# Metrics in Distributed Product Development

Maarit Tihinen and Päivi Parviainen
Software Technologies
VTT Technical Research Centre of Finland
maarit.tihinen@vtt.fi
paivi.parviainen@vtt.fi

Rob Kommeren
Digital Systems & Technology
Philips,
The Netherlands
r.c.kommeren@philips.com

Jim Rotherham
Project Management Office
Symbio,
Finland
jim.rotherham@symbio.com

*Abstract*— **Nowadays the products are increasingly developed globally in collaboration between subcontractors, third party suppliers and in-house developers. However, management of a distributed product development project is proven to be more challenging and complicated than traditional one-site development. From the viewpoint of project management, the measurements and metrics are important activities for successful product development. This paper is focused on describing a set of metrics that is successfully used in industrial practice in distributed product development. Based on the experiences, the reasoning for selecting these metrics was similar: they are easy to capture and can be quickly calculated and analysed on a regular interval. One of the most important reasons for choosing these metrics was that they were aimed especially to provide early warning signals, i.e., means to proactively react to potential issues in the project. This is especially important in distributed projects, where specific means to track project status are needed.**

*Keywords-metrics; measurements; global software development; distributed product development*

## I. INTRODUCTION

Globally distributed software development enables product development to take place independently of the geographical location of the individuals or organizations. In fact, nowadays the products are increasingly developed globally in collaboration between subcontractors, third party suppliers and in-house developers [1]. In practice distributed projects struggle with the same problems than single-site projects including problems related to managing quality, schedule and cost. Distribution only makes it even harder to handle and control these problems [2][3][4][5]. These challenges are caused by various issues, for example, less communication – especially informal communication – caused by distance between partners, and differences in background knowledge of the partners. That's why, in distributed projects the systematic monitoring and reporting of the project work is especially important, and measurement and metrics are an important means to do that effectively.

Management of a distributed product development project is more challenging than traditional development [6]. Based on an industrial survey [7], one of the most important topics in the project management in distributed software development is detailed project planning and control during the project. In global software development (GSD), this includes, e.g., dividing work by sites into sub-projects, clearly defined responsibilities, dependencies and timetables, along with regular meetings and status monitoring.

The main purpose of measurements and metrics in software production is to create means for monitoring and controlling and this way to provide support for decision making [8]. Traditionally, the software metrics are divided into process, product and resource metrics [9]. In the comprehensive measurement program, all these dimensions should be taken into consideration while interpreting measurement results, otherwise, the interpretation may lead to wrong decisions or incorrect actions. Successful measurement program can prove to be an effective tool for keeping on top of development effort, especially, for large distributed projects [10]. However, many problems and challenges have been identified that reduce and may even eliminate all interests to the measurements. For example, not enough time is allocated for measuring and metrics during a project, or not enough benefit is visibly gained by the project doing the measurement work (e.g., data is useful only at the end of project, not during the project). In addition, the "metric enthusiasts" may define too many metrics making it too time consuming. Thus, it's beneficial [10] to define core metrics to collect across all projects to provide benchmarking data for projects, and to build on measures that come naturally out of existing processes and tools.

This paper is focused on describing a metrics set that are successfully used in distributed product development. The main purpose of the paper is to offer a set of essential metrics with experiences of their use. The amount of the metrics is knowingly kept as limited as possible. Also, the metrics should be such, that they provide online information during the projects, in order to enable fast reaction to potential problems during the project. The metrics and experience presented in the paper are based on metrics programs of two companies, Philips and Symbio. Royal Philips Electronics is a global company providing healthcare, consumer life-style and lighting products and services. Digital Systems & Technology is a unit within Philips Research that develops first of a kind products in the area of healthcare, well-being and lifestyle. The projects follow a defined process and are usually distributed over sites and/or use subcontractors as part of product development. Symbio Services Oy provides tailored services to organizations seeking to build tomorrow's technologies. Well-versed in a variety of software development methodologies and testing best practices, Symbio's specialized approaches and proprietary processes begin with product design and stem through globalization, maintenance and support. Symbio has built a team of worldwide specialists that focus on critical areas of the product development lifecycle. Currently Symbio employs around 1400 people and their project execution is distributed between sites in the US, Sweden, Finland and China.

