An Example of Using Key Performance Indicators for Software Development Process Efficiency Evaluation

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 possible Key Performance Indicators (KPI) that can be used for software process efficiency evaluation. The overview is based on currently used KPIs in software development projects on CPP platform. The most important KPIs are analyzed, and their usage in the process efficiency evaluation is discussed. The outcome of the measurement is used to initiate further process adjustments and improvements. In addition, there is possibility to perform benchmarking between different development projects, and based on collected data easier search for best practices in the projects that can be broadly implemented. Some proposals and future directions in the area of process measurement are given.

I. INTRODUCTION

All successful software organizations implement measurement as part of their day-to-day management and technical activities. Measurement provides the objective information they need to make informed decisions that positively impact their business and engineering performance. In successful software organizations, measurement-derived information is treated as an important resource and is made available to decision makers throughout all levels of management.

The way measurement is actually implemented and used in a software organization determines how much value is realized in terms of business and engineering performance. Measurement is most effective when implemented in support of an organization's business and technical objectives and when integrated with the existing technical and management activities that define a software project. Measurement works best when it provides objective information related to the risks and problems that may impact a project's defined objectives. In other words, measurement works best when it is considered a significant, integral part of project management [1].

Top-performing organizations design their technical and management processes to make use of objective measurement data. Measurement data and associated analysis results support both short and long-term decision making. A mature software development organization typically uses measurement to help plan and evaluate a proposed software project, to objectively track actual performance against planned objectives, to guide software process improvement decisions and investments, and to help assess overall business and technical performance against market-driven requirements. A top-performing organization uses measurement across the entire life cycle of a software project, from inception to retirement. Measurement is implemented as a proactive discipline, and measurement derived information is considered to be a strategic resource.

Measurement is most important at the project level. Software measurement helps the project manager do a better job. It helps to define and implement more realistic plans, to properly allocate scarce resources to put those plans into place, and to accurately monitor progress and performance against those plans. Software measurement provides the information required to make key project decisions and to take appropriate action. Measurement helps to relate and integrate the information derived from other project and technical management disciplines. In effect, it allows the software project manager to make decisions using objective information.

In this article, the overview of process measurement in software development projects on CPP platform will be given, and some Key Performance Indicators (KPIs) will be discussed. Also, one example of project benchmarking will be presented. At the end, some improvement proposals, and directions for further work will be given.

II. ISO/IEC 15939 Software Measurement Process

The International Standard ISO/IEC 15939 identifies the activities and tasks that are necessary to successfully identify, define, select, apply, and improve software measurement within an overall project or organizational measurement structure. It also provides definitions for measurement terms commonly used within the software industry [2]. The software measurement process itself is shown on Fig. 1.

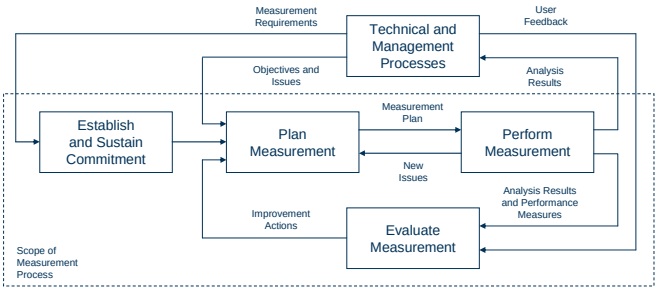

Fig. 1. ISO/IEC 15939 Software Measurement Process

III. CMMI Process Area Measurement and Analysis

According to Capability Maturity Model Integration (CMMI), the purpose of Measurement and Analysis is to develop and sustain a measurement capability that is used to support management information needs [3].

The Measurement and Analysis process area involves the following:

- Specifying the objectives of measurement and analysis such that they are aligned with identified information needs and objectives;
- Specifying the measures, data collection and storage mechanisms, analysis techniques, and reporting and feedback mechanisms;
- Implementing the collection, storage, analysis, and reporting of the data;
- Providing objective results that can be used in making informed decisions, and taking appropriate corrective actions.

The integration of measurement and analysis activities into the processes of the project supports the following:

- Objective planning and estimating;
- Tracking actual performance against established plans and objectives;
- Identifying and resolving process-related issues;
- Providing a basis for incorporating measurement into additional processes in the future.

