

Automatic Measurement at Nokia Mobile Phones: A Case of SDL Based Software Development

Minna Pikkarainen¹, Matias Vierimaa¹, Hannu Tanner¹ and Raija Suikki²

¹VTT Technical Research Centre of Finland, VTT Electronics, P.O.Box 1100, FIN-90571
Oulu, Finland

{firstname.lastname@vtt.fi}

²Nokia Mobile Phones, P.O.Box 50, FIN-90571 Oulu, Finland

{raija.suikki@nokia.com}

Abstract. Software measurement forms a sound basis for monitoring software development process and software product quality. Implementing software measurement is, however, difficult and creates several challenges and possible pitfalls. These challenges include the establishment of measurement practices, effective metrics collection and the use of collected metrics to improve development practices or product. Automated software measurement can have a significant role when adopting software metrics for systematic use. This paper is based on the results of a measurement automation project at Nokia Mobile Phones during the years of 1999-2000. This paper describes the experiences from defining metrics for controlling new SDL based development practices. The process through which the automation was implemented is described in detail and implications of the findings are discussed.

1 Introduction

The embedded software industry grows very fast and is R&D intensive. Products with embedded software have become more sophisticated and especially the software inside the products has increased in size and become more complex [3],[18]. SW development plays a major role in product development [3], and typically forms 50-80% of the overall effort.

While strict time-to-market requirements and business needs demand shorter development cycles, the software industry needs to invest heavily in the development of new methods and techniques, and in the adoption of existing best practices [14]. New development methods and technologies are often challenging to introduce to existing development process since they require learning and training, and sometimes even new ways of thinking [7]. Therefore, the benefits of these methods become visible only after some time has passed after introduction [11]. Moreover, in critical software development projects, the project staff has only a limited time to introduce new methods and even less time to effectively analyse the impacts of these new methods.

This paper is based on the results of a measurement automation project at Nokia Mobile Phones during the years of 1999-2000. The purpose of this paper is to describe the experiences from and the process of using automatic software measurement to monitor SDL (Specification Description Language) based SW development within three software development projects. It was found that while measurement automation can be difficult in a volatile software development environment, it is, however, possible and gives benefits to software developers. Detailed discussion of the measurement automation processes and findings in the SDL based environment provide much-needed experiences that can be further used by other practitioners starting and implementing their measurement automation activities.

This paper is composed as follows: the next section presents the background of this work, including a brief description of SDL based SW development, software measurement and automation, and a description of Nokia Mobile Phones and their starting point for the SDL based measurement. Third section provides a description of the actual implementation of the measurement automation process. In the fourth section the key findings and lessons learned from the case projects are discussed. The last section concludes the paper with final remarks and outlines future actions.

2 Background

When an old tool is replaced or a new tool is added to a SW development environment the impact of changes has to be compared to the previous situation [21]. New methods and technologies affect the project manager's ability to predict schedules and to control the product development. Monitoring of impacts is therefore particularly important when implementing process changes [17]. Projects that take new SW development tools and technologies into use need to pay special attention to the controllability and predictability of SW development. In this retrospect, this section describes the concepts and purpose of SDL based SW development, software measurement and automation. In addition, the background for our case is outlined.

2.1 SDL Development

The fast-growing complexity of embedded systems necessitates new design methods and tools in order to solve the problems of design, analysis, integration and validation of complex systems [9]. SDL is a widely used language and it is considered especially suitable for the development of complex telecommunication systems.

SDL is used mainly in real-time, distributed and communicating systems to increase productivity, reduce errors and improve software maintainability [16]. An SDL development process is usually divided into three phases: requirements specification, high level design, and detailed design [16]. Typically, the procedure from requirements analysis to product implementation and testing involves the following steps [8]:

- Collect the initial requirements in a text document.
- Study system analysis results, Message Sequence Charts (MSC) or/and use-cases depicting typical scenarios. The resultant classes are implemented in SDL as block diagrams and data-type definitions.
- Complete the SDL diagrams and specifications to a level where they can be simulated and checked for consistency with the system requirement analysis.
- Use verification and validation to determine whether required properties are correctly and completely implemented. When SDL design has proved consistent with the requirements, code for the application can be generated.
- Make a test suite. Tests can be generated from the SDL specification.
- Generate code to create an executable test suite that can be run in a test system.
- Run the executable tests and test the application in the target environment.

