Electronic Research Archive of Blekinge Institute of Technology
<http://www.bth.se/fou/>

This is an author produced version of a journal paper. The paper has been peer-reviewed but may not include the final publisher proof-corrections or journal pagination.

Citation for the published Journal paper:

Title:

Author:

Journal:

Year:

Vol.

Issue:

Pagination:

URL/DOI to the paper:

Access to the published version may require subscription.

Published with permission from:

Early Identification of Bottlenecks in Very Large Scale System of Systems Software Development

Kai Petersen*, Peter Roos⁺, Staffan Nyström⁺, Per Runeson**

*Blekinge Institute of Technology**, *FindOut⁺*, *Lund University***, Sweden

SUMMARY

System of systems are of high complexity and for each system many different requirements are implemented in parallel. Systems are developed with some degree of managerial independence, but later on have to work together. In this situation many requirements are written, implemented, and tested in parallel for different systems that are to be integrated. This makes identifying bottlenecks challenging and visualizations often used on project level (such as Kanban Boards or Burndown Charts) have to be extended/complemented to cope with the increased complexity. In response to these challenges the contributions of this study are to propose (1) a visualization for early identification and proactive removal of bottlenecks; (2) a visualization to check on the success of bottleneck resolution; and (3) to provide an industry evaluation of the visualizations in a case study of a system of systems developed at Ericsson AB in Sweden. The feedback by the practitioners showed that the visualizations were perceived as useful in improving throughput and lead-time. The quantitative analysis showed that the visualizations were able in identifying bottlenecks and showing improvements or the lack thereof. Based on the qualitative and quantitative data collected we conclude that the visualizations are useful in bottleneck identification and resolution. Copyright © 2012 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: System of Systems; Bottleneck Identification; Software Process.

1. INTRODUCTION

System of systems are very complex as each system is large in size, and at the point of integration the different systems have to be able to interact [13]. Furthermore, many different parties and stakeholders are involved in the development of the systems that are often managerially independent [4]. If these systems are developed for a large market with a continuous inflow of new requirements, this adds further to the complexity. This overall leads to a very high number of requirements being in different stages in the development lifecycle (being written, being implemented, being tested, being ready for release, and so forth).

Given the large amount of work done in parallel for the different systems identifying bottlenecks in the development process is a non-trivial task. We added the following definition to the paper: According to Anderson [1] a bottleneck “*is the process step with the lowest production rate in the system [note from the authors: system refers to the overall end to end process]*”. The production rate (also referred to as velocity) is the rate of completion (cumulative) within a specified period of time (e.g. an iteration or sprint). According to Anderson [1], the measure for production rate is “*how many units of inventory (story points) were delivered in each iteration*”. Other inventory

*Correspondence to: Kai Petersen, Blekinge Institute of Technology, Karlskrona, Sweden (Contact him at: kai.petersen@bth.se).

units could be requirements delivered, or requirements tested, etc. A bottleneck becomes visible when inventory levels in an activity are growing, given that requirements stay in an activity for a long time (cf [25]). Furthermore, a bottleneck is a symptom. The reasons for a bottleneck can be, e.g., non-availability of resources, lack of backward compatibility, or major events such as a system failure which requires people to attend to maintenance instead of new product development. It can also relate to inefficient work practices as defined and used in the process.

It is important to identify potential bottlenecks early and take corrective actions. Work on existing visualizations that have been successful on project level have to be extended in order to identify bottlenecks early in the system of systems context. Cumulative flow diagrams [1, 25], value stream analysis [16, 18], burn-down charts [27, 5], and line of balance [17] are backward looking and hence provide useful input for postmortems and retrospective process analysis, but have limited ability to show whether significant bottleneck will occur in the near future. They also do not integrate viewpoints (e.g. showing the situation for a system or set of systems, release or set of releases), which are important when dealing with multiple systems, but not when looking at a single project. Furthermore, an aggregated view (e.g. adding up the number of items in a specific phase in cumulative flow diagrams, burn-down charts, and line of balance) is shown, but one is not able to see individual requirements, and which requirements are stuck in the process.

In order to address the above mentioned challenges we propose visualizations that are designed with early bottleneck identification in mind, and that are able to assess whether bottlenecks were removed successfully. Early bottleneck identification here means that a potential bottleneck becomes visible before it actually becomes a bottleneck (e.g. a requirement is stuck in the design phase for a while, and in short time will be considered a bottleneck if it is not finalized for testing now). Another design goal was to have a simple and easy to interpret solution that can be realized with collecting few measurements. In particular, this study makes the following contributions:

- Propose a visualization that allows to identify bottlenecks early in complex system of systems development.
- Propose a visualization to check on the effect of bottleneck removal.
- Validate the use of both visualizations for a very large real-world system of systems developed at Ericsson AB.

The paper is structured as follows. Section 2 presents the related work focusing on visualizations for bottleneck identification. In Section 3 the research process for the study is presented, while Section 4 presents the proposed visualizations for identification and evaluation of bottlenecks. Section 5 presents the validation results, followed by a discussion in Section 6. Section 7 concludes the paper.

