Release Readiness Indicator for Mature Agile and Lean Software Development Projects

Miroslaw Staron[1], Wilhelm Meding[2], and Klas Palm[2]

[1] Software Centre, Computer Science and Engineering
Chalmers / University of Gothenburg
SE-412 96 Gothenburg, Sweden
miroslaw.staron@ituniv.se
[2] Ericsson Metrics Team, Ericsson Product Development
Ericsson AB, Sweden
{wilhelm.meding,klas.palm}@ericsson.com

Abstract. Large companies like Ericsson increasingly often adopt the principles of Agile and Lean software development and develop large software products in iterative manner – in order to quickly respond to customer needs. In this paper we present the main indicator which is sufficient for a mature software development organization in order to predict the time in weeks to release the product. In our research project we collaborated closely with a large Agile+Lean software development project at Ericsson in Sweden. This large and mature software development project and organization has found this main indicator – *release readiness* – to be so important that it was used as a key performance indicator and is used in controlling the development of the product and improving organizational performance. The indicator was developed and validated in an action research project at one of the units of Ericsson AB in Sweden in one of its largest projects.

1 Introduction

Continuous delivery of customer value is crucial for software development companies operating in the market-driven context. This way of operating in the global market today is supported by the Agile and Lean ways-of-working with short feedback loops, empowered teams and customer involvement [1, 2]. Such concepts as customer value, customer pull and continuous quality assessment from Lean have taken the development organization to a mature stage with teams that are aware of the goals of the organization as well as the technical details of the products. The Agile software development methods are used for both small projects and for large development projects with over 100 designers [3-6] (and this trend increases). The methods are used also in the context of product maintenance, thus making the work of maintenance personnel more efficient [7].

As Agile methods are increasingly often used to address the challenges of continuous deployment, customer responsiveness and empowering teams for development of large software products, new challenges occur which relate to the

C. Wohlin (Ed.): XP 2012, LNBIP 111, pp. 93–107, 2012.

parallel development of features of large software products. Multiple teams which develop code for distinct features on a single component might not be fully aware about the changes/development of the code which is done by other parallel teams working on the same component [8]. Since the teams are often independent (empowered, i.e. self-directed, self-selected, and self-managed), there is a need for automated support in maintaining the quality of the product [4, 9].

Due to the dynamics of multiple teams, parallel features and self-management/organization, the main area where the support is needed is *release readiness* – the area is important for project managers, product owners and for the release responsible that need predictions on when the product is going to be ready to be deployed to customer environments – also known as "definition of done". The release readiness is a concrete realization of the definition of done in the context of mature Agile and Lean organization.

However, metrics in this area cannot be done manually as the manual work significantly increases the costs of the metric and decreases its quality over time. Automation is the key aspect in the monitoring of these two areas as "measurement" is not an activity that contributed directly to the product development, but is very important for it. The efficiency of automated measurement outweighs costs of using the measures and makes the measures transparent for the whole development organization – from designers to unit managers.

Based on the above we addressed the following research question in this paper:

Which are the main indicators in the area of release readiness?

The research question was addressed through an action research project conducted collaboratively between University of Gothenburg and Ericsson AB in Sweden. The action research project took place in a product development organization which develops a mature core network telecommunication product handling mobile data traffic. The results of the project showed one single indicator which was introduced and validated in a large software development project at that unit and is under implementation at another large software development project.

The remaining of the paper is structured as follows: section 2 presents the most relevant related work. Section 3 describes the organizational context of our work and outlines the design and operation of our action research study. Section 4 presents the indicator and section 5 presents the results from the evaluation of the indicator in the organization. Finally section 6 contains the conclusions from our work.

2 Related Work

The work presented in this paper is the continuation of the work on monitoring bottlenecks in large software development organizations which was conducted in the same unit of Ericsson [4]. In the previous work we developed and introduced a method based on automated indicators for monitoring the capacity and bottlenecks in the work flow in the large software development project – i.e. the process/project view of software development in that organization. In our current work we addressed

the product view by developing and introducing the quality readiness indicator – i.e. indicator showing when the product is ready to be released/deployed to the customers.