The process of defining SW requirements and functional specifications using SDL is very similar compared to other SW development methods and languages. Only the notation for documenting the analysis results is different. Instead of data flow diagrams, an SDL based specification contains message sequence charts (MSC). The purpose of these charts is to provide sufficient understanding of the specification functionality, collect items that are missing from the specification to get an idea of the maturity level of the specification and to collect information of the protocol entities as well as to gain an understanding of the needed protocol entity partitioning [8].

2.2 Software measurement and automation

Measurement practices introduced to volatile software development must be as transparent as possible to the software developers. In order to be effective, metrics collection, analysis and the presentation of the results should be automated (e.g., [15]; [1]; [4]). While there is a great deal of literature on implementing software metrics programs, there is less agreement, however, on what exactly contributes to the success of implementing metrics [4]. Moreover, in spite of this literature being available, companies are facing serious problems initiating even the simplest metrics programs [4], [6].

Ideally, measurement activities should be driven by clear goals in order to focus and limit the costs [9]. Collected measurement data and the results of the measurements should be based on project and organisational goals [10]. The measurement process, in our case, was based on the GQM (Goal, Question, Metric) approach as defined by Latum and Lavazza [12];[13].

The GQM approach is commonly used in software measurement. The approach defines measurement goal which is then described by set of questions. Metrics are then defined to answer the questions. GQM uses structured interviews to define questions and metrics. Collected data is later analysed and presented in feedback sessions to find out whether project has met its measurement goals.

In automatic software measurement, the measurement process consists of manual and automatic process steps [3]; [10]. Manually implemented process phases are 1) Planning the measurement automation, 2) Defining measurement goals (for the first time) and 3) Defining metrics definition (for the first time). It is possible to use tool support in the goal and metrics definition phases [13], but the actual work is highly dependent on human decision [10]. The members of a measurement project should define:

- Data sources from where the measurement data will be collected
- Date (and time), or the trigger condition, when the data is collected
- Calculation templates for analysing the data
- Data storage where the result information will be saved.

Tasks that are repeated often should be made automatic. The tasks include, for example, the collection of the data from predefined data sources and calculation of the analysis data. However, the calculation formulas and the presentation format of the result graphs have to be predefined within the used tool.

The measurement automation process [10] starts by planning the automation, which includes defining the metrics, data sources, schedule of collection, and data analysis in detail. During the implementation of the environment for measurement automation the technical solutions for data retrieval are built and the first data analysis and presentation are carried out. Automated parts of the process are the repeated steps of collecting data, calculating the metrics, and the creation of analysis graphs. In the final phase the metrics' results and experiences are packaged.

2.3 The Case of Nokia Mobile Phones

Nokia Mobile Phones (henceforth NMP) is the world's largest mobile phone manufacturer. Nokia's comprehensive product portfolio covers all cellular protocols. As the functionality of mobile phones moves from voice-centric to multimedia, imaging, entertainment and business applications, the complexity of the software increases.

SDL and new unit testing methods were taken into use at NMP in three product development projects during the year 1999. This created needs to measure quality of design and implementation, and improve prediction and control of product development with new implementation language (i.e., SDL). A GQM based measurement project was launched as co-operation between the Technical Research Centre of Finland (henceforth VTT) and NMP.

Based on the experiences from previous measurement projects, measurement activities were seen as an important part of monitoring the status of NMP product projects. However, due to product time-to-market restrictions, the project staff had only a limited time to use for measurement activities. Thus, there was a need for a measurement project, which would specifically tackle the problems caused by the lack of time to collect the desired metrics. The purpose of the measurement project, which is the main

focus of this paper, was to automate the previous measurements in the product development projects. This was expected to enhance quality and predictability of SDL based product development projects.

