# Fault Slip Through Measurement Process Implementation in CPP Software Verification

Ž. Antolić
R&D Center
Ericsson Nikola Tesla d.d.
Complete Address: Krapinska 45, Zagreb, HR-10000, Croatia
Phone: +385 1 365 4584 Fax: +385 1 365 4082 E-mail: zeljko.antolic@ericsson.com

**Abstract - This paper gives an overview of Fault Slip Through (FST) process related to software verification improvements based on process measurement data. The Fault Slip Through Process is a way to secure that software faults are detected in the right verification phase. By learning of our earlier mistakes during verification, the organization can avoid doing them again. The process also implements mechanisms for enabling future efficiency work. Some proposals and future directions in the area of software integration and verification processes are given.**

## I. INTRODUCTION

Common situation in software industry today is that we are spending more than 50% of the development time for testing activities in order to achieve required quality level of our products [1]. Our intention is to reduce the testing time by preventing the slippage of faults through different testing phases. The right faults should be found in the right project phase. That requires a clear test strategy and focus on finding important faults early. Since most faults should be found earlier, we have to visualize the degree of fault slippage from earlier phases and provide feedback backwards on which types of faults that need to be captured earlier. In that way, we avoid doing the same mistakes over and over again, saving the testing effort and improving the software product quality. The root cause analysis and the right feedback to ongoing development projects is the key for success. Analyzing the findings and implementing the proposed improvements significantly reduce the repeating of the same faults in the future. Basic fault-slip-through measurement is ratio between faults found in the current verification phase that should have been found in earlier phase and number of faults found in the current phase.

Analyzing the results from previous projects we have found the following:

- 60-70% of faults found in current verification phase had slipped through previous verification phase (component test to function test, function test to system test, system test to customer);
- Fault-slip-through ratio is higher in early testing phases (component test compared with function test and system test).

After collection of results and root cause analysis, we have provided feedback to design project with recommendations for improvements and what should be avoided in order not to repeat the faults, and not allow fault slippage. By reducing the degree of fault-slip-through we wanted to achieve:

- Fewer stopping faults;
- Earlier and cheaper fault removal;
- Less redundant testing;
- Improved delivery precision.

One of the key improvements identified at the several workshops held during the year 2006 were to make Fault Split Through (FST) analysis and use the measurements to “close the gaps” [2]. That is to give feedback to earlier development phases so that improvements can be made in the organization to increase product quality in the most cost efficient way.

In the beginning of 2006 the R&D management decided to start an investigation regarding FST at CPP (Connectivity Packet Platform), and in July 2006 it was decided that FST shall be launched in our organization, based on study in [9].

## II. DEFINITION OF FAULT SLIP THROUGH

Fault Slip Through represents the number of faults not detected in a certain activity. These faults have instead been detected in a later activity [3].

Example (see Fig.1.): If a fault is detected in System Verification that was supposed to be detected in Unit Test, then there is a Fault Slip Through.

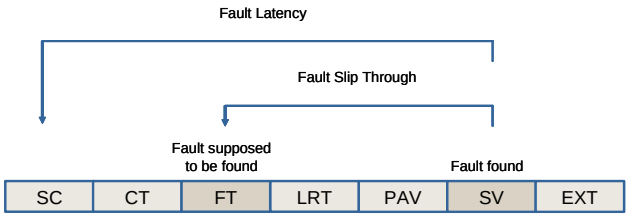



Fig.1. Definition of Fault Slip Through

The Test Strategy, Test Processes and CPP development process define in which phase’s different kind of faults are supposed to be detected [4], [5], [6].

If we take a look on fault patterns, it can be seen that some of the faults have common patterns, i.e. they are repeatable/systematic (see Fig.2.). This kind of faults is usually avoidable (waste). Fault slippages can to a large extent be considered ’waste’ [9].

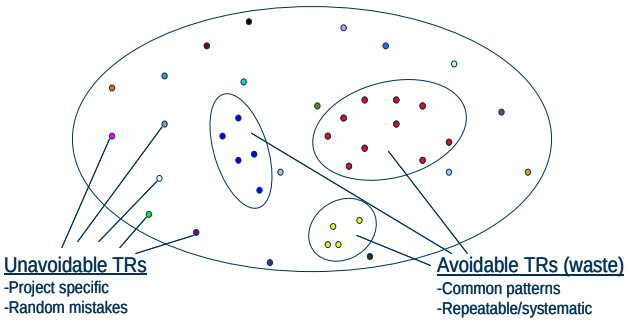



Fig.2. Fault Patterns

## III. THE FAULT SLIP THROUGH PROCESS

The Fault Slip Through Process is a way to secure that faults are detected in the right phase. By learning of our earlier mistakes, the organization can avoid doing them again. The process also implements mechanisms for enabling future efficiency work.

