

Ericsson Software Improvement through Software Quality Ranking

Sigrid Eldh

Verification Expert

Ericsson AB, Älvsjö

PhD Student @ IDT Mälardalens University

Sigrid.Eldh@mdh.se Sigrid.Eldh@ericsson.com

Ericsson



- ◆ Ericsson AB; CCND Core Network Unit
 - ◆ 3(4) Major Platform's for Telecom applications
 - ◆ APZ (AXE), CPP (3G), TSP and OTP
 - ◆ Own hardware, firmware, OS (partly), and several other proprietary solutions on network traffic
- ◆ My role is to support R&D verification at all levels and thus quality improvements, thus tools, measurements, methods and “how”...

The improvement

- ◆ Maturity in "old", but always newest, and but "bigger/smaller", faster, more complex with less people and not less but increased product quality -> how?, and where to spend the effort?
- ◆ Analysis resulting on many improvements, thus testing (and robustness!) is major (=make product "cheaper")
- ◆ One area is Component Test (the designers themselves must test better to achieve robustness)
 - ◆ Platform must be able to be "used" by applications!

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

The Result



- ◆ A improvement package now being deployed
- ◆ Ranking of components
- ◆ Tool support
- ◆ Better knowledge and understanding of quality matters
- ◆ Assessments
- ◆ "doing the right thing, doing it right"

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Software Design Quality Rank for software components

- ◆ 5 Levels, where 3 is where it is economical to be for most normal components (and should be the final goal for a project)
- ◆ Level 4 and 5 for safety-critical and extremely central components
- ◆ Level 1 2 3 are steps of improvement for most components
- ◆ First time, try level 1 at most level 2.
- ◆ Also, one could define levels to achieve during a project
- ◆ Example: Level 1 for 1:st release

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Content of Rank — The general categories

- ◆ Purpose
- ◆ Goal
- ◆ Description & behaviour
- ◆ Used when
- ◆ Tests
- ◆ Functionality
- ◆ Dependencies & Impact



2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Contents of Rank - Static Analysis

- ◆ Review
 - ◆ Method (R1 Walkthrough, R2 team, R3 extended team, R4 & R5 formal inspection)
 - ◆ Content (special focus during review)
 - ◆ Review Measurements (R2 and higher)
- ◆ Lines of code (only non-commented lines are measured)
- ◆ Complexity
 - ◆ Estimation (1-5, where 5 high) R1
 - ◆ Fan-in & Fan-out (for C++), nested calls (Coupling) R2
 - ◆ McCabe Cyclomatic Complexity R3
 - ◆ Halstead's Volume R3
 - ◆ Call-pair R4
- ◆ Flexelint or Lint (warning levels)
- ◆ Other: TBD, %Comments R2

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Contents of Rank - Run-time analysis

- ◆ Code Coverage
 - ◆ Estimated Statement Coverage (R1)
 - ◆ Statement Coverage (100 % R2)
 - ◆ Branch Coverage (80 % R3, 100% R4)
 - ◆ Modified Condition Decision Coverage (100% R5)
 - ◆ State Transition coverage (100% R5)
- ◆ Profiling
 - ◆ Performance
 - ◆ Memory error detection & analysis

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

One more specific example: Tests

1. Regression on selected old + "normal case"
(not particular fault handling)
2. + Fault handling, Inputs defined (valid and non-valid, Max-min for interface, 100% old test cases, 70% of new should be passed
3. +30 Automated test cases, Boundary values

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Summary



- ◆ Proposal to rank all software (comparable)
- ◆ Platform have more "rank 4" than others (which is not yet deployed)
 - ◆ aims to give better planning, and cost understanding
- ◆ Experience so far is very good
 - ◆ Easy to understand, communication and implement
 - ◆ Clear to set aims
 - ◆ Hard with tool dependence

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Ongoing Research: Transforming Experience on Failure Prediction to a Usable model

Sigrid Eldh

Verification Expert

Ericsson AB, Älvsjö

PhD Student @ IDT Mälardalens University

Sigrid.Eldh@mdh.se Sigrid.Eldh@ericsson.com

Why this approach?

- ◆ Failure predictions so far are not sufficient (practical) but many
 - ◆ Does not take circumstances into account
 - ◆ Are too simplistic
- ◆ Reliability growth models have the same problem as failure predictions
- ◆ Fault injection is not practical
- ◆ There exists experience from Ericsson and other companies that has been gathered for a long time

Important questions to answer for any project in industry

- ◆ When have we tested enough?
 - ◆ When are we ready for release?
 - ◆ How much failures can we expect in the future?
 - ◆ How can we measure our current quality?
- ◆ Assuming:
- ◆ “Any commercial software” but not safety critical systems

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Related work

- ◆ Fault-Injection (Voas)
 - ◆ Good, but difficult to get to industry general practice (there are exceptions)
 - ◆ Problem of introducing typical faults: input, integration (environment), logic
 - ◆ The havoc (cost, time) of removing the induced faults too
 - ◆ Gives good indication of “goodness of design (I.e. logic)” and test efficiency
- ◆ Failure models
 - ◆ Assumes code- debug-correct code- debug cycle (which is one level tested and not enough) – omits regression testing
 - ◆ Too simple, takes no other information into account which impact product quality
- ◆ Reliability models

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

My idea & approach

- ◆ There are many things that impact product quality
- ◆ Which of them really matter?
- ◆ My personal experience collecting data and what do I look for, result seems to fit with reality
- ◆ Make a general model that is also scalable and in principle easy enough to use (practical) but sophisticated enough to give better decision data than any other simpler model
- ◆ Check the model on many projects, levels and different areas

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Measurement areas

- ◆ Organisation (Project, size, time, ratio developers tester, ratio testers in project)
- ◆ System (Language, size, maturity, known architecture etc) (into a points system)
- ◆ Correction efficiency and number of faults
- ◆ Test efficiency and progress
- ◆ Some other variables

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

”Method”

- ◆ Calculation ”points” and ”risk” in conjunction with real data such as time (weeks), Failures and correction data. (Test efficiency also counts).
- ◆ Ranges: When is precieved good, medium and not ok
- ◆ Adding weeks (and adjust) should be possible meanwhile
- ◆ Compiling this into release time, and failure prediction (will be a rate or rather a ”curve”) for the system

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Questions not fully answered

- ◆ Are we measuring the ”right” things
 - ◆ Are we missing other important factors
- ◆ What measurements are strong and direct and what is implicit and indirect
- ◆ Can one minimize model?
- ◆ Is it useful, easy, practical and accurate enough?
- ◆ Is it applicable in all domains- generalibility?
- ◆ Is it scalable?

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Conclusions so far

- ◆ Predictions on failure
 - ◆ Assuming trendline holds
- ◆ Predictions on release-time "enough tested"
 - ◆ In weeks – but still reality check is not complete, cause current data are finished projects (will try model on ongoing projects this fall!)
- ◆ Prediction of product quality
 - ◆ Indirect of release-time and failure prediction
- ◆ All above is still "rough estimation" and more data is needed as well as fine tuning of model

2003-08-13

Copyright Sigrid Eldh, ARTES 2003

Summary

- ◆ If I succeed this will be a practical model for industry to use
 - ◆ Take more attributes into account
 - ◆ Explains the forces that decide outcome, hence hints at possible improvements of product quality and time
- ◆ Further work is needed – deployment and evaluation
 - ◆ Accuracy and possibility to generalize are important
- ◆ If you want to know more, or have feedback or questions, please contact me for discussion

2003-08-13

Copyright Sigrid Eldh, ARTES 2003