The initial focus for measurement activities is at the project level. However, a measurement capability may prove useful for addressing organization wide information needs. Projects may choose to store project-specific data and results in a project-specific repository. When data are shared more widely across projects, the data may reside in the organization's measurement repository.

The Measurement and Analysis contexts according to CMMI model is shown on Fig. 2.

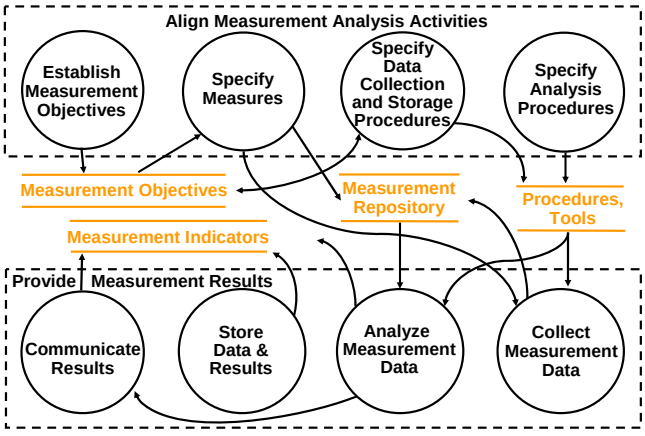

Fig. 2. Measurement and Analysis Context

IV. Data Collection Model

All projects have specific objectives that are typically defined in terms of system capability, resource budgets, milestones, quality, and business or system performance targets. Project success depends largely on how well these objectives are achieved. Project issues are areas of concern that may impact the achievement of a project objective: risks, problems, and lack of information, for example.

The most information needs in one project can be grouped into general areas, called information categories. We can identify seven information categories, which represent key areas of concern for the project manager [4]:

- Schedule and Progress;
- Resources and Cost;
- Product Size and Stability;
- Product Quality;
- Process Performance;
- Technology Effectiveness;
- Customer Satisfaction.

The information about project performance is collected in cycles. The typical data collection cycle is four weeks. Based on achieved results, product supplier (software development project) performs analysis and defines operation excellence action plan within two weeks time frame. When actions are established, the measurement results and operational excellence action plans are ready for presentation on Operating Steering Group (OSG) for the project. Results form the all projects and OSG meetings are input for R&D Center Steering Group meeting, organized each quarter [5].

The data collection process is shown on Fig. 3.

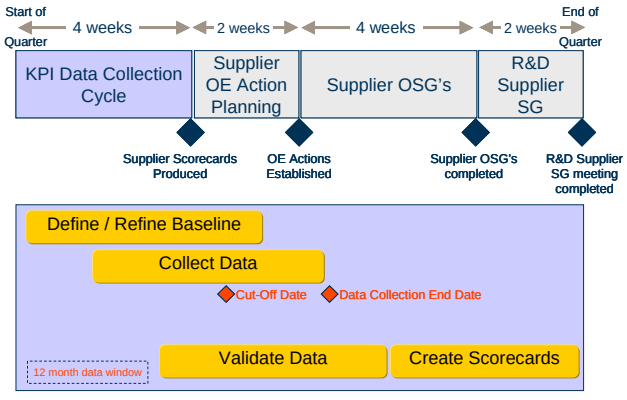

Fig. 3. Overview of data collection process

The measurement of project performance gives organization increased opportunities to improve and share good practices, and to increase the possibility to reach wanted operational efficiency. The measurement activities responsibility is on the corporate R&D level. This responsibility covers forum, processes, tools, definitions of metrics, collections, analyzing and reporting on corporate level.

V. KPI Definitions for CPP Development Projects

In order to follow performance of CPP software development projects, we have defined set of KPIs. Some of KPIs are applicable for early project development phases, some of them for complete life cycle, and some of them only for product maintenance.

The set of KPIs, their behavior and applicability are shown on Fig. 4 [6].

For each KPI we have defined:

- Description;
- Results format;
- Formula;
- Frequency.

A. Schedule Adherence

Definition:

Measures timeliness and 'quality' of deliveries relative to baseline schedule and acceptance criteria. Based on percentage deviation between planned and actual lead times [7].

Result format:

Reported as a percentage, 100% is the highest result.

Formula:

$$[1 - ABS(ALT - PLT) / PLT] \times 100$$

PLT = Planned Start Date – Planned Finish Date

ALT = Actual Finish Date – Planned Start Date

If no planned start date is specified for intermediate or parallel deliverables, the earliest planned start date (e.g. TG2 or assignment start date) may be used.