Right phase means the most cost efficient phase. A general guideline is that it's cheaper to detect faults early in the development process, "earlier is cheaper" [7]. This is not always true since it's also related to a cost to detect faults. Some faults might be more cost efficient to detect later in a development process.

Fault Slip Through analysis brings many advantages:

- Early and cost effective fault detection;
- Provides useful feedback regarding fault introduction in product and process;
- Enables fault slippage measurements which helps us to identify improvements areas throughout the development process;
- Learning from earlier mistakes and avoid doing them again;
- Less redundant testing and closer test coordination;
- Improved quality and less stopping TRs (Trouble Reports);
- Shorter lead times and improved delivery precision.

Fault Slip Through can also be used to:

- Evaluate effectiveness of verification methods and tools;
- Evaluate and predict product quality in a project;
- Input to characterize the capability and fault detection profile of an organization.

The FST can be related to the whole software development process, from specification to design and test. The quality of a product is built in during the early phases. The test at the end is only meant to be a confirmation of the adherence to the requirements.

Different kind of faults is supposed to be found in different phases. The most efficient and cost effective way is to capture the fault close to the introduction. Each development activity has to be responsible for its own errors [10].

An outstanding quality assurance method is reviewing each others work. It is no matter if it is a specification, source code or other kinds of written text. It is much more cost efficient to find faults in Reviews and Inspections than executing test cases [8].

With the FST measurements we are aiming for to give feedback to the different actors and teams. They have the responsibility to improve their own process and fault prevention strategy.

The Fault Slip Through TR (Trouble Report) analysis is an extension of the TR analysis. The FST process is therefore related to the TR Process. A Fault Slip Through Process can be divided into five different steps:

1. Fault Slip Through Analysis for each TR - Part of the TR analysis, "daily work";
2. Measure Fault Slip Through;
3. Analyze the Fault Slip Through results - Might trigger a Root Cause Analysis (RCA) - Can be made on all levels within the organization, at all times;
4. Identify and implement improvements;
5. Reporting and follow up.

The TR Fault Slip Through analysis shall be made for all CPP TRs and is an extension of the ordinary TR analysis. This analysis is the base for all further measurements and analyses and can be seen as a part of the daily work.

Five fields have been introduced in the TR handling tool on order to do the FST analysis. These five fields are the following:

- DIDDET – Where the fault was detected;
- SHODET – Where the fault was supposed to be detected;
- INTROD – Where the fault was introduced;
- TC – Was there a TC in the SHODET phase;
- INFO – Free text field for additional information.

Step two of the Fault Slip Through analysis is to measure fault slippage. Only the DIDDET and SHODET fields shall be used for Fault Slip Through measurements.

There are two ways to measure FST:

- Fault slippage from a phase;
- Fault slippage to a phase.

The project and/or line organization have the responsibility to generate the Fault Slip Through measurements.

At CPP a special FST Measurement Tool has been created for FST measurements [12].

The Fault Slip Through Matrix is the source to all measurements and further analysis. In the Fig.3. below is the Fault Slip Through Matrix described in detail, and how measurements are performed on a general level.

To make the FST measurements more useful, three KPI's (Key Performance Indicators) have been identified:

- FST to Design I&V;
- FST to System I&V;
- FST to CPP Customers.

where

- Design is the Document Inspection, Code Review, HW Basic Test, Component Test and Multiple Component Test phases;