As a result of the previous measurements the developers at NMP had many useful measurement experiences and documents including defined and analysed metric results. Due to the need for measurement automation, it was not efficient enough to define all metrics using the GQM approach. Instead, when defining metrics, the possibility of automatic collection, analysis and presentation played a significant role.

3 Goal-oriented automatic measurement in SDL based SW development

The purpose of this section is to describe the implementation of the automatic measurement process. The measurement automation process at NMP is shown in Figure 1:

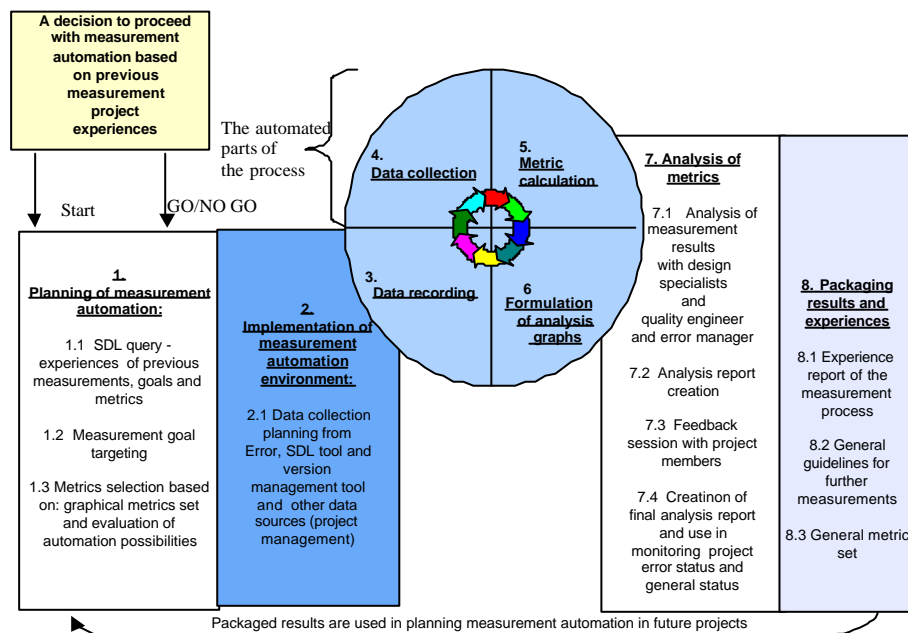


Figure 1. Measurement automation process at Nokia Mobile Phones

Figure 1 process is based on measurement automation process [10]. The process illustrated in Figure 1 will be used as a framework throughout the paper when the results of measurement automation are presented, analysed and discussed.

3.1 Planning of measurement automation

The work was started by planning the automation (Step 1, Figure 1). Instead of the GQM interviews [12], measurement experts decided to conduct a survey to initiate the SDL based measurement (Step 1.1, Figure 1). The questionnaire was directed to project members, project managers, error managers and quality engineers. The purpose was to obtain the information to define the measurement goals and metrics. The questionnaire included questions about the respondents' expectations on SDL measurement and the most suitable focus for the measurement. Experiences from the previous GQM project were used as the basis for this action. Based on the results, the measurement goals were defined (Step 1.2, Figure 1) together with measurement experts from VTT and the stakeholders from NMP.

Defining the goals is important and creates the basis for all measurement activities [17]; [10]. The project managers especially had difficulties in the following areas: 1) predicting the SDL module testing effort/time in the development projects and 2) evaluating SDL module quality in order to start integration. The main goals for the measurements were defined as follows:

1. To improve the predictability of SDL based SW development
2. To evaluate the quality of SDL based SW modules

The metrics were selected (Step 1.3, Figure 1) in a meeting together with the VTT measurement experts, NMP project managers, error managers, quality engineers and the main designer. Metrics selection was based on a set of graphical presentations of metrics, which was created based on previous project experiences, results from SDL query, literature study, and evaluation of metrics automation possibilities.