Measuring business value is one of the main measures which should be used by Agile teams and companies [10]. The awareness of how the team contributes to the value is an important driver for the success of Agile projects. What the authors of the cited article postulate is similar to what we intend to achieve – provide key information without introducing manual work overhead. The focus of the cited article is on the customer value, whereas the focus of this article is on quality risk monitoring and predicting delivery time – both articles complement each other.

Another important measure which is claimed to stimulate agility in software development teams is the RTF (Running Tested Features) measure, popular in XP [11]. The metric combines three important concepts – the feature (i.e. a piece of code useful for the end-user, not a small increment that is not visible to the end user), execution (i.e. adding the value to the product through shipping the features to the customer), and the testing process (i.e. the quality of the feature – not only should it be execute, but also be of sufficient quality). This measure stimulates smart continuous deployment strategies and is intended to capture similar aspects as our release readiness indicator although in smaller projects.

A set of other metrics useful in the context of continuous deployment can be found in the work of Fritz [12] in the context of market driven software development organization. The metrics presented by Fritz measure such aspects as continuous integration pace or the pace of delivery of features to the customers. These metrics complement the two indicators presented in this paper with a different perspective important for product management.

The delivery strategy which is an extension of the concept of continuous deployment has been found as one of the three key aspects important for Agile software development organizations in a survey of 109 companies by Chow and Cao [13]. The indicator presented in this paper is a means of supporting organizations in their transition towards achieving efficient delivery processes which are in line with the delivery strategy prioritized by practitioners in this survey.

3 Organizational Context

The indicator presented in this paper was based on a study of one unit of Ericsson with approximately 700 persons. The unit was located at multiple sites with over 30 development teams working in a software development project of ca. 150 persons. The unit was mature in both the software development paradigm (*streamline development*) and the measurement processes (adoption of ISO/IEC 15939, [14]). This ISO standard is one of the key aspects of our work since it defines one of the main concepts – *indicator* – used in the organization. The other key concepts are *information need* and *stakeholder*. The notion of indicators has shown itself to be one of the key aspects of the successful adoption of such "metrics" as quality readiness [14].

3.1 ISO/IEC 15939

The current measurement processes in the organization are based on ISO/IEC 15939:2007 standard, which is a normative specification for the processes used to define, collect, and analyze quantitative data in software projects or organizations. The central role in the standard is played by the information product which is a set of one or more indicators with their associated interpretations that address the information need [15]. The information need is an insight necessary for a stakeholder to manage objectives, goals, risks, and problems observed in the measured objects [15]. These measured objects can be entities like projects, organizations, software products, etc. characterized by a set of attributes. We use the following definitions from ISO/IEC 15939:2007 [16]:

- Base measure – measure defined in terms of an attribute and the method for quantifying it. This definition is based on the definition of base quantity from.
- Derived measure – measure that is defined as a function of two or more values of base measures. This definition is based on the definition of derived quantity from.
- Indicator – measure that provides an estimate or evaluation of specified attributes derived from a model with respect to defined information needs.
- Decision criteria – thresholds, targets, or patterns used to determine the need for action or further investigation, or to describe the level of confidence in a given result.
- Information product – one or more indicators and their associated interpretations that address an information need.
- Measurement method – logical sequence or operations, described generically, used in quantifying an attribute with respect to a specified scale.
- Measurement function – algorithm or calculation performed to combine two or more base measures.
- Attribute – property or characteristics of an entity that can be distinguished quantitatively or qualitatively by human or automated means.
- Entity – object that is to be characterized by measuring its attributes.
- Measurement process – process for establishing, planning, performing and evaluating measurement within an overall project, enterprise or organizational measurement structure.
- Measurement instrument – a procedure to assign a value to a base measure.

The view on measures presented in ISO/IEC 15939 is consistent with other engineering disciplines, the standard states that it is based on ISO/IEC 15288:2007 (Software and Systems engineering - Measurement Processes) [17], ISO/IEC 14598-1:1999 (Information technology - Software product evaluation) [18], ISO/IEC 9126-x [19], ISO/IEC 25000 series of standards, or International vocabulary of basic and general terms in metrology (VIM) [16]. Conceptually, the elements (different kinds of measures) which are used in the measurement process can be presented as in Figure 1.