The paper is structured as follows. Firstly, an overview of related work – literature studies and their limitations related to measurements and metrics of distributed product development – is introduced in Section II. Then, proposed metrics are presented using Rational Unified Process (RUP) [11] approach as a framework. After that, industrial experiences of using the metrics are discussed. Finally, the conclusions are drawn in Section V.

## II. Measurements in GSD

There are several papers that discuss globally distributed software engineering and its challenges, for example, [10], [12] and [13]. Also, metrics in general and for specific aspects have been discussed in numerous papers and books for decades. However, little global software development (GSD) literature has focused on metrics and measurements or even discusses the topic. Da Silva et al. [6] report similar conclusion based on analysis of DSD literature published during 1999 – 2009: they state as one of their key finding that the "vast majority of the reported studies show only qualitative data about the effect of best practices, models, and tools on solving the challenges of distributed software development (DSD) project management. In other words, our findings indicate that strong (quantitative) evidence about the effect of using best practices, models, and tools in DSD projects is still scarce in the literature."

The papers that have discussed some metrics for GSD usually focus on some specific aspect, for example, Korhonen and Salo [13], discuss quality metrics to support defect management process in a multi-site organization. Simmons and Ma [14] discuss a software engineering expert system (SEES) tool where the software professional can gather metrics from CASE tool databases to reconstruct all activities in a software project from project initiation to project termination. Misra [15] presents a cognitive weight complexity metric (CWCM) for unit testing in a global software development environment. Lotlikar et al. [16] propose a framework for global project management and governance including some metrics with main aim to support work allocation to various sites. Peixoto et al. [12] discuss effort estimation in global software development, and one of their conclusions is that "GSD projects are using all kinds of estimation techniques and none of them is being consider as proper to be used in all cases that it has been used", meaning, that there is no established technique for GSD projects.

Some effort has also been invested in defining how to measure success of GSD projects [17], and these metrics mainly focus on cost related metrics and are done after project completion. The focus of this paper is to discuss metrics for monitoring ongoing GSD projects and that way identify needs for corrective actions early.

### A. *Traditional metrics and project characteristics*

Software measurements and metrics have been discussed since 1960's. The metrics have been classified many different ways, for example, they can be divided into basic and additional metrics [18] where basic metrics are size, effort, schedule and defects, and the additional metrics are typically metrics that are calculated or annexed from basic metrics (e.g., productivity = software size per used effort). The metrics can be divided also into objective or subjective metrics [18]. The objective metrics are easily quantified and measured, examples including size and effort, while the subjective metrics include less quantifiable data such as quality attitudes (e.g., excellent, good, fair, poor). An example of the subjective metrics is customer satisfaction. Furthermore, software metrics can be classified according to the entities of product, processes and resources [9]. Example metrics of product entities are size, complexity, reusability and maintainability. Example metrics of process entities are effort, time, number of requirements changes, number of specification/coding faults found and cost. Furthermore, examples of resource entities are age, price, size, maturity, standardization certification, memory size or reliability. These classifications, various viewpoints and the amount of examples merely prove how difficult the selection of metrics really can be during the project.

In addition to different ways of metrics classification, development projects can also be classified. Typically, the project classification is used as a baseline for further interpretation of the metrics and measurements. For example, all kind of predictions or comparison should be done within the same kind of development projects, or the differences should be taken into account. Traditional project characteristics are, e.g., size and duration of a project, type of a project (development, maintenance, operational lifetime etc.), project position (contractor, subcontractor, internal development etc.), type of software (hardware-related software development, application software, etc.) or used software development approaches (agile, open source, scrum, spiral-model, test driven development, model-driven development, V-model, waterfall model etc.). Furthermore, different phases of development projects have to be taken consideration while analyzing gathered measurement data.

### B. *Metrics and measurements during product development*

A phase of lifecycle of development project affects to the interpretation of the metrics. Thus, in this paper, proposed metrics are introduced by using commonly known approach of software development Rational Unified Process (RUP). RUP is a process that provides a disciplined approach to assigning tasks and responsibilities within a development organization. Its goal is to ensure the production of high-quality software that meets the needs of its end-users, within a predictable schedule and budget [11].