## CPP FST KPI's

| DIDDET → / SHODET ↓ | DI | CR | HWBT | CT | MCT | FST to Design I&V: FT | LRT | QCT | FST to System I&V: SV | FST to CPP Customers: APP | NWL | EXT | FST TRs | Tot. TRs | FST from |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DI | 28 | 3 | 0 | 4 | 2 | 8 | 4 | 0 | 9 | 17 | 2 | 2 | 51 | 79 | 65% |
| CR | | 58 | 0 | 8 | 1 | 21 | 5 | 1 | 40 | 17 | 2 | 8 | 103 | 161 | 64% |
| HWBT | | | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 3 | 4 | 4 | 100% |
| CT | | | | 53 | 5 | 87 | 5 | 2 | 41 | 12 | 2 | 11 | 184 | 237 | 78% |
| MCT | | | | | 31 | 3 | 1 | 0 | 4 | 2 | 2 | 2 | 17 | 48 | 35% |
| FT | | | | | | 320 | 15 | 0 | 32 | 27 | 3 | 19 | 171 | 491 | 35% |
| LRT | | | | | | | 69 | 1 | 10 | 3 | 0 | 0 | 20 | 89 | 22% |
| QCT | | | | | | | | 21 | 0 | 1 | 0 | 0 | 1 | 22 | 5% |
| SV | | | | | | | | | 213 | 14 | 2 | 9 | 25 | 238 | 11% |
| APP | | | | | | | | | | 47 | 1 | 8 | 9 | 56 | |
| NWL | | | | | | | | | | | 4 | 8 | 8 | 12 | |
| WP Slip | | | | 19 | 3 | 75 | 6 | | | | | | 103 | 103 | |
| FST TRs | 0 | 3 | 0 | 31 | 11 | 194 | 36 | 4 | 137 | 93 | 14 | 70 | 593 | | |
| Tot. TRs | 28 | 61 | 0 | 84 | 42 | 514 | 105 | 25 | 350 | 140 | 18 | 70 | | 1437 | |
| FST to | 0% | 5% | 0% | 37% | 26% | 38% | 34% | 16% | 39% | 13% | | | | | |

No fault slippage (faults found in "right" phase)
Fault slippage (faults found in "wrong" phase)
Fault slippage between Work Packages (within the same phase)
Not relevant

Fig 3. Fault Slip Through Matrix

- Design I&V is the Light Regression Test, Function Test and Quality Criteria Test phases;
- System I&V is the System I&V phase;
- CPP Customers is Application, Network Level and Ericsson External Customer.

Fault Slip Through to a phase is measured vertically in the matrix. The summary of all phases "above" the measured phase is compared to the total number of TRs in that phase. Fault Slip Through from a phase is measured horizontally in the matrix. The summary of all phases "after" the measured phase is compared to the total number of TRs in that phase.

There are many ways to measure Fault Slip Through and it's up to the user of the process to decide how the measurements shall be performed.

It's important that the number of TRs in the measurement data is large enough to generate relevant statistics. The database query shall not be narrow so the number of TRs becomes to low.

Using FST for performance benchmarking of products and organizations is not recommendable because of:

- Product differences: Product maturity, complexity and architecture will affect fault slippage ratio;
- Process differences: Organizations using parallel testing processes tends to have higher fault slippage;
- Definition differences: Fault Slip Through might be defined differently.

The FST measurements are not covering the following scenarios:

- Slippage between product releases;
- Slippage between projects.

## IV. ANALYSIS OF FST RESULTS

Once the measurement results are available it's time to analyze the Fault Slip Through results. It is possible to identify:

- Phases with low slippage;
- Phases with high slippage.

Phases with low slippage can be used as good practices. Phases with high slippage can trigger a Root Cause Analysis (RCA) in order to identify why there are slippage problems in this phase [11].

The INTROD and TC fields can be very useful and support the analysis. It's the owner of each phase that's responsible to do the analysis.

Fault Slip Through is just not a measure, it is a concept for continuous improvements, and phases with high fault slippage need to be improved. A Root Cause Analysis (RCA) is one way to deal with the problems in this phase. An RCA is a problem analysis aimed at investigating why faults were not detected in the phase where they were supposed to be detected.

It's the line organization that's responsible to identify improvements, perform RCA and implement identified improvements.

Reporting and follow up of Fault Slip Through is important to keep focus on results and implemented improvements.

It's up to the process user to decide in what way and how often FST shall be reported. Reporting can be made both by project and/or the line organization.

Recommendation is to use already established ways of reporting such as:

- Monthly reports from Line and Project Management;
- Line Management meetings;
- Operating Steering Group (OSG) meetings;
- Project meetings.

