, and accurately perform reviews, according to corporate standards. The template and the <hints> are a de facto representation of the specification standards. Source: CE.

Concept Definition Standards:

These define the specification parameters precisely and officially, and can help us determine if specifications are correct, as intended.

Constraint **Concept *218**

A constraint is a requirement that *explicitly* and *intentionally* tries to directly restrict any system or process. A key property of a constraint is that a penalty or loss of some kind applies if the constraint is not respected.

Constraints include limitations on the engineering process, a system's operation, or its lifecycle.

"A constraint is 'something that restricts'."
(The American Heritage Dictionary (Dell))

Notes:

1. There are two kinds of constraints: Binary and Scalar.
   Binary constraints are statements that tend to include the words 'must' or 'must not': that is, they tend to make demands about what is mandatory for the system or what is prohibited for the system. Binary constraints are declared either by using a Constraint parameter or by specifying 'Type: Constraint.'

   Example:
   C1: Constraint: A design idea must not be made of material, components or products only produced outside the European Market, if there is any EU material which can be used.
   C2: Constraint: The design must contain ideas based on our own patents whenever possible.

Illustration: a portion of the concept definition for the term 'Constraint'. One of over 650 definitions in Planguage. The corporation can have their own tailored glossary. Clear and official definitions can help train people, and help people decide if a specification is correct in relation to these standards. <Hints> embedded in templates 9previous illustration) serve the purpose of giving an official definition for everyday needs.

Spec Process Standards:

Review Standards

Rates, and other quantitative guidelines:

- Checking Rates
- Checking Duration
- Entry Defect Density
- Exit Defect Density
- Review Staffing
- Expected % Effectiveness
- Remaining Defects Estimation (see next example)

Estimating the Remaining Major Defect Density

Assumptions:
A logical page (page) is 300 non-commentary words.

- 30 major defects/page have been found during SQC.
- Your SQC effectiveness is 60% and your SQC is a statistically stable process.
- One sixth of your attempts to fix defects fail (One sixth is average failure to fix).
- New defects are injected during your attempts to fix defects at 5%.
- The uncertainty factor in the estimation of remaining defects is ±30%.

Probable remaining major defects/page = 'Probable unidentified majors' + 'Bad fix majors' + 'Majors injected'

Let E = Effectiveness expressed as a percentage (%) = 60%

Probable unidentified majors = major defects acknowledged-by-editor for each page at Edit × (100 – E) / E = 30 major defects/page found × (100 – 60) / 60 = **20 major defects/page**.

Bad fix majors = One sixth of fixed majors = So, of 30 attempted fixes, **5 major defects in each page** are not fixed.

Majors injected = 5% of majors attempted to be fixed = **1.5 major defects/page**.

Probable remaining major defects/page = 20 + 5 + 1.5 = 26.5 remaining major defects/page.

Taking into account the uncertainty factor of ±30% and rounding down to the nearest whole number gives **26 ± 7 Remaining Major Defects/Page**

(Minimum = 19, Maximum = 33 remaining major defects/page).

Example of a process for estimating remaining defects after a review. Source CE page 249, Fig. 8.7

Review Processes

See above example for a sub-process of a review process.

Review Templates

For example the one below for a simplified review process.

SQC Date: **May 29, 200X**. SQC Start Time: _____
SQC Leader: **Tom**.
Author: **Tino**.
Other Checkers: **Artur**.

Specification Reference: **Test Plan**.
Specification Date and/or Version: **V 2** Total Physical Pages: **10**.

Sample Reference within Specification: **Page 3**.
Sample Size (Non commentary words): **approx. 300**.

Rules used for Checking: **Generic Rules, Test Plan Rules.**
Planned Exit Level (Majors/logical page): _____ or less.

Checking Time Planned (Minutes): **30**. Actual: **25**.
Checking Rate Planned (Non commentary pages/hour): **2**.
(*Note this rate should be less than 2 logical pages/hour*)

Actual Checking Rate (Non commentary words/minute): _____
Number of Defects Identified by each Checker:
Majors: **6, 8, 3**. Total Majors Identified in Sample: **17**.
Minors: **10, 15, 30**.

Estimated Unique Majors Found by Team: **16 ± 5**.
(*Note 2 × highest number of Majors found by an individual checker*)

Estimated Average Majors/Logical Page: ~**16 × 3 = 48**.
(*A Logical Page = 300 Non commentary words*)
Majors in Relation to Exit Level: **48/1 (47 too many)**.
Estimated Total Majors in entire Specification: **48 × 10 = 480**.
Recommendation for Specification (*Exit/Rework/Rewrite*): **No exit, redo and resubmit.**

Suggested Causes (of defect level): **Author not familiar with rules.**

Actions suggested to mitigate Causes: **Author studies rules, All authors given training in rules.**

Person responsible for Action: **Project Manager**.
SQC End Time: **18:08**. Total Time taken for SQC: _____

Example: a specialized review template, filled out example, for capturing data and making estimations and drawing conclusions. Source CE, page 245.