After the selection meeting, the selected metrics were evaluated and subsequently updated. These metrics were then included in the NMP case project's quality plan. Examples of the goals and metrics are shown in Table 1:

Table 1. Examples of the goals and the metrics defined for the case project

Goal	Metrics
Goal 1: Improve the predictability of the SDL based SW development in SDL projects	Metric.1 Fault status by SDL module Metric.2 Fault status by time Metric.3 Effort used in each testing phase Metric.4 Effort used in test development Metric.5 Testing time / errors found Metric.6 SDL module complexity / number of faults
Goal 2: Evaluate the Quality of SDL based SW development in SDL projects	Metric.1 Number of faults by SDL module and priority Metric.2 Number of faults by priority Metric.3 Number of faults by SDL module and error classification

Typically, the project managers will use informal assessments of critical factors during SW development to assess whether the end product is likely to meet the requirements [2]. This can be difficult when new design methods and tools are used. In our case, the predictability of the SDL based SW development was measured by the fault, complexity and effort metrics (Table 1). One purpose of the metrics was to help to estimate the remaining testing effort based on the quality, complexity, and effort used. The metrics defined were found to help the design phase, and also to reduce effort later in the coding, testing and maintenance phases. Literature has also indicated that a better design can lead to less complex code and make software easier to maintain [20].

3.2 Implementation of measurement automation environment

Collecting metrics data involves the expenditure of resources such as time, labour and money [20]. Automation saves effort, is more efficient and produces more accurate results, i.e. if the data collected is reliable [13]. The implementation of the environment for measurement automation (Step 2, Figure 1) was initiated after the selection of the measurement goals and metrics. Some automation activities were already planned as a part of the previous projects. Implementation of the automatic measurement environment was started by an analysis of metrics that could be automated. This analysis was performed based on the location and reliability of the data, and the ease of collecting and calculating the metrics. The environment was implemented by the VTT measurement experts using MetriFlame [19].

3.3 Data recording, collection, calculation and presentation

Planning the data collection (Step 2.1, Figure 1) was started by defining the data sources for the selected metrics. The main data source was the error database. This was due the fact that this database contained most of the data needed. The principal idea of automatic measurement was to facilitate the metric data collection and calculation activities using the MetriFlame. The following data sources were used:

- Fault data was collected automatically from the Error and Inspection databases.
- Status information was collected from the configuration management database.
- Complexity information was calculated using the SDL design tool data.
- Effort information was collected from project management tools.

All collected data was converted to a MetriFlame-specific format using tailored data converters. The converter for the SDL design tool was most challenging to implement. This was due to the necessity of calculating the complexity of the SDL modules, which was not readily available. After the conversion the calculation formulas and the presentation style for the resulting metrics could be created. The data collection from the different data sources is shown in Figure 2.

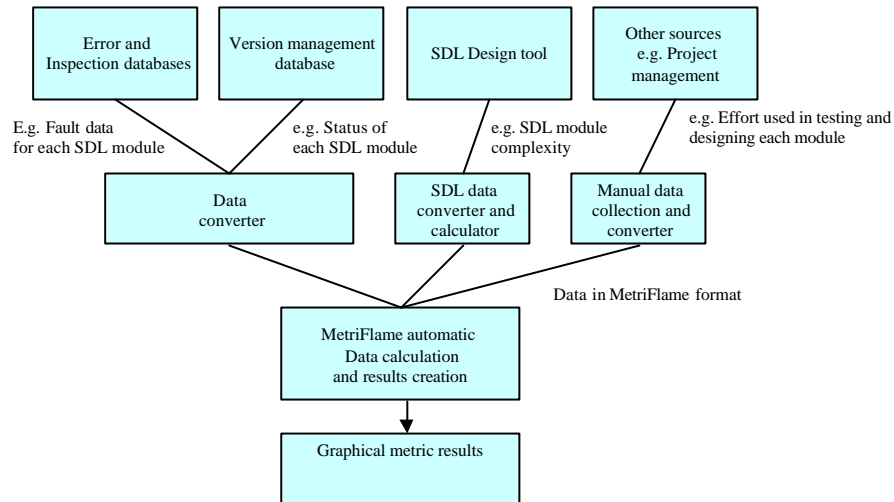


Figure 2. Data collection from the different data sources

Measurement automation does not, however, end with the automatic collection of metric data. It is continued with the calculation and the formation of the resulting analysis graphs (Steps 3-6 in Figure 1). In our case, the metrics were gathered between May and December 2000 in three case projects. The time between measurements in each project was determined based on the quantity of available data and the time-cycle was tailored to meet the specific needs of each case project.