Planned Start/Finish Dates are replaced with revised dates in case of Ericsson caused/mandated CRs.

B. Assignment Content Adherence

Definition:

Measures supplier's ability to deliver full assignment scope by end of assignment. It is based on percentage of completed functionality/requirements [7].

Result format:

Reported as a percentage, 100% is the highest result.

Formula:

$$(No.\ of\ Compl.\ Req.\ /\ No.\ of\ Commit.\ Req.) \times 100$$

Requirements are smallest measurable 'packages' of functionality; e.g. features, documents, items in Statement of Compliance, Requirement Specification, Requirement Management Tool, or Implementation Proposal.

Number of Completed Requirements counts packages of functionality delivered during the entire assignment.

Total Number of Committed Requirements counts the packages of functionality originally planned for the assignment; may be revised based on Change Request guidelines.

Frequency:

Measured and reported at the end of an assignment.

KPI measurement has to be based on requirements that are the smallest objects of measurement and easily measurable. For example, content adherence for an assignment with 2 major deliveries should not be based at the 'delivery level' but rather based at the core functionalities/requirements within each delivery. Assignments where scope is not frozen at TG2 (Project GO decision) need to handle scope additions through the CR handling guidelines.

Performance Metric	KPIs	Desired Behaviour	Assignment Phase			
			Pre-TG2 and RXI	*Development (TG2-PRA)*	*Design Follow-up (Maintenance PRA to GA)*	*Maintenance (Post GA)*
Time	Schedule Adherence	▪ Delivery to Commitment ▪ Accurate Estimating ▪ Avoid Delaying Scope Until Later Assignments	✓	✓	●	●
Content	Overall Assignment Content Adherence	▪ Delivery to Commitment ▪ Delivery of Full Scope	✓	✓	●	●
Cost	Cost Adherence	▪ Delivery to Commitment ▪ Accurate Estimating ▪ Avoidance of Buffers	✓	✓	●	●
Quality	Fault Slip Through (FST)	▪ Quality of Supplier Deliverables to Ericsson ▪ Delivery of fault free deliverables to Ericsson	✗	✓	✗	✗
Service Levels	TR Closure Rate	▪ TR response to Commitment	✗	✗	✓	✓
Cost of Quality	Cost per TR (CTR)	▪ Efficiency in resolving quality issues ▪ Reduction in cost of poor quality	✗	✗	✓	✓

Key: ✓ KPI Applicable to Assignment ✗ KPI N/A to Assignment ● May be applicable (decided case by case)

Fig. 4. KPI Definitions for CPP projects

C. Cost Adherence

Definition:

Measures supplier's ability to deliver assignment scope within the agreed/committed cost, including man-hour, lab and travel costs. Based on deviation between committed (baseline) and expected (actual + forecast) costs at assignment/deliverable level [7].

Result format:

Reported as a percentage, 100% is the highest result.

Formula:

$$[1 - (ECost - CCost) / CCost] \times 100\%$$

Committed cost is the baseline at assignment start. Contingency value (buffer) should be specified separately, if known.

Expected Cost to Complete is (actual + forecast) each month:

- Actual costs incurred so far;
- Forecast of all remaining Costs to Complete;
- Forecast of contingency sums (optional).

Delivering an Assignment under the Committed Costs will have neutral impact on the KPI. Aim is to discourage unnecessarily using budgeted hours;

Frequency:

Measured monthly at assignment level, or at end of each major deliverable.

Costs have to be defined at assignment level (mandatory), and optionally (if possible) at deliverable level, to enable precise change control.

D. Fault Slip Through

Definition:

Measures supplier's ability to capture faults before making deliveries to I&V>

- Assuming that supplier conducts Function Testing (FT);
- Supplier or external organization may conduct I&V (Integration and Verification) Testing.

Based on Trouble Report (TR) slippage between FT and I&V test phases.

- Assuming that TRs are analyzed to identify 'true' slipped TRs;
- If TRs are not analyzed, then 0% may not be the expected best result due to the different scope in FT and I&V testing [7].

Result format:

Reported as a percentage, 0% is the lowest result.