Digital Review Tools
- Spec Analysis Tools
- Review Process Databases and Data Collection and Reporting

Note: while there are a few general tools specialized in collecting review data, primarily for software reviews, I have observed that most of my clients make their own tool, usually starting from a spreadsheet.

Review Training Materials:
- Slides, Papers, Books, Teacher Notes, Exercises, Tests.

Organizational Policy

Management has to make some clear decisions, about reviews, and make them stick. Here is an example of a set of management policy ideas, I would recommend:

1. REVIEW TO RULE: All technical specifications will be reviewed against their standards (rules, templates, concept definitions, exit and entry conditions), and with regard to related specifications (like requirements which were input to the spec writing process).

2. QC BY SAMPLING: At the minimum a sufficient sample of a large specification will be taken, in order to determine its defect density.

3. REVIEW RIGOROUSLY: The reviews will be conducted with corporate defined rigor, and sufficient rigor for management to trust their conclusions.

4. DATA DRIVEN REVIEWS: The data collected from reviews will be made available for the purpose of improving the review process, and the related specification processes.

5. ENTRY CONTROL: There will be intelligently defined entry conditions to start reviews, that will prevent us wasting time on them.

6. EXIT CONTROLS: There will be intelligently defined formal exit conditions from reviews that will be respected, and will prevent us from approving work under pressure, which is going to cause problems later.

7. QUALITY MEASURED: The primary exit condition for a specification shall be based on a realistic measure of major defects/page, and a realistic estimate of remaining defects per page after corrections. If there is any doubt that the level of exit defects standard has been met, a new sample review will be used to determine safe exit.

8. DEFECT DENSITY MANAGEMENT: The purpose of most reviews will primarily be to determine if the specs are of corporate standard quality, in terms of *major defects possibly remaining per page*. Approval decisions will *not* be made in the review process *itself*, but will *use the quality data* from the review (major defects/page remaining) as partial input to an approval process.

9. AVOID CLEANUP: The purpose of reviews should never be to clean up bad engineering specification work. The reason is that the best review processes are very ineffective in cleaning up (20%-60% level). Bad work needs to be re-done properly.

10. REVIEW EARLY: We need to do partial reviews early in the production of a large quantity of specification. For example after a week of work, a random, representative sample of specs should be taken, especially of untried and unproven individuals, teams or suppliers. We need to manage QC so well that we never have a large amount of work that is done substandard.

Summary

It is possible to engineer your review processes to meet quantified engineering requirements for performance and cost. A review can be far more rigorous and objective than 'people at a meeting talking about a specification'.

As I finish this paper within a page limit of 15, I feel more could be said – which is not surprising as I have once written a 500 page book about the subject [SI]. But I hope the lists and ideas in the paper make it clear to the reader that we can quantify the review discipline, and then engineer the process to meet quantified requirements. I hope you see the tools to do the job?

References

Gilb, Tom [SQC]: Specification Quality Control, INCOSE 2005, Rochester NY Paper.
http://www.gilb.com/community/tiki-download_file.php?fileId=57

Gilb (SI) and Dorothy Graham, Software Inspection, 1993, Pearson Publishing. 2007 approximately 14th printing.

Gilb, Tom [RBDR]. Rule-Based Design Reviews:
http://www.gilb.com/community/tiki-download_file.php?fileId=62
Presented at INCOSE 2007, San Diego.

Gilb, Tom [CE], Competitive Engineering, A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage, ISBN 0750665076, **2005**, Publisher: Elsevier Butterworth-Heinemann. Sample chapters will be found at Gilb.com. There is a Chapter on Reviews: 'Specification Quality Control'. Available from the author on email request.

Gilb.com [WWW]: www.gilb.com. our website has a large number of free supporting papers , slides, book manuscripts, case studies and other artifacts which would help the reader go into more depth

Gilb, Slides [I4MB], 1 day training, "Inspection for Managers'.
http://www.gilb.com/community/tiki-download_file.php?fileId=88
This is particularly rich in cases and facts. It may be used for your own presentations freely (with source credit).

INCOSE Systems Engineering Handbook v. 3 [SEH]
INCOSE-TP-2003-002-03, June 2006 , www.INCOSE.org

Biography

Tom Gilb is an international consultant, teacher and author.

His 9th book is '**Competitive Engineering**: A Handbook For Systems Engineering, Requirements Engineering, and Software Engineering Using Planguage' (August 2005 Publication, Elsevier) which is a definition of the planning language 'Planguage'.

In 1993, with co-author Dorothy Graham, he published the first book on 'Software Inspection'. Now in at least 14th Printing. He has assisted clients and taught review processes since early 1970's His 1976 book 'Software Metrics' had 85 pages of text on Inspections. In December 2002 an Agile Inspection was developed by Gilb's and practiced at Citigroup. With consequent 8:1 reported reduction in defects in requirements. This is based on measurement by sampling. It motivates engineers to follow standards, and avoid defect injection.

He works with major multinationals such as Credit Suisse, Schlumberger, Bosch, Qualcomm, HP, IBM, Nokia, Ericsson, Motorola, US DOD, UK MOD, Symbian, Philips, Intel, Citigroup, United Health, Boeing, Microsoft, and many smaller and lesser known others See www.Gilb.com .