The software lifecycle is divided into cycles, each cycle working on a new generation of the product. RUP divides one development cycle in four consecutive phases [11]: 1) inception phase, 2) elaboration phase, 3) construction phase and 4) transition phase. Furthermore, there can be one or more iterations within each phase during the software generation. The phases and iterations of RUP approach are illustrated in following Figure 1.

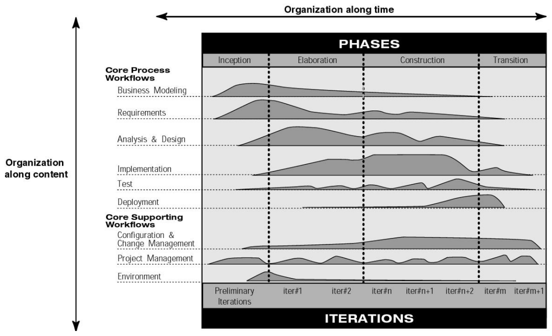

Figure 1. Phases and Iterations of RUP approach [11].

From a technical perspective the software development is seen as a succession of iterations, through which the software under development evolves incrementally [11]. From measurement perspective this means that some metrics can be focused on one or two phases of the development cycle, and some can be continuous metrics that can be measured in all phases, and can be analysed, e.g., in iterations.

### C. Measurements and metrics in GSD

Software measurement is defined by [19] as follows: "The software measurements is the continuous process of defining, collecting and analysing data on the software development process and its products in order to understand and control the process and its products and to supply meaningful information to improve that process and its products". In the daily software development work, the measurements are still seen as unfamiliar or even an extra burden for projects. For example, project managers feel it as time consuming to collect metrics for the organization (e.g., business-goal-related metrics) while they need to have metrics that are relevant to the project. Furthermore, they have impressed that there has not been budgeted enough time for measurements, and that's why it's really difficult to get approval from stakeholders for this kind of work [10].

Globally distributed development generates new challenges and difficulties for the measurements. For example, the gathering of the measurements data can be problematic because of different development tools or their versions, work practices with related concepts can vary by project stakeholders or reliability of the gathered data can vary due to cultural differences, especially, in subjective evaluations. In addition, distributed projects are often so unique (e.g., product domain and hardware-software balance vary, or different subcontractors are used in different phases of the project) that their comparison is impossible. Thus, the interpretation of measurements data is more complicated in GSD than one site projects. That's why it's recommended to select moderate amount of metrics. In this paper we will present a set of metrics to use during GSD. Also industrial experiences about the metrics will be discussed.

The common metrics (effort, size, schedule etc.) are also applicable for GSD projects. However, special attention may be needed in training the metrics collection, to ensure common understanding of them (e.g., used classifications). Also, as measurements also tend to guide people's behavior, it's important to ensure that all are aware of the purpose of the metrics (i.e., not to measure individual performance), specifically in projects distributed over different cultures.

## III. EXAMPLES OF INDUSTRIAL PRACTICES

In this Section the metric set used in the companies is introduced. The metrics are introduced according to the RUP phases where the metric is seen most relevant to measure. For each metric, a name, a notation and a detailed definition is introduced. The main goal is to offer a useful, yet a reasonable amount of metrics, for supporting the on-time monitoring of the GSD projects. Thus, the indicators are supposed to be leading indicators rather than lagging indicators, for example, planned / actual schedule measurements should be implemented as milestone trend analysis: measure the slip in the first milestone and predict the consequences for the other milestones and project end.

### A. Metrics for Inception Phase

During the inception phase, the project scope has to be defined and the business case has to be established. The business case includes success criteria, risk assessment, and estimate of the resources needed, and a phase plan showing dates of major milestones. Inception is the smallest phase in the project, and ideally it should be quite short. Example outcomes of the inception phase are a general vision document of the core project's requirements, main constraints, an initial use-case model (10% -20% complete), and a project plan, showing phases and iterations [20]. Proposed metrics to be taken consideration in this phase are introduced in Table I.

