No.2 Review

Hironori Washizaki

Twitter: @Hiro_Washi washizaki@waseda.jp

http://www.washi.cs.waseda.ac.jp/

Ver1.1

Review [Pressman05]

- A process or meeting during which a software product is examined by a project personnel, managers, users, customers, user representatives, or other interested parties for comment or approval[IEEE 1028]

- Product quality improvement: technical review
- Process quality improvement
- Management
- Consensus and approval
- Education

Review process and related tech.

- 欠陥の発見と除去の手順 Process
  - 1. 兆候の特定 Specify symptom
  - 2. 兆候から欠陥の位置と問題内容の特定 Specify defect location and problem
  - 3. 修正方法の決定 Decide how to fix it
  - 4. 修正 Fix it
  - 5. 問題解決の確認 Confirm the problem has been solved
- 技術
  - コンパイラ Compiler: 自動。ただし兆候を示すのみ。automated, but, only indicating symptoms
  - レビュー Review: 兆候ではなく欠陥に集中。ただし時間＋難しい。Focusing on defects rather than symptom. Time-consuming and difficulty.
  - テスト Testing: ほとんど機能を検査可。ただし兆候のみ＋網羅困難。Can cover all functions. Focusing on symptom. Hard to test all things.
  - 静的テスト Static testing: 欠陥の一部を自動発見 Finding some of defects automatically.

Defects?

- 欠陥（Defect）: Fault or failure [IEEE Std982]
- 障害（Fault）:
  - Hardware defect
  - Incorrect step, process, or data definition in a computer program [IEEE Std 610]
- ずれ（Anomaly）: Anything observed in the documentation or operation of software that deviates from expectations [IEEE Std829,1028,1044]
- 故障（Failure）: Inability of system or component to perform its required functions within specified performance requirements [IEEE Std 610]
- 課題（Issue）: Concern suspected of defect but NOT determined as defect explicitly
- エラー（Error）
  - Difference between observed values/conditions and expected correct ones
  - Fault, failure, mistake [IEEE Std610]

Work: technical review

- Goal: motivating methods of efficient and effective review

- Target: source code corresponding to design
- Objective: detecting defects and issues
- Time: 10min

Retrospective of the work

- Ad-hoc review
  - Quick response of self or colleague
  - Easy, reviewer-dependent, non-systematic
  - Often quite inefficient
- What needed for systematic and efficient review
  - What to record?
  - How to proceed and manage?
  - How to read?

What to record

- Severity: Better to identify many critical defects
  - 1. Critical. MUST BE resolved.
  - 2. Major. SHOULD BE resolved.
  - 3. Minor. Resolution needs to be negotiated
- Type
  - Functional defects: lacking functionality, timing-error, incorrect logic, interface-mismatch, …
  - Evolvability defects: structure, visual representation, documentation
  - Issue: concern suspected of defect but NOT determined as defect explicitly
- When injected: requirements, design, construction, …

Mäntylä, M. V. and Lassenius, C., "What Types of Defects Are Really Discovered in Code Reviews?," IEEE Trans. Softw. Eng., vol. 35, no. 3, pp. 430-448, 2009.

Defects recording form

Project		XXX system	Current phase	Constructio	Page No. / Total			1 / 2
Target		source code	Ver	1.0				
Time and date		14:00 2014/4/16	Amount of time	120min	Perspective: Designer, Teter, User			
Modelator		Ichiro Tanaka (1W10XXX)	Recorder	XXX	Type: Func., Evo.. Issue			
Attendee		XXX (NNN), YYY (NNN), …			Severity: 1 Critical, 2 Major, 3 Minor			
ID	Location	Content	Perspective	Type	Severity	Injected	Action	Date
1	XXX.java	The necessary function of XXX under the state of YYY is not defined.	D	F	1	Design		
2	YYY.java Line 20–25	There is no way to cofirm the value of …	T	I	2	Construction		
3	ZZZ.java Line 30	The meaning of the attribute … is hard to understand by its name.	D, T	E	3	Construction		

参考: 日本科学技術連盟, ソフトウェア品質技術者 初級セミナー 資料, 2013.