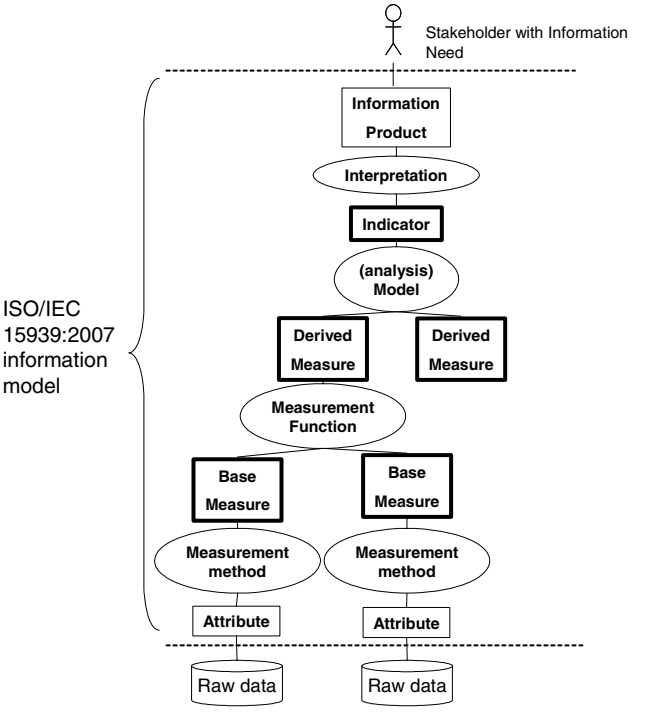

Fig. 1. Measurement system information model (from ISO/IEC 15939:2007)

One of the key factors for every measurement system is that it has to satisfy an information need of a stakeholder – i.e. there needs to be a person/organization who/which is dependent on the information that the measurement system provides. Typical stakeholders are project managers, organization managers, architects, product managers, customer representatives, and similar [20-23]. The indicator is intended to provide information along with interpretation, which implies the existence of an analysis model that eases the interpretation. The analysis model is a set of decision criteria used when assessing the value of an indicator – e.g. describing at which value of the indicator we e.g. set a red flag signaling problems in the measured object. The derived measures (based on the definition of the derived quantity) and base measures (based on the definition of the base quantity) are used to provide the information for calculating the value of the indicator.

3.2 Streamline Development, SD

The context of the case study was one of the software development organizations of Ericsson AB. The organization and the project within Ericsson, which we worked closely with, developed large products for the mobile telephony network. The size of the organization was several hundred engineers and the size of the projects can be up to 200 engineers. Projects were executed according to the principles of Agile software

development and Lean production system referred to as Streamline development (SD) within Ericsson [24]. In short, the principles of Streamline development postulated that software development was organized as a series of activities centered around the product and executed mainly in cross-functional teams responsible for the design, implementation and partially testing of their part of the product (e.g. a feature) [25]. This was found to be rather typical process design that addressed market pull in Lean development [26].

A noteworthy fact observed in our study was that in SD the releases of new software versions to customers were frequent and that there was always a release-ready version of the system: referred to as Latest System Version, LSV [25]. It is the defects existing (and known) in that version of the system that were of interest for the project and product management. Ideally the number of known defects in that version of the system should be 0, however, during the development (i.e. between releases) this number might vary as there might be defects which are discovered during the process of integrating new features (parts of code) into the latest system version. In practice there are almost always defects being fixed as integration and testing is continuous. However, at release times the main branch was frozen with the purpose of removing all defects – this removal was the responsibility of the release project, which was a project aimed at packaging the product and preparing it for the market availability.

An overview of the development process with the list of metrics used by the relevant stakeholders is presented in Figure 2.