The first metrics collections were conducted and mainly carried out by the VTT measurement experts. However, during the later cycles, also the NMP quality engineers involved in the case projects used MetriFlame to automatically collect and analyse the data.

3.4 Analysis of the metrics

Metrics analysis in feedback sessions forms a vital part of the measurement process. In the case projects, metrics were analysed every four weeks.

The results were first analysed in a session with the VTT measurement experts and the main designer, the quality engineer and the error manager from the NMP case project. During the first analysis session:

1. All collected metrics were analysed,
2. Problems and improvement ideas that arose from the results were documented,
3. Most valuable metrics were selected for more detailed analysis,
4. Notes and defects in the results were collected and documented.

After the first session, a second analysis session was organised. During the second analysis the main results, improvement ideas and problems found during the first session were presented, discussed and analysed together with the NMP project staff. Project staff had valuable knowledge about the progress of the case projects and could further interpret the main results. The feedback received during these meetings was taken into account when planning for the next measurements. The results were used in project management and SDL quality assurance activities.

3.5 Packaging the results and experiences

Packaging the results and experiences is an important part of a metric program since it enables the reuse of metrics and analysis results in future projects. As a result of packaging, a description of the metrics, a guideline for automating measurements, and an experience report were drawn up.

Information about the metrics and their analysis procedures can be reused in organisations [3]. In reality this requires, however, that the metrics and related background and analysis information are carefully documented and packaged during the measurement projects. In our case, the main purpose of the metric set document was to facilitate and speed up the measurement planning procedures. This was done by defining a set of possible measurement goals, metrics and metric analysis presentation styles for the case projects.

The guideline for automatic measurement included detailed descriptions on how to get started and how to implement automatic measurement in the organisation. Experience report included the key findings discovered during the SDL based measurement project. These reports were directed to the project quality engineers and error managers, aiming at facilitating the work of the quality manager and the person responsible for measurement in future projects.

4 Results: Experiences and lessons learned

The findings presented in this section are based on the experiences and documentation drawn up. The data was collected from the following sources: Goal definition and metrics selection workshops, measurement results feedback sessions and experience reports drawn up by the SDL measurement automation project. The key findings are shown in Table 2 and discussed later with more detail.

Table 2. Key findings for each measurement phase

Planning measurement automation
- A targeted survey facilitates the collection of ideas for measurement goals and metrics and is easy to distribute to the relevant stakeholders. The questionnaire

<p>may be reused for each metrics definition cycle.</p> <ul style="list-style-type: none"> - Prioritisation of metrics (what to collect) can be difficult since the project members may initially want to collect many kinds of metrics. This can be facilitated if: <ul style="list-style-type: none"> - Constraints for metrics automation are analysed before selecting the metrics. - Graphical examples of the metrics results are created. - A plan for measurements should be included to the project plan and quality plan.
Implementing the environment for measurement automation
<ul style="list-style-type: none"> - Measurement automation requires a lot of time, results should be repeatable. - The use of measurement tools that support automation (MetriFlame or similar) will help and speed up the implementation. - Changes in the software development environment may cause changes to the measurement automation. Predictable future changes should be taken into account.
Data recording
<ul style="list-style-type: none"> - Data correctness can be improved by defining correct input formats for data, training project staff to record data correctly and reminding project staff regularly about data correctness.
Data collection
<ul style="list-style-type: none"> - Useful data sources for SDL-specific measurement data are inspection databases, SDL design charts, error management databases, project management data and configuration management tools (i.e., SDL module versions, etc.). - The use of complementary data sources increases the value of the analysis. - Define clear goals and metrics in order to avoid problems during data conversion and calculation from different tools.
Formulation of analysis graphs and analysis of metrics
<ul style="list-style-type: none"> - In order to save effort, review results first with a small group of experts. After the first session, arrange feedback session with focused results with project staff. - Make the results available (on-line) for project managers and quality managers at minimum and preferably to all who have participated to measurement process. - Using the results without project staff analysis is very risky because the necessary background information behind the metrics may remain unknown.
Packaging results and experiences
<ul style="list-style-type: none"> - Guidelines and experience reports can often be directly used in forthcoming projects. - The process steps, roles and automation environment may need, however, to be modified to match the needs of the new project.