TABLE I. METRICS FOR THE INCEPTION PHASE

| Metric | Notation | Definition |
|---|---|---|
| Planned Schedule | $D_{PLANNED}$ | The planned Date of delivery (usually the completion of an iteration, a release or a phase) |
| Planned Personnel | # $FT_{PLANNED}$ | The planned number of Full Time persons in the project at any given time |
| Proposed Requirements | # Reqs | The number of proposed requirements. |

The metrics Planned Schedule and Planned Personnel are mostly needed for comparison with actual schedule and personnel, in order to identify lack of available resources as well as delays in schedule quickly. The amount of Proposed Requirements tells about the progress of the product definition.

## B. *Metrics for Elaboration Phase*

During the elaboration phase a majority of the system requirements is expected to capture. The purpose of the phase is to analyze the problem domain, establish a sound architectural foundation, develop the project plan, and eliminate the highest risk elements of the project. The final Elaboration phase deliverable is a plan (including cost and schedule estimates) for the construction phase. Example outcomes of the elaboration phase are a use-case model (at least 80% complete), a software architecture description, supplementary requirements capturing the non-functional requirements and any requirements that are not associated with a specific use case, a revised risk list and a revised business case, and a development plan for the overall project. Proposed metrics to be taken consideration in this phase are introduced in Table II.

TABLE II. METRICS FOR THE ELABORATION PHASE

| Metric | Notation | Definition |
|---|---|---|
| Schedule:<br>Planned<br>/Actual Schedule | $\mathbf{D}_{PLANNED}$<br>$\mathbf{D}_{ACTUAL}$ | The planned/actual Date of delivery (usually the completion of an iteration, a release or a phase) |
| Staff:<br>Planned<br>/Actual Personnel | $\mathbf{\#FT}_{PLANNED}$<br>$\mathbf{\#FT}_{ACTUAL}$ | The planned/actual number of Full Time persons in the project at any given time |
| Requirements<br>-Proposed<br>-Accepted<br>-Not implemented | $\mathbf{\#Reqs}_{PROP.}$<br>$\mathbf{\#Reqs}_{ACCEP.}$<br>$\mathbf{\#Reqs}_{NOT\_IMPL}$ | The number (#) of<br>- proposed requirements<br>- reqs accepted by customer<br>- not implemented reqs |
| Tests<br>-Planned | $\mathbf{\#Tests}_{PLANNED}$ | The number (#) of<br>- planned tests |
| Documents:<br>-Planned<br>-Proposed<br>-Accepted | $\mathbf{\#Docs}_{PLANNED}$<br>$\mathbf{\#Docs}_{PROPOSED}$<br>$\mathbf{\#Docs}_{ACCEPTED}$ | The number (#) of planned /proposed /accepted documents to be reviewed during the project. |

The metrics related to requirements, tests and documents indicate the technical progress of the project from different viewpoints. Staffing metric may explain deviations in the expected progress vs. the actual progress, both from technical as well as from schedule viewpoint. Note that those metrics that are more relevant to measure by iterations (e.g., effort and size) are introduced later (in Section E).

## C. *Metrics for Construction Phase*

Construction is the largest phase in the project. During the phase, all remaining components and application features are developed and integrated into the product, and all features are thoroughly tested. System features are implemented in a series of short, time boxed iterations. Each iteration results in an executable release of the software. Example outcomes of the phase consist of a software product integrated on the adequate platforms, user manuals, and a description of the current release. Proposed metrics to be taken consideration in this phase are introduced in Table III. Note that those metrics that are continuously measured are introduced later (in Section E).