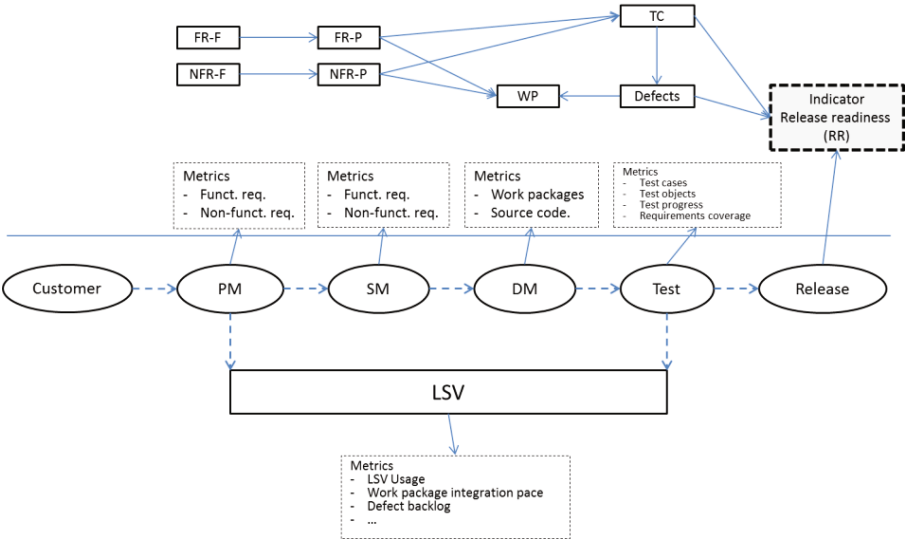

Fig. 2. Overview of the development process and measures collected

In Figure 2 the process started with the customer requirements being captured by Product Management (PM) who divided the requirements into functional and non-functional requirements for customer features (FR-F and NFR-F). These requirements

were broken down into functional and non-functional requirements for product features (FR-P and NFR-P). These requirements were then packaged into work packages (WP) for design teams (DM). The requirements were also traced to test cases (TC) which verified that the requirements were implemented and that they were implemented with the appropriate quality (e.g. performance, security). All requirements were traced to test cases monitored by the quality manager at the organization (through automated measurement systems like [4, 27]). Defects could appear when the test cases were executed and these defects (Defect) were fed back to the design teams which had to remove them from the code before the product was released.

The main criterion for releasing the product was that all functionality was in place. This meant that all requirements were linked to test cases, all test cases were executed and all test cases were passed. This also meant that all defects were removed since discovering of a defect equals to failure of a test case which entails the need to re-execute the test case.

Because the above mechanisms were in place, including automated measurement systems for monitoring that all requirements were traced to test cases and that all defects are reported, only one indicator was needed and sufficient to check whether the product is ready to be released – RR (Release Readiness).

3.3 Research Method and Study Design

In this research we chose *action research* as the most appropriate approach [28]. We worked in close collaboration with product development projects at one of the units of Ericsson and provided measures (indicators) based on the requests from the organization. In this section we present in detail the design of cycles which focused on the development of the RR indicator and its evaluation. The summary of all action research cycles is as follows:

1. Initial cycle: elicitation of "theory" behind the measures. In this cycle we planned how understand/verify which factors are important for the measures – e.g. filtering criteria for test cases or types of changes. The whole research team was involved (the research team is described in the next paragraphs).
2. Initial development cycle: development of the measures and application of them on historical data. In this cycle we developed and validated the measures on the historical data available at the company. The university researcher was involved in development and the research team was involved in the feedback loop.
3. Deployment cycle: the developed indicators were deployed in the project with ca. 100 developers working on a large telecom network product. The university researcher was involved in data collection and monitoring of the quality of the data. The rest of the research team was involved in the monitoring of the situation in the project and checking whether the indicators reflect the current situation in the project.
4. Evaluation and Improvement cycle: the final cycle of the action research evaluated the empirical validity of the indicators. The evaluation was done by measuring the

correctness of the prediction. The whole research team was involved in the evaluation cycle whereas the measurement team leader was involved in the maintenance of the developed measurement system.

The sampling of the project was convenience sampling [29] – the research team was part of this project and the development of the indicator was needed for that particular development project.