When starting to plan the metrics, project staff is often interested to have many kinds of metrics. Project staff may be busy and purpose of the measurements can easily be forgotten. Therefore, the active participation of managers to keep focus is required.

Metrics selection is one of the most challenging and important phases of measurement projects. Metrics must be useful also to project, not only to management. The metrics must also describe the defined measurement goals and be automatically collectable. According to our experiences, a graphical presentation is a good way to clarify and

give idea of future results. In our case, we used the results from the previous measurements and literature. The technical restrictions that affect automatic measurement should be analysed before the selection. Several requirements for the automatically collectable metrics exist, for example, the data sources must support automatic data retrieval. Early analysis of automation possibilities will reduce the amount of work later. Metrics selection requires some time from the project managers, the measurement experts and the quality managers. This is, however, worthwhile, as it guarantees that the project will collect only useful metrics, for which collection, calculation and setting the presentation format can be automated.

A good way to plan and report a measurement process is to include it as an integral part of project and quality plans. This combines measurement activities as part of the project's normal working routines and project staff does not regard it as extra work.

In practice, the preparation work for automatic measurement is one of the most time-consuming tasks when starting a measurement project. Metric data sources, calculation and schedules, and storage for the metrics' results should all be defined and implemented before actual measurement collection and analysis can start. We found out that the use of specific tool support for automatic measurement (MetriFlame in our case) makes the measurement data collection, calculation and presentation of the analysis graphs steps straightforward. In MetriFlame, we already had the mechanism for automatically calculating the metrics, creating analysis graphs, and scheduling the tasks. We only had to implement the data collection converters, which convert the data from the different data sources into a format that MetriFlame can read. However, the development of mechanisms for collecting data automatically from different sources requires time and effort from measurement experts because technical constraints of the tools that act as data sources must be understood. Thus, developing the measurement automation is a task, which should be carried out like a development project including the definition of tasks and responsibilities. However, one should remember that the measurement automation environment must only be created once, must be flexible and usable beyond one development project. When this has been done, the project staff can use the automatic tools to collect, calculate and create graphs.

It should be noted that changes in the measurement automation environment are always possible. If the metrics or tools where the data is collected are updated or changed, the changes must also be done in the measurement automation environment. These changes can be laborious or even impossible to implement. Measurement automation is most efficient in a stable working environment, but possible future changes should be predicted where feasible and the measurement tools should be made flexible enough.

Mistakes made during data recording are a very common problem. Erroneous data can easily be inserted into a database and it may lead to false measurement results. This kind of defects can be reduced in two ways:

1. By restricting the formats for recording data during the planning phase
2. By training the project members for entering the data, and by reminding project staff about the data correctness during the development project.

Collected metric data is dependent on the measurement goals and selected metrics. In our case, the data was collected from several sources like the SDL design tool, error and inspection databases and also some data was collected manually. During SDL based measurement, we found out that it is possible to automatically collect measurement data related to SDL based SW development process from various data sources. The SW development tools, processes and predefined metrics that could be used in measuring an SDL based SW development process are shown in Table 3:

Table 3. Examples of the SDL based measurement tools and metrics

Tool	Example metrics	Notes
Inspection database	- Inspected diagrams vs. all diagrams. For each SDL module: - Number of defects - Defect severity and type	Inspection tools contain useful information of inspected SDL and MSC diagrams, including the number and types of defects found. This data gave us a possibility to analyse the quality of the SDL modules in SW development.
SDL design	For each SDL module: - SDL design chart complexity, size, performance	We found design information useful in evaluating SDL module complexity. The complexity metrics were integrated with data describing faults and effort.
Error database	For each SDL module: - Number of faults found during testing - Type and severity of faults found during testing	Error databases were found out to be very useful from the automatic measurement viewpoint. Useful information about the tested SDL modules was obtained, such as the number and types of faults reported.
Project management	For each SDL module: - Planned effort vs. actual effort	Effort information can be collected from a project management tool. One condition for collecting metrics was that effort data must be entered separately for each SDL module.
Configuration management	For each SDL module: - Number of created versions	Configuration management tools contain useful background information for measurement data analysis. The number of versions gave us useful information for the effort analysis.