TABLE III. METRICS FOR THE CONSTRUCTION PHASE

| Metric | Notation | Definition |
|---|---|---|
| Planned<br>/Actual Schedule<br>Planned<br>/Actual Personnel | $\mathbf{D}_{PLANNED}$<br>$\mathbf{D}_{ACTUAL}$<br>$\mathbf{\#FT}_{PLANNED}$<br>$\mathbf{\#FT}_{ACTUAL}$ | Defined in the elaboration phase. |
| Requirements:<br>-Proposed<br>-Accepted<br>-Not implemented<br>-Started<br>-Completed | $\mathbf{\#Reqs}_{PROP.}$<br>$\mathbf{\#Reqs}_{ACCEP.}$<br>$\mathbf{\#Reqs}_{NOT\_IMPL}$<br>$\mathbf{\#Reqs}_{STARTED}$<br>$\mathbf{\#Reqs}_{COMPLETED}$ | The number (#) of<br>- proposed requirements<br>- reqs accepted by customer<br>- not implemented reqs<br>- reqs started to implement<br>- reqs completed |
| Change Requests:<br>-New CR<br>-Accepted<br>-Implemented | $\mathbf{\#CRs}_{NEW}$<br>$\mathbf{\#CRs}_{ACCEPTED}$<br>$\mathbf{\#CRs}_{IMPL.}$ | The number (#) of<br>- identified new CR or enhancement<br>- CRs accepted for implementation<br>- CRs implemented |
| Tests:<br>-Planned<br>-Passed<br>-Failed<br>-Not tested | $\mathbf{\#Tests}_{PLANNED.}$<br>$\mathbf{\#Tests}_{PASSED}$<br>$\mathbf{\#Tests}_{FAILED}$<br>$\mathbf{\#Tests}_{NOT\ TESTED}$ | The number (#) of<br>- planned tests<br>- passed tests<br>- failed tests<br>- not started to test |
| Defects<br>-by Priority: e.g., Showstopper, Medium, Low | $\mathbf{\#Dfs}_{\mathbf{PRIORITY}}$ | The number (#) of<br>- defects by Priority during the time period |
| Documents:<br>-Planned<br>-Proposed<br>-Accepted | $\mathbf{\#Docs}_{PLANNED}$<br>$\mathbf{\#Docs}_{PROPOSED}$<br>$\mathbf{\#Docs}_{ACCEPTED}$ | Defined in the elaboration phase. |

The metrics related to requirements, tests and documents indicate the technical progress of the project from different viewpoints. Metrics related to changes indicate both on the stability of the project technical content, and can explain schedule delays, and unexpected technical progress. Defect metrics tell both of the progress of testing, as well as maturity of the product.

## D. *Metrics for Transition Phase*

The final project phase of the RUP approach is transition. The purpose of the phase is to transfer a software product to a user community. Feedback received from initial release(s) may result in further refinements to be incorporated over the course of several transition phase iterations. The phase also includes system conversions, installation, technical support, user training and maintenance. From measurements viewpoint the metrics identified in the phases relating to schedule, effort, tests, defects, change requests and costs are still relevant in the transition phase. In addition, customer satisfaction is generally gathered in the transition phase.

## E. *Metrics for Iterations*

Each iteration results in an increment, which is a release of the system that contains added or improved functionality compared with the previous release. Each release is accompanied by supporting artifacts: release description, user's documentation, plans, etc. Although most iterations will include work in most of the process disciplines (e.g., requirements, design, implementation, testing) the relative effort and emphasis will change over the course of the project. Proposed metrics to be taken consideration in this phase are introduced in Table IV.

TABLE IV. METRICS FOR ITERATIONS

| Metric | Notation | Definition |
|---|---|---|
| Effort:<br>-Planned Effort<br>-Actual Effort | $\mathbf{E}_{PLANNED}$<br>$\mathbf{E}_{ACTUAL}$ | The planned/actual effort required of any given iteration of the project. |
| Size:<br>-Planned size<br>-Actual size | $\mathbf{SIZE}_{PLANNED}$<br>$\mathbf{SIZE}_{ACTUAL}$ | The planned /actual size of each iteration can be measured as SLOC (Source Lines of Code), Points or any other commonly accepted way. |
| Cost:<br>-Budgeted<br>-Expenditure | $\mathbf{COST}_{BUDGET}$<br>$\mathbf{COST}_{ACTUAL}$ | The budgeted cost /actual expenditure for any given iteration. |
| Velosity:<br>-planned /actual story points | $\mathbf{\#PTS}_{PLAN}$<br>$\mathbf{\#PTS}_{ACT}$ | How many story points are planned to be /actually implemented of any given iteration of the project. |
| Productivity: | $\frac{E_{ACTUAL}}{\# PTS_{ACTUAL}}$ | Use effort per acutally implemented story points for each sprint /iteration |

All of these metrics provide indication of the project progress and reasons for deviations should be analysed. These metrics should be analysed together with other metrics results (presented in Tables I-III) in order to gain comprehensive picture of the status.