Each cycle contained relevant roles, but the core research team consisted of:

- University researcher with background in software engineering and focus on software metrics
- Metric team leader with the responsibility to develop, deploy and maintain measurement systems in the company
- Quality manager with the responsibility for the quality in the project/product
- Test leader with the responsibility for test analysis and execution for the project/product where the measures were deployed.
- Deputy project manager with the responsibility to monitor delivery of features in the project/product where the measures were deployed.

The additional involved roles (in particular cycles) were team managers, line/section managers, and technical responsible for software components in the system.

Development of Indicators (cycle 2). For the development of the quality readiness indicator (QR) the research team started with an initial set of predictor variables per week, including:

- #[1] executed test cases
- # planned test cases (per priority and type)
- # failed test cases
- # passed test cases
- # defects discovered per test case
- # defects discovered
- # defects to be removed (defect backlog in Figure 2)
- # defects removed
- # defects verified
- # features integrated into the main code branch (work package integration pace in Figure 2)
- # empty integration spots (LSV usage Figure 2)

According to the ISO 15939 Measurement Information Model we characterized these predictor variables as base measures with precisely defined measurement methods for data collection. Statistical correlation analyses were performed and they were complemented with expert opinion on whether there exist causal relationships in the data (a method which was previously shown to be successful in that organization [5]). In particular the research team was interested in such aspects as for example *if more*

[1] Reads: “number of“.

features are integrated, will that cause higher defect reporting rate after a number of weeks in a project? These aspects could be captured by statistical analyses of correlations or time series, but it was important for the team to capture the empirical properties of the development process together with the statistical dependencies in the data.

The goal of the research team was to find the minimal set of predictor variables that would not be correlated to each other and that would reflect the empirical properties of agile software development at Ericsson. This resulted in the set of 4 predictor variables (discussed in section 4).

Evaluation (Cycle 4). The evaluation cycle was planned to take place in a longer period of time after the measures were deployed into the organization. The evaluation was planned to be a number of meetings during a period of 6 months with the key stakeholders for the indicators – each meeting was a short interview. Two main questions were posed in the interviews:

- The indicator shows that the product will be ready to release in week <X>. Does it seem "right" according to your professional opinion and to your judgment of the situation in the project?
- In case you see any discrepancies, could you briefly describe why (according to you) this is the case?

The goal of these questions was to focus the stakeholder to reason about the empirical assessment of the same situation which was calculated by the indicators.

4 Release Readiness (RR) Indicator

In this section we present the release readiness indicator and its visualization at the company.

4.1 Time to Release

The product release-quality readiness indicator predicted when the product under development would have the sufficient quality for being released. The quality was measured in a number of open defect reports for the product – meaning that the right quality for releasing of the product was 0 defects. The 0-defect criterion was sufficient only when another criterion was fulfilled – all functionality was tested and all test cases were passed. This means that in practice the indicator of quality-release readiness had to take into account both.

In the case of the studied organization at Ericsson we introduced the following indicator (RR):

$$RR = \left(\frac{\#\,defects}{defect\_removal\_rate - (test\_execution\_rate - test\_pass\_rate)}\right)$$

Where *#defects* was the number of open defects for the product[2], *defect_removal_rate* was the average number of removed defects during the last 4 weeks, *test_execution_rate* was the average number of test cases executed during the last 4 weaks and *test_pass_rate* was the average number of test cases passed during the last 4 weaks. The 4 weeks period was chosed based on statistical and empirical analyses. These analyses showed that based on the length of the test cycles and the defect removal activities the 4 week period was the most logical length for this prediction and provided the most accurate results.

The indicator predicted in which week the release would be possible given the number of known defects now, how many defects were removed on average in the last few (4) weeks, and how many defects were expected to be discovered given the number of test cases being executed[3].

The fact that the prediction formula contains only the test execution and pass rate instead of test planning was dictated by the fact that the organization was mature and its processes used continuous testing (i.e. running tests continuously and observing the number of passed test cases) rather than pre-planned testing sequences executed once in the project. The nature of the stakeholder for this indicator (project manager) and the auxiliary metrics of traceability of requirements to test cases made it possible to use this prediction formula with confidence that no vital information was omitted.