## V. FAULT SLIP THROUGH GOALS

It is recommended that Fault Slip Through goals to be used are set by the organization. The goals can be set for the entire organization or for a certain project phase. The following goal setting process can be used:

1. Get baseline;
2. Set improvement goals
   - Example: FST to System Verification shall decrease by 10%;
3. Determine how to achieve the goal
   - Example: Do an RCA on phases with high slippage, and identify and implement suitable improvements;
4. Decide how and when to follow up goals
   - Example: At the OSG meetings;
5. Start over from step 1.

Use Fault Slip Through to a phase when defining FST goals. FST from a phase is not recommended since it can take long time to obtain required measurement data (the product must have been used live in the field for a while before you know the final value). FST from a phase is a better way to identify phases with high slippage.

Once a decent FST level is reached by the organization, the goal shall be to keep the slippage on this level.

It is not recommended to set FST goals to zero. Some amount of slippage is natural within a design organization. It's also the most cost efficient strategy since it's related to a high cost to find the last faults and difficult to determine when the product is completely free from faults.

The one project example of FST goals and follow-up of results is shown on Fig.4.

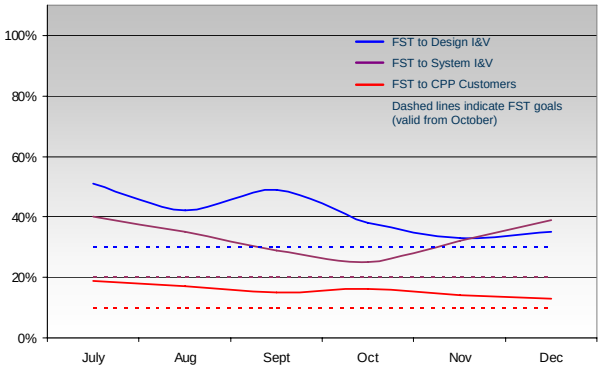


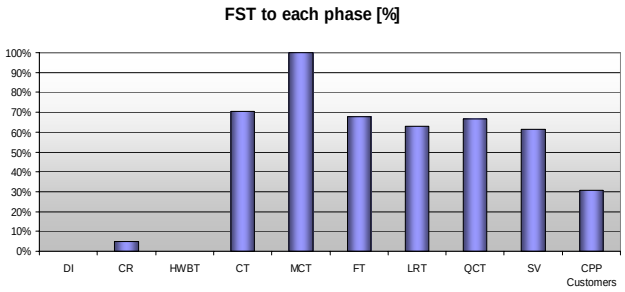

Fig.4. FST Goals and follow-up

## VI. ADDITIONAL ANALYSIS OF FST RESULTS

Based on the measurement results generated in FST Matrix, it is possible to perform additional analysis. Some of project examples are shown on the next figures.

### *A. FST to Each Phase*

The results of 'FST to Each Phase' measurement are shown on the Fig.5. It can be noticed that slippage level is about 60-70% between phases. It means that 60-70% of faults found in current verification phase are supposed to be found in the previous verification phases.




| | DI | CR | HWBT | CT | MCT | FT | LRT | QCT | SV | CPP Customers |
|---|---|---|---|---|---|---|---|---|---|---|
| FST TRs | 0 | 3 | 0 | 31 | 11 | 252 | 46 | 4 | 159 | 330 |
| Total TRs | 28 | 61 | 0 | 44 | 11 | 372 | 73 | 6 | 259 | 1081 |
| FST | 0% | 5% | 0% | 70% | 100% | 68% | 63% | 67% | 61% | 31% |

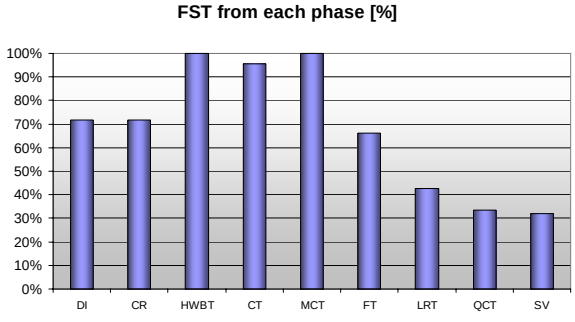
Fig.5. FST to Each Phase

### *B. FST from Each Phase*

The results of 'FST from Each Phase' measurement are shown on Fig.6. It can be noticed very high level of fault slippage from early phases of development.

The HW basic test has 100% of fault slippage, but too small number of TRs to make some conclusions.