2. RELATED WORK

Five commonly known visualizations for bottleneck identification are of relevance, these are value stream maps [16, 18], burn-down charts [27, 5], cumulative flow diagrams [1, 25], line of balance [17], and Kanban board [10].

Burn-down Charts (BDC): Burn-down charts start with a backlog of items for a project and count down whenever a work-task is completed. Burn-down charts are used to visualize the progress of projects in agile methodologies, such as SCRUM [27]. An example of a burn-down chart is shown in Figure 1(a). The y-axis shows the number of work-items remaining to be completed after a specific time (x-axis). A general limitation of burn-down charts is that the work backlog is mostly considered fixed and not expandable [5]. Furthermore, burn-charts are backward looking and show an aggregated view (total number of work items), i.e. one cannot see whether a specific item is soon to become a bottleneck.

Cumulative Flow Diagrams (CFD): Cumulative flow diagrams are a way of visualizing the flow of software through the software development process. A fictional example of the construction of a cumulative flow diagram for requirements is shown in Figure 1(b). The x-axis shows the time-line and the y-axis shows the cumulative number of items. The top-line is the total number of

items in development. The line below the top-line represents the number of requirements that have been specified and are now in development. The vertical distance between lines represents work in progress at a specific point in time. For example, in week 6 there are around 20 requirements in the specification phase, 7 requirements in the design and implementation phase, 18 in the software testing phase, 5 requirements waiting for release, and 30 released requirements. That means, the lines represent the rate at which work is completed and handed over in between phases. Further details on evaluation and interpretation of cumulative flow diagrams are published by Anderson [1], Petersen [25], and Miranda and Bourque[17]. CFDs are also backward looking and show an aggregated view.

Line of Balance (LOB) Status Chart: The line of balance shows the number of items that should have passed a control point (plan) versus the number of items that have actually passed the control point at one specific point in time [30, 2]. A control point is, for example, completion of requirements specification, completion of design, etc. Figure 1(c) shows an example of line of balance. The figure visualizes the backlog still to be developed at a specific point in time. In addition, it shows that fewer requirements have been specified than planned, not all requirements that should have been implemented are finalized, design is in perfect balance, and a fewer requirements are already released than planned. LOB is backward looking and presents an aggregated view as well.

Value Stream Maps (VSM): A value stream map follows a specified item through the process in order to determine value added in each processing step. For example, an item is a specific requirement, and a processing step is the detailed specification of the requirement. First, a current state map is drawn and waiting times (WT) as well as processing times (PT) are added. Secondly, wastes are identified that, when being removed, lead to the highest improvement potential. Thirdly, process improvements are proposed and a new map of the process flow with estimations for waiting and processing times is drawn. An example of a value stream map is shown in Figure 1(d). Further explanations and an industry evaluation of value stream mapping are presented by Mujtaba et al. [18]. The limitation of value stream mapping is that it looks at bottlenecks retrospectively and only considers an individual work item in a single run through the process.

Kanban Board: The Kanban Board shows the work in progress for different activities. The work in progress is split into ongoing work and completed work for each activity. Completed work in the “done” category can be pulled by the following activity (e.g. requirements that are specified are ready to be pulled for design and implementation). An example of a Kanban board is illustrated in Figure 1(e). Kanban boards provide a snapshot, but are mostly project focus and managing them for multiple systems and with a very high amount of work items on a whiteboard leads to easily losing the overview.

None of the approaches provides a systematic way of integrating different viewpoints. As most of the approaches are not independent of individual projects/releases certain viewpoints are not well catered for, such as the product manager who requires information across systems and releases.

3. RESEARCH PROCESS

The research was conducted as a technology transfer project (cf. [7]). An overview of the research process for transferring solutions to industry is shown in Figure 2.

In the first step problems and issues were identified that should be addressed and were considered relevant by the practitioners, in this case the identification of bottlenecks. The related works on bottleneck identification, as presented in Section 2, showed that there was a gap between features needed and features provided by existing visualizations for bottleneck identification.

As a consequence, two visualizations were introduced to support the discovery of bottlenecks and to assess the effect of improvements, which are presented in Section 4. As can be seen in Figure 2, two validation types are distinguished, namely static validation and dynamic validation, which are reported in Section 5.

During the static validation practitioner feedback was collected based on presentations/explanations of the visualization for bottleneck identification. The static validation served two

Figure 1. Overview of Visualizations for Bottleneck Identification

Figure 2. Use of Technology Transfer Model in this Study

purposes. First, feedback was collected in order to improve the solution before going “live”. Second, the acceptance of the solution could be tested.

The dynamic validation was the application of the solution to 15 months of industrial data (five quarters) and thereby observations were made with regard to its usefulness, in this case the usefulness with respect to identifying bottlenecks and achieving improvements according to the visualization.