Figure 3 presents how the indicator was spread in the organization on a daily basis – in a form of MS Vista Sidebar gadget.

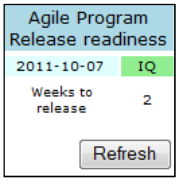

Fig. 3. MS Vista Gadget with predicted week of release

The content of the gadget shows 2 weeks for the project to obtain the release-quality (*weeks to release)*. The presentation is simple and succinct giving the stakeholder (the manager of the studied product development project) the necessary information. The gadget also contains information about the validity of the measure – Information Quality which abbreviates to *IQ*, [30].

[2] This measurement included all defects that need to be removed from the product before the release.

[3] One could notice that there is no measure like ”#of test cases planned” in the formula. This is intentional as the rationale behind this indicator is that the test cases are to be executed until all defects are fixed – that in particular means that “failing” test cases are executed until they are passed.

5 Results from Evaluation

The results from the evaluation of our indicator were collected from the mentioned product development project at Ericsson. The introduction of the indicator showed that the organization effectively adopted it as a KPI – Key Performance Indicator and that the quality manager for the project used it to replace other activities in his work.

5.1 Evaluation Results

Figure 4 presents an excerpt from a chart where the release week is shown. Due to the confidentiality of the data we cannot provide accuracy metric like MRE (Mean Relative Error) as it would reveal the length of the development cycle for the product. However, we can show differences in the prediction in the absolute terms – shown in the diagram in Figure 4, alike the evaluation of other predictions in the same organization [31]. We followed up the indicator in a discussion with the stakeholder on a bi-weekly basis. The chart shows that the indicator predicted the release to be between week number 8 in 2011 and week number 14 in 2011. During the discussion with the stakeholder, the stakeholder was able to relate to the differences and pointed to particular events in the project which slowed the testing process, increased the pace of defect removal activities or reallocated resources. Each of the events caused the change of the indicator in the correct direction, for example: decreasing the pace of defect removal has increased the predicted release week when making the prediction in week 4, 5, and 6; increasing the test execution rate has decreased the time to release in week 9. The stakeholder perceived it to be normal that the indicator changes the prediction since the situation in the project changed during the evaluation period – in fact the stakeholder expected the indicator to show this trend.

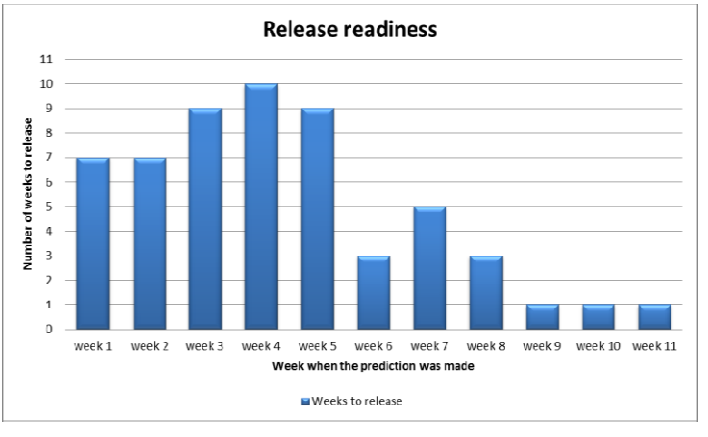

Fig. 4. Predicted release week

During the evaluation of the indicator with the quality manager, we obtained the following quote which summarizes his perception of the indicator:

"Thanks to this indicator I do not have to 'walk the floor' anymore and ask about the status. I have everything I need in one number and I can monitor that all necessary assumptions hold"

By saying "*monitor that all necessary assumptions hold"* the quality manager expressed his confidence in the indicator which was presented to him together with the associated statistics about such assumptions as:

- Traceability between test cases to requirements
- Traceability between test cases to work packages
- Traceability between work packages and requirements
- Code coverage by test cases

These assumptions reflect the ways of working in the project as presented earlier in Figure 2 in section 3.2. The measures which measured to what degree the assumptions hold for the indicator provided the complete picture for the quality manager and allowed him to assure the quality of information provided to the stakeholder – the project manager.