Formula:

$$[1 - FT\ Faults / All\ Faults] \times 100\%$$

Faults are classified as FT or I&V based on testing phase, not who does the testing. All parties conducting the testing need to capture the Function Test and I&V Faults, based on assignment TR Handling guidelines/tools.

Frequency:

Monthly from start to end of I&V (cumulative data collecting); or at each 'drop' on completion of the respective I&V.

TRs that do not relate to 'genuine' faults, i.e. cancelled, postponed, duplicated, and rejected TRs, are to be excluded. All 'minor' faults, faults that do not affect the main operation of the system are to be excluded.

E. Trouble Report Closure Rate

Definition:

Measures supplier's ability to answer TRs within the specified goals. It is based on deviation between the actual TR answering times and TR goals, set by the Assignment Owner [7].

Result format:

Reported as lost days, averaged across TR priority. The lowest result is 0, indicating that the TRs are answered within the goals.

Formula:

$$NLD / (OTR + NTR)$$

NLD = number of lost days within the time increment for all open and new TRs

OTR = number of open TRs at beginning of the time increment

NTR = number of new TRs during time increment

The TR handling time starts at the point at which the TR enters the supplier organization, and ends at the point at which the TR is answered.

Time increment is typically 12 months in the past from reporting date.

Frequency:

Measurement is done on a monthly basis.

F. Cost per Trouble Report

Definition:

Measures supplier's efficiency in fixing TRs (answer plus solution), i.e. maintenance costs relative to TRs resolved, in man-hours [7].

Result format:

Reported as man-hours.

Formula:

$$Cost\ of\ Maintenance / Number\ of\ TRs\ Resolved$$

Cost of Maintenance activities is total hours spent on TR Handling activities.

Number of TRs resolved are TRs that include a fix/solution.

The result is expressed as a rolling average, over past 12 months from the current reporting date, across all product areas in maintenance.

Frequency:

Measurement is done on a monthly basis.

VI. Project Benchmarking

Benchmark office measures and analyzes the development unit performance in order to improve their Operational Efficiency, for example by good practice sharing across organization. The measurements are focused towards the project perspectives of the development unit performance, and possibilities to make external or internal benchmarking possible. The Benchmark office supports the development unit steering in analyzing their operational and process efficiency and improvements. The Benchmark Office is responsible for the process, definitions and tools, as well as performing analysis on corporate level.

The one example of CPP project benchmarking is shown on Fig. 5. It can be seen from the figure that most of the KPIs are on the commitment or stretched level. That means project has fulfilled its goals.

The project marked with D1 is the oldest measured project, and it has the lowest achieved results. The successor projects have performed much better. That is achieved by performing the root cause analysis of the KPIs. The analysis has resulted with corrective and preventive actions in the next projects, and positive result is visible.

The project marked with D2 had problems with budget (visible from Cost Adherence KPI). Detailed analysis shown that initial estimations were too optimistic, and 3rd party supplier part of the project has spent much more than it was planned.

The benchmarking itself has no intention to initiate only the competition between projects and organization. The full benefit can be achieved if results are deeply analyzed, and preventive and corrective actions are set for the ongoing and future projects (learning from experience).

VII. Improvements and Future Directions

The set of KPIs described in this article is the basic set, established 18 months ago. We are today in the position where we have enough measurement results to perform precise analysis. But, it is obvious that this is not complete list of KPIs that can be measured in the software development project. Many other interesting data can be collected.

Two product life cycle phases are the most important for new, more advanced KPIs and measurements in the future; verification phase and maintenance phase.

In the verification phase of the project we have to measure how efficient our verification activities are. It is not enough to measure number of executed test cases, and pass rate. These measurements are not telling us much about expected product quality. The idea is to establish fault rate measurement. The fault rate measures how many faults we discover in certain time interval (typically one week). If fault rate decrease with time, that means product quality is improving by performing test activities. Additionally, we can set lower fault rate limit, in order to plan how long we will go with our testing, and when we can stop with testing assuming that product has reached expected quality level.

In the maintenance phase of the life cycle we would like to measure product maintenance cost compared with development effort. At the moment we know the average cost to remove the fault. According to this measurement it is difficult to compare quality level for two different products. The new KPI can measure total product maintenance cost in the first year of operation, and compare it with total development cost. The result can be expressed as percentage of product development cost. With this measurement we will be able to compare quality level for different products in the maintenance phase of the life cycle.