Review methods: formality

- 非公式（非形式的）Informal
  - アドホックレビュー: 同僚に即席で意見をもらう。欠陥の指摘が属人的。
  - Ad-hoc: quick response of self or colleague. Review-dependent.
- 中間 Medium
  - ピアレビュー: 仲間たちによる幾らか形式的なレビュー。Peer review: review by peers.
  - パスアラウンド: 複数への配布・回覧によるレビュー。Pass around software and obtain review results.
  - ペアプログラミング: 1台のPCを2人で共有してプログラミング。常時の対話とレビュー。Pair programming.
- 公式（形式的）Formal
  - セルフ（個人）レビュー: 作成者（または非作成者）が個人で、定義された形式に基づいて自身（または他人）の成果をレビュー。小規模。コンパイル前。Self review according to well-defined process and forms, usually before compilation.
  - ウォークスルー: 作成者を含めて複数で対話しつつ、レビューを仮定して成果を確認していく。手順があまり公式でない。Walkthrough: reading and confirmation through the target software by group
  - チーム・レビュー: チーム・集団による多様な観点に基づくレビュー。Team review leads to diversity of perspectives.
  - インスペクション: 関与者の役割や検査基準、手順、記録方法などを明確化・文書化して複数名で実施。大規模。テスト前。Inspection: large scale and strict review method by group. Developed by Michael Fagan.

Review methods: reading

- アドホック Ad-hoc (ABR)
  - 適当に。簡単に。
  - 抜けの可能性大。Hard to cover everything
- チェックリスト Checklist-based (CBR)
  - 品質特性などの特定観点に基づいた検査項目一覧の利用 A list of checking items such as based on quality characteristics and coding standards
  - 大抵は、コーディング標準を組み込んで併用
  - チェックリストの肥大化、修正・改善忘れに注意。Ensure updating checklist
- シナリオ Scenario-based (SBR)
  - 利用・品質シナリオによるシミュレーション。例: ATAM Usage/quality scenario-based simulation
  - 列挙に手間がかかる Time-consuming
- パースペクティブ Perspective-based (PBR)
  - 参加者が様々な利害関係者（例えばユーザ、テスト担当者）になりきって各立場・観点から検査 Inspection from various perspectives of possible stake holders
  - 手間がかかる Time-consuming

チェックリストとは Checklist

- 成果の欠陥発見のために実施すべき一連の処理ステップ、項目集
- A set of items and/or procedural steps for finding defects in software
  - より多くの欠陥の発見、時間の短縮。
  - For finding more defects in less time
  - 項目は、品質特性や目的で分類・階層化されることが多い
  - Items are usually classified by characteristics and objectives.
  - コードが対象の場合、コーディング標準（注意すべき・守るべき項目集）を含む、あるいは、一体化
  - Sometimes including coding standards
- 項目のタネ Source of items
  - 言語仕様（プログラミング言語、UMLなど）
  - Programming and modeling language specification
  - 過去に頻出した欠陥型
  - Frequent defect types in previous developments
  - 経験に基づく「よい」習慣
  - Empirically verified good practices
- 例:
  - 全てのファイルはオープンされた後、必ずクローズされているか？ Are all files always closed after opened?
  - モジュール、変数、関数の名前は簡潔で、機能を適格に表しているか？ Are names simple and appropriate for representing targets?