The release-readiness indicator and the metrics controlling the assumptions were fully automated – i.e. the process of collecting, analyzing and presenting information was executed entirely without the manual work. The data was collected from the defect database and the database containing log of executed, planned, passed and failed test cases. The full automation was appreciated and contributed to maintaining the high quality of the presented indicator [30] – and in this way making it trustworthy in the organization to such extent as being adopted as Key Performance Indicator (KPI). The stakeholder's evaluation of this indicator can be summarized in the following quote: "*This measurement combines all the information I want to know in order to understand the quality of the product as well as the status and progress of the development program, into one understandable and very clear KPI"*. The adoption of this indicator as one of the few KPI in the organization showed that the organization (with up to 200 engineers) perceived this indicator to be trustworthy enough to be used in observing organization's capability to continuously deliver the customer value.

5.2 Validity Evaluation

As every empirical study, our study has certain threats to validity, which are considered using the categories presented by Wohlin [32].

The main *external validity* threat was the fact that we evaluated the indicator at a single company. We minimized this threat by using the same indicator in another development project at Ericsson with similar results (although the evaluation was not as strict as the one presented in this paper). Based on the literature review and the presence of such metrics as feature backlog or RTF (Running Tested Features) in smaller Agile development projects, we believe that our results are applicable to other large software development projects where the assumptions described in section 3 are met. The main *conclusion validity* threat was the lack of statistical power in the evaluation of our indicator – we evaluated it in a single project and due to the

confidentiality of the data we cannot reveal the accuracy of the indicator (as this would reveal the length of the development cycle in the organization). In order to minimize this threat we performed the empirical validation of the indicator in a series of meetings with the stakeholder. The main *construct validity* of our study is the fact that the organization's high maturity meant that the dependencies between metrics (such as traceability of requirements to test cases) were implicitly embedded into the formula. In order to minimize the threat we monitored these dependencies in supporting measurement systems in the organization.

6 Conclusions

In this paper we addressed the problem of supporting mature Agile and Lean software development organizations in effective and efficient prediction when the software product is ready to be released. The problem is evident in larger software development organizations with multiple parallel empowered teams contributing to the development of a single product. Before this indicator was introduced, the organization had worked in a release-planning matter with standard release planning. After the indicator was introduced the organization could stimulate its change towards a direction of being able to continuously release the current version of its large software product without the risk of jeopardizing the operations of the customer – i.e. at the high quality level.

The indicator presented in this paper was supported with a number of metrics that controlled the assumptions of the indicator. All measurement systems collecting the information were automated and therefore the metrics and the indicators were very much appreciated and spread throughout the company. Based on our experiences from the introduction of the presented indicator we could recommend companies interested in introducing this indicator to put special attention to deliver to the stakeholder the metrics for controlling the assumptions as part of the information quality evaluation.

Acknowledgements. This research has been carried out in the Software Centre, University of Gothenburg, and Ericsson AB.

References

[1] Poppendieck, M., Poppendieck, T.: Implementing Lean Software Development: From Concept to Cash. Addison-Wesley, Boston (2007)

[2] Salo, O., Abrahamsson, P.: Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum. IET Software 2, 58–64 (2008)

[3] Korhonen, K.: Exploring Defect Data, Quality and Engagement during Agile Transformation at a Large Multisite Organization. In: Agile Processes in Software Engineering and Extreme Programming, pp. 88–102 (2010)

[4] Staron, M., Meding, W.: Monitoring Bottlenecks in Agile and Lean Software Development Projects – A Method and Its Industrial Use. In: Product-Focused Software Process Improvement, Tore Cane, Italy, pp. 3–16 (2011)