The Document Inspection and Component Test are definitely areas for improvements. They have high number of TRs, and high percentage of fault slippage. In addition, fault removal in these phases is the most effective.



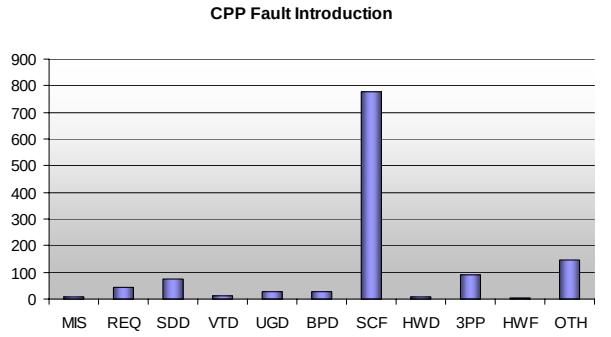

| | DI | CR | HWBT | CT | MCT | FT | LRT | QCT | SV |
|---|---|---|---|---|---|---|---|---|---|
| FST TRs | 71 | 148 | 4 | 275 | 17 | 234 | 20 | 1 | 47 |
| Tot TRs | 99 | 206 | 4 | 288 | 17 | 354 | 47 | 3 | 147 |
| FST from | 72% | 72% | 100% | 95% | 100% | 66% | 43% | 33% | 32% |

Fig.6. FST from Each Phase

### C. CPP Fault Introduction

The results of 'CPP Fault Introduction' measurement are shown on Fig.7. It can be noticed that most of the faults are classified as Source Code Faults, meaning that fault is introduced in coding phase. The more detailed analysis resulted with fact that these data are maybe not representative, because some of faults related to requirements and specification documents are classified as Source Code Faults. In fact, the source code is faulty, i.e. problem solution is implemented in the code, but fault origin is in many cases in the early development phases.

This measurement requires further improvements, and TR handlers' education about correct specifying fault introduction phase.

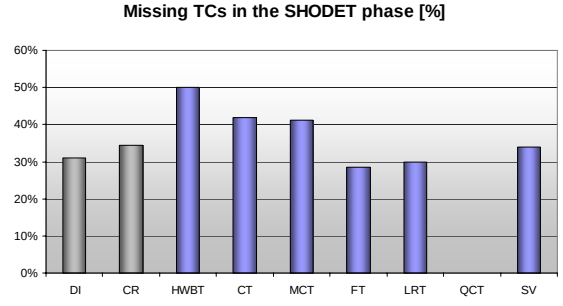



| MIS | REQ | SDD | VTD | UGD | BPD | SCF | HWD | 3PP | HWF | OTH |
|---|---|---|---|---|---|---|---|---|---|---|
| 8 | 44 | 74 | 13 | 26 | 29 | 779 | 7 | 89 | 4 | 148 |

Fig.7. CPP Fault Introduction

### D. Missing Test Cases

The results of 'Missing Test Cases' measurements are shown on Fig.8. It can be noticed high percentage of missing test cases in the verification phases. That means the fault would not occur or slip through if test case exists. Based on these results, we have started improvement activities on writing of new test cases, and improving of the existing test configurations. The positive effects are expected in the next development project.




| | DI | CR | HWBT | CT | MCT | FT | LRT | QCT | SV |
|---|---|---|---|---|---|---|---|---|---|
| FST from | 71 | 148 | 4 | 275 | 17 | 234 | 20 | 1 | 47 |
| Missing TCs | 22 | 51 | 2 | 115 | 7 | 67 | 6 | 0 | 16 |
| Percentage | 31% | 34% | 50% | 42% | 41% | 29% | 30% | 0% | 34% |

Fig.8. Missing Test Cases

## VII. FST IMPLEMENTATION IN THE ORGANIZATION

In order to implement the FST in the organization, it is recommended to take the following steps [9]:

- Determine the business goal of introducing FST, for example whether the goal is to reduce the number of customer faults with X% or to reduce the lead-time with X weeks;
- Create a common understanding and commitment;
- Identify a driver. Someone with authority and true interest in implementing the concept is crucial to make the implementation successful. Otherwise, it will as most other improvement work be down-prioritized every time a project emergency occur (which tend to be very often);
- Make sure that the test strategy is well-defined and communicated throughout the organization;
- Perform a baseline measurement on a finished project (or at least a subset of TRs from a project). Doing this is a good test to see that the measure is possible to apply in relation to the defined test strategy, and the baseline is very good to have as comparison when applying it on the first new project;
- Add the measure to the local TR process. If the measure is included in the TR process, follow-up will be a lot easier;
- Educate. People must understand how to report the measured data, and even more importantly understand why it is important to measure;
- Identify a pilot project to apply it in. This first project needs extra attention regarding follow-up of how well the measurement reporting works. Correct eventual issues directly;
- Visualize results early to determine status and to further show people that it is important (people tend to care more about visible measurements);
- Continuously monitor the status in relation to implemented improvements (and adjust measurement process when issues are identified).

Success factors for the implementation in first project are the following:

- Make sure that people understand the purpose
  - It is not just a measure for managers, it is about determining how efficient the process is, and identify improvements so that we can become better;
- Make people think in the same way
  - Explain the definition and educate with example Trouble Reports;
- Provide hints to use when unsure
  - Example: "In the component test all code should be executed". Means that if a code segment fails no matter the input, it should have been found in component test.
- In the beginning, check regularly that people follow the guidelines.

The activities related to deployment of FST process in the organization and CPP projects had required the following investments:

- FST Measurement Toll development (initial investment, 50 man-hours);
- Competence build-up for designers and testers (3 man-hours per designer/tester);
- Measurement activities in the project (1 man-hour per month for each project);
- Analysis of measurement results (5 man-hours per month for each project).

Savings achieved by implementation of improvements based on analyzed results are few times higher than investments, already in the first project where FST process is introduced.

Project quality manager is responsible for implementation, and control of the FST process. Test manager is responsible for data analysis, and suggested improvements implementation, based on collected results.

## VIII. CONCLUSION

The Fault Slip Through process is coming from software development projects at Ericsson for CPP product. We have started this process as improvement program, and have implemented in all development projects from the middle of year 2006. The data were collected and analyzed on monthly basis, and used as input for further improvement activities in the verification phase of the projects.

By applying Fault Slip Through process we have achieved in the short time period improvements of fault slippage to customer, improvements of test configurations, and improvements of test cases used in verification phase of the projects.

The further work on Fault Slip Through process is expected on more automated data collection, and more accurate data, less depending on software testers' judgment.

In the future, we can expect more demands on software product quality, reduced project lead-time, and reduced project budgets. The only possible way how to answer on these demands is to work continuously on the software development and verification process improvements, with focus on fault-free software delivered to our customers [11].

## REFERENCES


[1] ***, "An Introduction to Software Testing," IPL Information Processing Ltd., Bath, United Kingdom, 2002.

[2] ***, "The three main R&D problems and how to attack them", Internal Ericsson documentation, Stockholm, Sweden, 2005.

[3] ***, "Faults-slip-through measurement", Internal Ericsson documentation, Karlskrona, Sweden, 2005.

[4] ***, "PP&T I&V General Test Strategy", Internal Ericsson documentation, Stockholm, Sweden, 2005.

[5] ***, "Cello Packet Platform Integration and Verification Process", Internal Ericsson Documentation, Stockholm, Sweden, 2004.

[5] Z. Antolic, "Integration Centric Approach in Software Development as Solution for Process and Quality Improvements", Proceedings MIPRO 2006, 29th International conference, pp. 117-122, Opatija, Croatia, 2006.

[7] Z. Antolic, "Software Integration and Verification Process and Quality Improvements on CPP System", Proceedings MELECON 2006, 13th IEEE Mediterranean Electrotechnical Conference, Malaga, Spain, 2006.

[8] Z. Antolic and S. Golubic, "Integration & Verification - Possible approaches and impacts on software product quality", Proceedings MIPRO 2005, 28th International conference, pp. 170-175, Opatija, Croatia, 2005.

[9] L.O. Damm, "Fault Slip Through Measurements – Tutorial", Internal Ericsson documentation, Karlskrona, Sweden, 2006.

[10] E. van Veenendaal, M. Pol, "A Test Management Approach for Structured Testing", Netherlands, 1996.

[11] ***, "Towards Testing Excellence", Internal Ericsson documentation, Stockholm, Sweden, 2005.

[12] ***, "Faults Slip Through Process for CPP", Internal Ericsson documentation, Stockholm, Sweden, 2006.