Combined information from different sources is much more useful than information from single source alone. It also makes integration of results from different sources possible. We also found out that measurement goals and measurement automation

possibilities are affected by the tools and the processes, during which the data is collected. In our case, the main goal was to improve the predictability and ensure the quality of SDL based SW development.

Feedback from the project staff has a critical impact on the success of measurement. Due to large amount of metrics and graphs produced automatically, the most important and critical results were selected for the project staff feedback session. First metrics analysis took two hours from the small group of measurement experts and project/quality managers. This preliminary work decreased the time required for the feedback sessions where the project staff was present. The feedback sessions of the most important and critical results took only 1.5 hours from the project members each month. These feedback sessions coupled to project meetings, which also decreased the time used in arranging the meetings.

Up-to-date metric results should be available for project managers and error managers in a pre-defined location. Automatically collected, regularly analysed and updated metric results are a useful way of monitoring project status and defects and predicting the remaining effort and defects. It is, however, important to remember that using the metric results without project staff feedback is very risky. Without feedback from the project, the managers may draw conclusions based on fault results.

Packaging the results and experiences is useful because it summarises the work done on automatic measurement and enables reuse of the results. The most important metrics were used also in later SW development projects at NMP. The guidelines for automatic measurement and the experience report have also been used when starting new SW development projects. However, the measurement process steps and roles were modified to correspond with the needs of the new projects.

5 Conclusions

While metrics programs are challenging to implement, literature has shown that automated software measurement has a significant role when adopting software metrics into systematic use. Many reports show how measurement has been successfully used in improving quality and productivity. Automating measurement is recognised as an important factor especially in a situation where strict time-to-market requirements leave little time for the project staff to introduce new methods and analyse their impact. This paper has discussed the processes and key findings from systematic automation of SDL based measurement in three case projects at Nokia Mobile Phones.

As a result of the automatic measurement project the product projects gained a set of metrics they found very useful. These could be collected automatically from the metrics databases. The result graphs were regularly analysed in the project meetings and used to facilitate project management.

The key findings discussed in this paper were:

- Metrics selection is one of the most important task in measurement automation. However, the prioritisation of metrics (i.e., what to collect) can be difficult since the project members may initially want to collect all possible metrics. Metrics selection can be facilitated by analysing constraints for metrics automation before selecting the metrics and creating graphical examples of the metrics.
- Changes in the software development environment may cause changes to the measurement automation environment, these changes can be laborious to implement. Thus, measurement automation is most efficient in a stable working environment.
- The useful data sources for SDL-specific measurement data are inspection databases, SDL design charts, error management databases, project management data and configuration management tools (i.e., SDL module versions, etc.). Using different data sources increases the reliability of the metrics analysis.
- The metrics and the results of their analysis should be made available on-line at minimum for project managers and quality engineers. It is suggested, however, to make the data available also to individual project members. Using the metrics' results without analysis is very risky because the necessary background information behind the metrics may remain unknown.

It should be reminded that these findings are based on only three measurement case projects. However, the results were obtained over a period of two years, which mitigates the limited number of projects. It should also be noted that these findings are based on analysis carried out during the start-up phase of measurement automation. The processes and findings can be different if measurement automation was more established.

During this measurement automation project the importance of metrics became evident. Planning the metrics and implementing the metrics collection may be time-consuming and hard work, but the results are rewarding. After the measurement automation project described here, several measurement activities have started in Nokia Mobile Phones projects. As one result, MetriFlame was not chosen as an automatic measurement tool at NMP. MetriFlame wasn't compatible enough with other tools and it was therefore used as a reference when improving measurement automation. Some of the final results - the most important metrics, the guidelines and experience reports - have been used in the initiation of new SW development projects. Also, the process steps, roles and automation environments have been tailored for use in many Nokia Mobile Phones projects.