## IV. EXPERIENCES AND DISCUSSION

The metrics presented in previous section were common for both of the companies. Although the metrics were chosen independently by both companies, the reasoning behind choosing these metrics was similar. An important reason was to come from a re-active into a pro-active mode, i.e., to introduce 'early warning' signals for the project and management. Specifically these metrics have been chosen as they indicate a well-rounded view of status in the various engineering disciplines and highlight potential issues in the project. This creates real possibilities to act proactively based on signals gathered from various engineering viewpoints. This is especially important in GSD, where information of project status is not readily available but needs special effort, distributed over sites and companies. Accordingly, the metrics set can be seen as a 'balanced score card', on which management can take the right measures, balancing insights from time, effort (e.g., staffing), cost, functionality (requirements) and quality (tests) perspective.

An important aspect was also that the metrics are easy of capture and that they can be captured from the used tools "for free", or can be quickly calculated at regular intervals. Costs and budgets are good examples of metrics that can be easily captured from the tools. This is also important from GSD viewpoint, as automated capturing reduces the chance of variations caused by differences in recording the metrics data in different sites. Neither of the companies use metrics based on lines-of-code as they did not find it to be a reliable indicator of progress, size or quality of design.

As can be seen, the metrics are quite similar as in single site development. However, the metrics may be analysed separately for each site, and comparisons between sites can thus be made in order to identify potential problems early. Also, while interpreting or making decisions based on the measurement results the distributed development implications need to be taken into account. Distributed development requires 'super-balancing': how to come to the right corrective action if for instance, in one side the % of not accepted requirements is high, and in the other side the # of passed tests is lagging behind. Distributed development may also affect the actual results of the measurements. For example, relating to subjective metrics, such as effort estimation, differences between backgrounds of the people (e.g., cultural or work experience) in different sites may affect the result.

The companies also use the measurement results to gain insight into why a measure varies between similar single site and multi-site projects in order to try to reduce potential variances. This also partially explains the use of the same metrics as single-site development. Furthermore, the challenges in communication and dynamics of distributed teams mean that working practices need to be addressed continuously. However, in addition to metrics results, paying close attention and acting on feedback from retrospectives is as important, if not more important than drawing strong conclusions from metrics alone.

Currently, both companies are in process of revamping their metric usage, but feel confident that these metrics are the right ones. Easy implementation and by that easy acceptance is the most crucial thing to get these metrics as established practice within the company.

Both companies are careful in introducing new metrics, as it's well known that too many metrics leads to overkill and rejection by the organization, and does not provide the right insights and indication for control measures. However, a potential measurement to be added to the set specifically from distributed development viewpoint, could be measurements related to time spent idling, i.e., waiting for something, and the time blocked because of the impediments elsewhere in the team as these affect productivity and highlight when a team is not performing. These additional metrics should be focused on measuring the project performance, especially task and team performance in GSD.

## V. CONCLUSION

The management of the increasingly common distributed product development project is proven to be more challenging and complicated than traditional one-site development. Metrics are seen as important activities for successful product development as they provide means to effectively monitor the project progress. However, defining useful, yet reasonable amount of metrics is challenging, and there is little guidance available for a company to define metrics for its distributed projects.

Globally distributed development generates new challenges and difficulties for the measurements. For example, the gathering of the measurements data can be problematic because of different development tools or their versions, work practices with related concepts can vary by project stakeholders or reliability of the gathered data can vary due to cultural differences, especially, in subjective evaluations. Furthermore, especially interpretation and decision-making based on the measurement results require that the distributed development implications are taken carefully into consideration.

This paper focused on describing a set of metrics that is successfully used in industrial practice in distributed product development. These metrics, are aimed especially to provide means to proactively react to potential issues in the project, and are meant to be used as a whole, not interpreted as single information of project status.

The metrics presented in the paper were common for both of the companies. Based on experiences, the reasoning for selecting these metrics was similar: they are easy to capture and can be quickly calculated and analysed at regular interval. Also, one of the most important reasons was that these metrics were aimed especially to provide means to proactively react to potential issues in the project. The balancing insights from time, effort, cost, functionality and quality was also seen as very important aspect.