[5] Staron, M., Meding, W., Söderqvist, B.: A method for forecasting defect backlog in large streamline software development projects and its industrial evaluation. Information and Software Technology 52, 1069–1079 (2010)
[6] Gabrielle, B.: Rolling Out Agile in a Large Enterprise. In: Hawaii International Conference on System Sciences, pp. 462–462 (2008)
[7] Korhonen, K.: Adopting Agile Practices in Teams with No Direct Programming Responsibility – A Case Study. In: Product-Focused Software Process Improvement, pp. 30–43 (2011)
[8] Ball, T., Nagappan, N.: Use of relative code churn measures to predict system defect density. In: 27th International Conference on Software Engineering, St. Louis, MO, USA, pp. 284–292 (2005)
[9] Staron, M., Meding, W.: Defect Inflow Prediction in Large Software Projects. e-Informatica Software Engineering Journal 4, 1–23 (2010)
[10] Hartmann, D., Dymond, R.: Appropriate agile measurement: using metrics and diagnostics to deliver business value. In: Agile Conference, pp. 126–134 (2006)
[11] Jeffries, R.: A Metric Leading to Agility (2004), http://xprogramming.com/xpmag/jatRtsMetric
[12] Fitz, T.: Continuous Deployment at IMVU: Doing the impossible fifty times a day (2009), http://timothyfitz.wordpress.com/2009/02/10/continuous-deployment-at-imvu-doing-the-impossible-fifty-times-a-day/
[13] Chow, T., Cao, D.-B.: A survey study of critical success factors in agile software projects. Journal of Systems and Software 81, 961–971 (2008)
[14] Staron, M., Meding, W., Karlsson, G., Nilsson, C.: Developing measurement systems: an industrial case study. Journal of Software Maintenance and Evolution: Research and Practice, n/a–n/a (2010)
[15] International Standard Organization and International Electrotechnical Commission. Software engineering – Software measurement process. ISO/IEC, Geneva (2002)
[16] International Bureau of Weights and Measures. In: International vocabulary of basic and general terms in metrology = Vocabulaire international des termes fondamentaux et généraux de métrologie, 2nd edn., International Organization for Standardization, Genève (1993)
[17] International Standard Organization. Systems engineering – System life cycle processes 15288:2002 (2002)
[18] International Standard Organization. Information technology – Software product evaluation 14598-1:1999 (1999)
[19] International Standard Organization and International Electrotechnical Commission. ISO/IEC 9126 - Software engineering – Product quality Part: 1 Quality model. International Standard Organization / International Electrotechnical Commission, Geneva (2001)
[20] Umarji, M., Emurian, H.: Acceptance Issues in Metrics Program Implementation, pp. 20–20 (2005)
[21] Gopal, A., Mukhopadhyay, T., Krishnan, M.S.: The impact of institutional forces on software metrics programs. IEEE Transactions on Software Engineering 31, 679–694 (2005)
[22] Umarji, M., Emurian, H.: Acceptance issues in metrics program implementation, p. 10 (2005)
[23] Kilpi, T.: Implementing a Software Metrics Program at Nokia. IEEE Software 18, 72–77 (2001)
[24] Tomaszewski, P., Berander, P., Damm, L.-O.: From Traditional to Streamline Development - Opportunities and Challenges. Software Process Improvement and Practice, 1–20 (2007)

[25] Akg, A.E., Keskin, H., Byrne, J., Imamoglu, S.Z.: Antecedents and consequences of team potency in software development projects. Inf. Manage. 44, 646–656 (2007)
[26] Liker, J.K.: The Toyota way: 14 management principles from the world's greatest manufacturer. McGraw-Hill, New York (2004)
[27] Staron, M., Meding, W., Nilsson, C.: A Framework for Developing Measurement Systems and Its Industrial Evaluation. Information and Software Technology 51, 721–737 (2008)
[28] Susman, G.I., Evered, R.D.: An Assessment of the Scientific Merits of Action Research. Administrative Science Quarterly 23, 582–603 (1978)
[29] Yin, R.K.: Case Study Research: Design and Methods. SAGE Publications Inc. (2008)
[30] Staron, M., Meding, W.: Ensuring Reliability of Information Provided by Measurement Systems. In: Software Process and Product Measurement, pp. 1–16 (2009)
[31] Staron, M., Meding, W.: Short-term Defect Inflow Prediction in Large Software Project - An Initial Evaluation. In: International Conference on Empirical Assessment in Software Engineering (EASE), Keele, UK (2007)
[32] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B., Wesslèn, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publisher, Boston MA (2000)