Available checklists

- 公開チェックリスト Checklist
  - C++, Ada のコードレビューチェックリスト C++ Ada code review checklist [Humphrey01]
  - システム、ソフトウェア成果全般の品質特性別チェックリスト [小笠原06]
  - 要求仕様書、設計、COBOLコード、テストのレビューチェックリスト Review checklists for req. spec., design, ode and testing [Freedman87]
- 公開コーディング標準 Coding standard
  - C: GNU Coding Standards, MISRA-C, Safer-C, C言語コーディング作法 など
  - C++: Effective C++, High Integrity C++ Coding Standard など
  - Java: Writing Robust Java Code, Sun Java Code Conventions, Javaコーディング規約 など
    - 著:Scott W. Ambler, 訳 : 高橋徹, 頑健なJavaプログラムの書き方
      The AmbySoft Inc. Coding Standards for Java v17.01d, http://www.alles.or.jp/~torutk/oojava/codingStandard/writingrobustjavacode_pidid93_c11.html#doc1_id2248
    - コーディング規約, http://www.objectclub.jp/community/codingstandard/

CBR

- 0. 対象成果（およびチェックリスト、記録様式）を印刷 Prepare materials, checklist and recording form
- 1. 成果全体を調査し、対象が求められる全機能を実現していることを確認し欠陥があれば記録 Scan entire material and functionality
- 2. 成果中の各対象ごとに全項目を確認し欠陥があれば記録 Scan each module/part
  - 全体設計 -> 型 -> 変数 -> 構文
  - Whole design -> type -> variable -> statement
  - パッケージ -> クラス/インタフェース -> メソッドシグニチャ/フィールド -> （メソッド内の）構文/変数
  - Package -> class/interface -> method signature -> method inside
- 3. 成果全体を調査し項目にない予想外の問題・欠陥の探索 Re-scan entire material and specify unexpected issues
- 4. チェックリストの改善 Update checklist
  - 累計をとり欠陥発見に繋がっていない項目の削除 Remove unnecessary items
  - 重複・類似の項目をグルーピング Re-group redundant/similar items
  - 新たな問題・欠陥を項目として新規追加 Add newly found issues as items

SBR

- Scenarios as stories (i.e. what if …)
- Focusing on stakeholders' concerns
- Representation of functional and non-functional requirements
  - Usecase scenarios
  - Quality scenarios

[繁在家08] 繁在家学, 鷲崎 弘宜, "品質要求駆動型のアーキテクチャ分析設計手法によるWEBシステム開発", 情報処理学会研究報告ソフトウェア工学研究会報告 SE160, pp.33-40, 2008.

PBR

- Quality characteristics: quality models, quality scenarios
- Actors: designer, tester and user

Role	Method of reading	Concrete considerations, for example:
End user 利用者	Read from the viewpoint of end users, such as satisfaction of requirements, needs and UseCases.	・Are all needs and requirements satisfied by the target? ・Is it easy to use the target?
Tester テスト担当者	Read from the viewpoint of testers, such as ease of testing and adequacy of necessary information for test design and implementation.	・Is it easy to test the target? ・Is there enough information for testing the target? ・Is the target robust for any input?
Designer 設計者	Read from the viewpoint of designers, such as design complexity and future design extensibility.	・Is the design complexity of the target appropriate for future maintenance? ・Is the design quality enough and adequate? ・Is it easy to extend the target design?

日本科学技術連盟, ソフトウェア品質技術者 初級セミナー 資料, 2013.

Fact of review

- Formal review could find at most 75% of defects
- Effectiveness of reading methods
  - PBR is superior to ABR and CBR
  - More experiences, more defects in SBR and CBR [野中03]
- Effort benchmarks [Weiter00][Humphrey00]
  - Req. spec. 2 pages/hour
  - Design spec. 5 pages/hour
  - Code 200SLOC/hour

岡本博幸ほか, “要求仕様書の特性に着目した個人レビュー手法の実験的評価”, SQiP研究会, 2003
日本科学技術連盟, ソフトウェア品質技術者 初級セミナー 資料, 2013.
Edward F. Weller, Practical Applications of Statistical Process Control, IEEE Software, May/June 2000.
W.S. Humphrey, The Team Software Process (TSP), CMU/SEI-2000-TR-23, 2000.

Summary of review

- Review is about reading and commenting
- Various objects: identifying defects, management and education
- Various ways: process formality and reading methods
- “Good” organizations tend to
  - deal with anomalies in addition to usual defects
  - continuously improve review methods
  - utilize review results for further prevention of similar defects/issues in other projects