## Acknowledgment

This paper was written within the PRISMA project that is an ITEA 2 project, number 07024 [21]. The authors would like to thank the support of ITEA [22] and Tekes (the Finnish Funding Agency for Technology and Innovation) [23].

## References

[1] J. Hyysalo, P. Parviainen, and M. Tihinen, "Collaborative embedded systems development: Survey of state of the practice," 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS 2006), IEEE, 2006, pp. 1-9.

[2] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," In Proceedings of Future of Software Engineering FOSE '07, IEEE Computer Society, 2007, pp. 188-198.

[3] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "Distance, dependencies, and delay in a global collaboration," In Proceedings of the ACM Conference on Computer Supported Cooperative Work, ACM, 2000, pp. 319-328.

[4] M. Jiménez, M. Piattini, and A. Vizcaíno, "Challenges and improvements in distributed software development: A systematic review," Advances in Software Engineering, 2009, pp. 14.

[5] S. Komi-Sirviö and M. Tihinen, "Lessons learned by participants of distributed software development," Knowledge and Process Management, vol. 12, (2), 2005, pp. 108-122.

[6] F. Q. B. da Silva, C. Costa, A. C. C. França, and R. Prikladinicki, "Challenges and solutions in distributed software development project management: A systematic literature review," In Proceedings of International Conference on Global Software Engineering (ICGSE2010), IEEE, 2010, pp. 87-96.

[7] S. Komi-Sirviö and M. Tihinen, "Great challenges and opportunities of distributed software development - an industrial survey," 15th International Conference on Software Engineering and Knowledge Engineering (SEKE2003), San Francisco, USA, 2003, pp. 489-496.

[8] V. R. Basili, "Software modeling and measurement: The Goal/Question/Metric paradigm," 1992.

[9] N. E. Fenton and S. L. Pfleeger, Software Metrics: A Rigorous and Practical Approach. PWS Publishing Co. Boston, MA, USA, 1998.

[10] M. Umarji and F. Shull, "Measuring developers: Aligning perspectives and other best practices," IEEE Software, vol. 26, (6), 2009, pp. 92-94.

[11] P. Kruchten, The Rational Unified Process: An Introduction. Addison-Wesley Professional, 2004.

[12] C. E. L. Peixoto, J. L. N. Audy, and R. Prikladnicki, "Effort estimation in global software development projects: Preliminary results from a survey," In Proceedings of International Conference on Global Software Engineering, IEEE Computer Society, 2010, pp. 123-127.

[13] K. Korhonen and O. Salo, "Exploring quality metrics to support defect management process in a multi-site organization - A case study," In Proceedings of 19th International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2008, pp. 213-218.

[14] D. B. Simmons and N. K. Ma, "Software engineering expert system for global development," In Proceedings of 18th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'06), IEEE, 2006, pp. 33-38.

[15] S. Misra, "A metric for global software development environment," In Proceedings of the Indian National Science Academy 2009, pp. 145-158.

[16] R. M. Lotlikar, R. Polavarapu, S. Sharma, and B. Srivastava, "Towards effective project management across multiple projects with distributed performing centers," In Proceedings of IEEE International Conference on Services Computing (CSC'08), IEEE, 2008, pp. 33-40.

[17] B. Sengupta, S. Chandra, and V. Sinha, "A research agenda for distributed software development," In Proceedings of the 28th International Conference on Software Engineering, ACM, 2006, pp. 731-740.

[18] K. H. Möller and D. J. Paulish, Software Metrics: A Practitioner's Guide to Improved Product Development. Institute of Electrical & Electronics Enginee, London, 1993.

[19] R. Van Solingen and E. Berghout, The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development. McGraw-Hill, 1999.

[20] P. Kruchten, "A rational development process," CrossTalk, vol. 9, (7), 1996, pp. 11-16.

[21] PRISMA, Productivity in Collaborative Systems Development, PRISMA project (2008-2011) homepage, URL: http://www.prisma-itea.org/ (Accessed 1.6.2011).

[22] ITEA 2, Information Technology for European Advancement, ITEA 2 homepage, URL: http://www.itea2.org/ (Accessed 1.6.2011).

[23] Tekes, the Finnish Funding Agency for Technology and Innovation, Tekes homepage. URL: http://www.tekes.fi/eng/ (Accessed 1.6.2011).