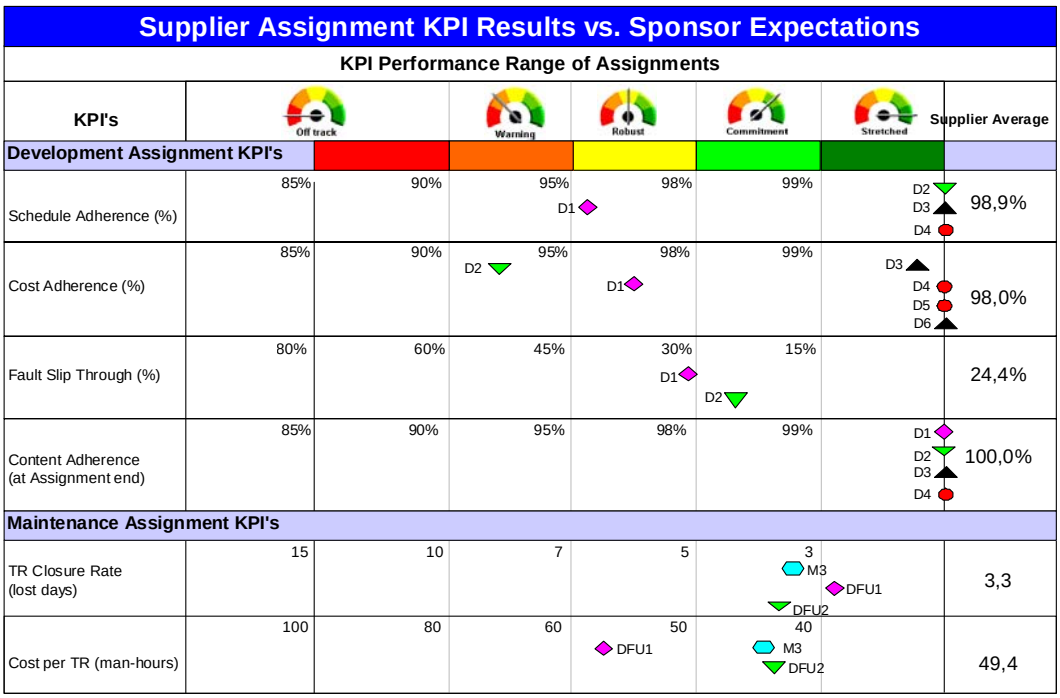

Fig. 5. CPP Project Benchmarking

VIII. Conclusion

The measurement method and KPIs described in this article are coming from the CPP software development projects at Ericsson. We have started this process as measurement program, and have implemented in all development projects from the end of year 2006. The data were collected and analyzed on monthly basis, and used as input for further improvement activities in the development projects.

The measurement process should be an integral part of the way business is conducted. Data must be provided early enough to allow management to take actions. Results must be communicated throughout the organization in a timely manner. Decisions should not wait for perfect data, but should be based on accurate data, supported by risk management and root cause analysis.

Both the measurement process and the specific KPIs should be periodically evaluated and improved. Measurement is an iterative process; the KPIs are refined as information needs change and the organization implements improvement actions.

In the future, we can expect more demands on software product quality, reduced project lead-time, and reduced project budgets. The possible answer on these demands is to always have accurate data about project and product performance, and fast improvement programs, preventive and corrective actions based on analysis of key performance indicators in the project.

References

[1] J.McGarry, "Measurement Key Concepts and Practices," Practical Software & System Measurements, USA, 2003.

[2] ***, "Systems and Software Engineering - Measurement Process", International Organization for Standardization, Geneva, 2002.

[3] M.B.Chrissis, M.Konrad, S.Shrum, "Capability Maturity Model Integration – Guidelines for Process Integration and Product Improvement", Pearson Education Inc., Boston, 2004.

[4] D.Ishigaki, C.Jones, "Practical Measurement in the Rational Unified Process", Rational Software, USA, 2003.

[5] ***, "Data Collection Process 2007 – R&D Consultancy Supplier Benchmark", Internal Ericsson documentation, Stockholm, Sweden, 2007.

[6] Z.Antolic, "CPP KPI Measurements 2008", Internal Ericsson Documentation, Zagreb, Croatia, 2004.

[7] C.Braf, "KPI Definitions and Guidelines", Internal Ericsson Documentation, Stockholm, Sweden, 2006.