6 Acknowledgements

The authors would like to thank all people at NMP and VTT who reviewed the paper. Special thanks to Dr. Pekka Abrahamsson for his valuable comments.

7 References

- [1] Basili, V. R. and Rombach D.(1988). "The Tame Project: Towards Improvement-Oriented Software Environments." IEEE Transactions on software engineering 14(6).
- [2] Fenton, N., Krause P. and Neil M. (2002). "Software Measurement:Uncertainly and Causal Modelling." IEEE Software 19: 116-122.
- [3] Fugetta, A., Lavazza, L. Morasca, S. Cinti, G. Oldano and Orazi E. (1998). "Applying GQM in an industrial software factory." ACM Transactions on Software Engineering and Methodology 7(4): 411-448.
- [4] Gopal, A., Krishnan, M. S. Mukhopadhyay T. Goldenson D. R. (2002). "Measurement Programs in Software Development: Determinants of Success." IEEE Transactions on software engineering 28(9): 863-875
- [5] Hall, T. and Fenton N. (1997). "Implementing Effective Software Metrics Programs". IEEE Software 14: 55-65.
- [6] Herbsleb, J. D. and Grinter R. E. (1998). Conceptual simplicity meets organizational complexity: case study of a corporate metrics program. Proceedings of the 1998 International Conference on Software Engineering.
- [7] Humphrey, W. S. (1989). Managing the Software Process. Reading, Mass., Addison-Wesley.
- [8] International Engineering Consortium (2002). "Specification and Description Language (SDL).", <http://www.iec.org/online/tutorials/sdl/topic02.html> (available 18.9.2002).
- [9] Jong, G. (2002). UML-Based Methodology for Real Time and Embedded systems design. Automation and Test in Europe Conference, Paris.
- [10] Komi-Sirviö, S.,Parviainen P. and Ronkainen J.(2001). Measurement Automation: methodological background and practical solutions - a multiple case study. Software Metrics conference, London, McGraw-Hill Publishing Company.
- [11] Laitinen, M. and Fayad M. E. (1998). "Surviving a process performance crash." Communications of the ACM 41(2): 83-86.
- [12] Latum, F., Solingen R. , Oivo, M. Rombach D. Ruhe G. (1998). "Adopting GQM Based Measurement in an industrial environment." IEEE Software 15: 78-86.
- [13] Lavazza, L. (2000). "Providing Automated Support for GQM measurement process." IEEE Software 17(3): 56-60.
- [14] Niemelä, E., Kuikka, S. Vikuna, K. Lampola, M. Ahonen, J. Forssel, M. Korhonen, R. Seppänen V. and Ventä O. (2000). Teolliset komponenttiosjelmistot -Kehittämistarpeet ja toimenpide- ehdotukset. Helsinki, Tekes.
- [15] Pfleeger, S. L. (1993). "Lessons learned in building a corporate metrics program." IEEE Software 10: 67-74.

- [16] SDL Forum Society (2002). "What is SDL?", <http://www.sdl-forum.org/SDL/index.htm> (available 19.8.2002).
- [17] Solingen, R. and Berghout E. (2001). Integrating goal-oriented measurement in industrial software engineering: industrial experiences with additions to the Goal/Question/Metrics (GQM) method. Metrics 2001, London, McGraw-Hill publishing company.
- [18] Vierimaa M., Kaikkonen T., Oivo M., Moberg M. (1998) Experiences of practical process improvement, Embedded Systems Programming Europe. Vol. 2 (1998) No: 13, 10 - 20.
- [19] VTT, Technical Research Centre of Finland (2002). "MetriFlame www-pages." <http://www.vtt.fi/ele/research/soh/products/metriflame/index.html> (available 18.9.2002)
- [20] Zage, W. and Zage D. (1993). "Evaluating Design metrics on large scale software." IEEE Software 10: 75-81.
- [21] Zahran, S. (1998). Software process improvement: practical guidelines for business success. Reading, Mass., Addison-Wesley Pub. Co.