Software Quality Management I

Lecture 9

Software Engineering
TDDC88/TDDC93
autumn 2013

Kristian Sandahl
Department of Computer and Information Science
Linköping University, Sweden
kristian.sandahl@ida.liu.se
A Software Life-cycle Model
Which part will we talk about today?

- Requirements
- System Design (Architecture, High-level Design)
- Module Design (Program Design, Detailed Design)
- Implementation of Units (classes, procedures, functions)
- Unit testing

Validation and Verification:
- Validate Requirements, Verify Specification
- Verify System Design
- Verify Module Design
- Verify Implementation

Testing:
- Acceptance Test (Release testing)
- System Testing (Integration testing of modules)
- Module Testing (Integration testing of units)

Support:
- Project Management
- Software Quality Assurance (SQA)
- Supporting Tools
- Education

Maintenance
Agenda - What will you learn today?

Part I
Introduction to Software Quality

Part II
Software Metrics
Views on quality

- Transcendent – something we learn to recognize
- Product-based – measurable variable
- Usage-based – in the eyes of the beholder
- Manufacturing-based – conformance to requirements
- Value-based – market sets the value
The Shewhart cycle

PDCA Cycle

Decide goal (the right quality)
Select process (activities)
Determine present state
Run the process (project)

Act
Evaluate process
(Change the process)
Evaluate PDCA

Plan
Formulate facts about goal fulfilment

Check

Part I
Introduction

Part II
Software Metrics
Levels of quality assurance

- Appraisal – eg. defect detection
- Assurance – eg. prediction of defects
- Control – adjust the process
- Improvement: reduce variation, increase precision

Analysis | Design | Coding | Test-cases
---|---|---|---
Measurement | Measurement | Measurement | Measurement

Measurement
Quality factors

- Correctness
- Reliability
- Efficiency
- Usability
- Integrity
- Maintainability
- Flexibility
- Testability
- Security

- Portability
- Reusability
- Interoperability
- Survivability
- Safety
- Manageability
- Supportability
- Replaceability
- Functionality

Measuring these requires both research, experience and imagination.
Part II
Software Metrics
Most common use:

- **Measurement** – directly measured on:
  - Document, no of pages
  - Design, no of model elements
  - Code, no of lines
  - Process, iteration length
  - Quality, avg no of hours to learn a system

- **Metrics** – is a combination of measurements, eg. number of faults found in test/hours of testing
Cyclomatic complexity has a foundation in graph theory and is computed in the following ways:

1. Cyclomatic complexity $V(G)$, for a flow graph, $G$, is defined as:

   $$V(G) = E - N + 2P$$

   E: number of edges
   N: number of nodes
   P: number of disconnected parts of the graph

2. Cyclomatic complexity $V(G)$, for a flow graph, $G$, with only binary decisions, is defined as:

   $$V(G) = b + 1$$

   b: number of binary decision
Examples of Graphs and calculation of McCabe’s Complexity Metric

E= 1, N=2 , P=1
V= 1-2+2 = 1

E= 4 , N=4 , P=1
V= 4-4+2 = 2

E= 2 , N= 4 , P=2
V= 2-4+4 = 2

E= 10, N= 7 , P=1
V= 10-7+2 = 5

E= 12, N= 11 , P= 3
V= 12-11+6 = 7
Software metrics

- Usage-based metrics
- Verification & Validation metrics
- Volume metrics
- Structural metrics
- Effort metrics
- Direct measurement
- Indirect measurement

Note: Pedagogical model only!
Usage based metrics - example

- **Description:** Number of good and bad features recalled by users.
- **How to obtain data:** Set up a test scenario. Let test users run the scenario. Collect number of good and bad features in a questionnaire afterwards.
- **How to calculate the metric:** Take the average of number of good and no. bad features. Two values.
- **Relevant quality factor:** Relevance – many good and few bad features indicates a good match with the users’ mind-set.
Verification and validation metrics - example

- Description: Rate of severe defects found in inspection of design description.
  - How to obtain data: Perform an inspection according to your process. Make sure that severity is in the classification scheme.
  - How to calculate the metric: Divide the number of defects classified with highest severity with total number of defects in the Inspection record.
  - Relevant quality factor: Safety – a high proportion of severe defects in design indicates fundamental problems with the solution and/or competence.
Volume metrics - example

- **Description:** Number on non-commented lines of code.
- **How to obtain data:** Count non-commented lines of the code with a tool.
- **How to calculate the metric:** See above.
- **Relevant quality factor:** Reliability – it is often hard to understand a large portion of code, the fault density is often higher for large modules.
Structural metrics - example

- Description: Maximum depth of inheritance tree.
- How to obtain data: Count the depth of the inheritance tree for all classes with a tool.
- How to calculate the metric: Take the maximum value of the classes.
- Relevant quality factor: Understandability – It is hard to determine how a change in a higher class will affect inherited/overridden methods.
Effort metrics - example

- Description: Time spent in testing.
- How to obtain data: Make sure that testing activities are distinguished in time reporting forms. Make sure that all project activities are reported.
- How to calculate the metric: Sum the number of hours for all activities in testing for all people involved.
- Relevant quality factor: Testability – a comparably long testing time indicates low testability.
The Goal Question Metric approach

Outside the written exam we can use a top-down approach: Goal-Question-Metric (GQM)

<table>
<thead>
<tr>
<th>Goal</th>
<th>Purpose</th>
<th>Improve</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Issue</td>
<td>the timeliness of</td>
</tr>
<tr>
<td></td>
<td>Object (process)</td>
<td>change request processing</td>
</tr>
<tr>
<td></td>
<td>Viewpoint</td>
<td>from the project manager's viewpoint</td>
</tr>
<tr>
<td>Question</td>
<td></td>
<td>What is the current change request processing speed?</td>
</tr>
<tr>
<td>Metrics</td>
<td></td>
<td>Average cycle time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Standard deviation</td>
</tr>
<tr>
<td></td>
<td></td>
<td>% cases outside of the upper limit</td>
</tr>
<tr>
<td>Question</td>
<td></td>
<td>Is the performance of the process improving?</td>
</tr>
<tr>
<td>Metrics</td>
<td></td>
<td>Current average cycle time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>* 100</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Baseline average cycle time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subjective rating of manager's satisfaction</td>
</tr>
</tbody>
</table>

Basili, Caldiera, Rombach (1994)