4. VISUALIZATIONS FOR BOTTLENECK IDENTIFICATION

In this paper, we present two visualizations used in industry to manage and resolve bottlenecks. The first visualization's goal is to show and locate bottlenecks to specific activities in the software development life-cycle in order to take corrective actions for improving the flow (Section 4.1). We refer to this as *in-project* visualization. The second visualization investigates the effect of the improvements, referred to as *post-project* visualization. The visualization is explained in Section 4.2. Furthermore, depending on the capability of the organization the visualizations have to be calibrated (i.e. thresholds for times have to be set to decide when something becomes a bottleneck), the calibration being explained after the two visualizations have been presented (see Section 4.3). In order to realize the visualizations measures have to be collected, which is explained in Section 4.4.

4.1. In-project Visualization to Identify Bottlenecks

An overview of the in-project visualization is provided in Figure 3. The in-project visualization is represented through a web-based tool that can be accessed through the web-browser. Figure 3 shows what the practitioners see on their screens. The different elements of the visualization are highlighted by circled numbers one to four. In the following the elements are explained in more detail.

① *Activities*: On the left-hand side of the tool the different activities are shown. Requirements are associated to the activities through states (e.g. an item is in the state “RS/US ongoing” meaning that requirements/user stories are specified). If an item passes a control point (e.g. “finalized RS/US”) it is moved to a new state.

② *Work in progress status*: The work in progress is shown as horizontal bars, each bar belonging to an activity. With regard to the bars the following should be observed:

- Each bar has three different color categories, indicated by the light grey, white, and dark grey. When the visualization is in live-operation the number of requirements in each category is stated. Requirements in light grey have exceeded the threshold duration within that activity. Requirements in the white category provide a warning that the duration is getting closer to the maximum threshold. The dark grey category indicates that the item is well within threshold time. Each bar at the same time shows the relative distribution of requirements within each activity between the color categories. As an example, in the activity “QSM ongoing” only a small portion of requirements is above the threshold (light grey), few are approaching the threshold (white). The majority of requirements (over 50%) is well within threshold time. On the other hand, the majority of requirements that reached the release project are above the maximum threshold (light grey).
- In order to make the bottlenecks explicit and identify reasons for their presence further detail is needed. That is, only knowing the distribution and the number of requirements in the categories is not enough, we also need to know more about the requirements within each category. Therefore, the visualization allows to open up each activity to reveal further detail (+sign on the right side of activity name). In the detailed view each box represents one specific requirements, the boxes have different height indicating estimated development effort (given that the calibration has shown that small/medium/large requirements have different lead-times and hence require setting different maximum durations). In live operation each box states the requirements ID. An example of this is shown for the activity “RS/US ongoing”.
- Further detail is provided by selecting one of the requirements, showing e.g. information regarding the associated system, release, entry date, description of the requirement, and comments. In the Figure, this is shown in the box stating “PMR id 9999-015”.

③ *Viewpoints*: On the right-hand side of the visualization viewpoints are provided. The viewpoints are grouped into nodes (which is the term for systems in the company), releases, and teams. It should be noted that below the nodes/system viewpoints there are two options, these are used to select all systems that belong to system of systems (SysoS), the system of systems concept

Figure 3. In-project Visualization to Identify Bottlenecks

being further elaborated in Section 5.1. The viewpoints allow the visualization to answer questions such as: What is the situation with respect to bottlenecks

- on the overall SysoS level?
- on the single system level?
- for a specific team?
- for a sub-set of teams working for a specific system?
- for three road-mapped releases of a specific system and the overall SysoS?

Different stakeholders can filter the data based on their interest/responsibilities. That is, the team leader can focus on his/her team, while the product manager can focus on the overall SysoS level. Due to the filtering ability the visualization becomes independent of individual project/release plans and depending on the viewpoint stakeholders can look at what they are interested in. This is specifically important for the SysoS context as we argued earlier in this paper.

4.2. Post-project Visualization to Investigate Effects of Process Improvements

The second visualization illustrating the effect of improvements is presented in Figure 4.

The bars illustrate the number of requirements outside the threshold range in comparison to the number of requirements inside the threshold range within a specific period. Threshold range here means that at least 50 % of the requirements should be completed in x days within the specified time period(s).

Based on the achievement several levels are defined: stretched, committed, robust, and alarming. The levels are defined according to the percentage of requirements that have been completed within the threshold. The levels are a means of structuring the achievement and being able to report the levels in form of a meter for balanced scorecards [9].

- *Stretched*: The threshold has been excelled in terms of achievement.
- *Committed*: The threshold committed has been achieved.
- *Robust*: The threshold has not been reached, but is within reach.
- *Alarming*: The threshold is well out of reach.

Figure 4. Post-project Visualization showing Effects of Improvements

Figure 5. Lifecycle of Visualization Usage and Calibration

As mentioned earlier, the analysis is done for specified time periods, as throughput is defined as the rate of completion within a time range. In order to track improvements over time different periods should be compared with each other. That way the previous period(s) serve as a baseline to determine if improvement actions led to the desired improvements. As can be seen in the example in Figure 4 an improvement was achieved between the two periods from robust level (close to alarming) to committed (while being close to stretched). The figure also shows that improvements are not only visible in between levels, but also within levels, which has to be considered when looking at the meters in balanced scorecards.

Different organizations have different capabilities, hence there is a need to calibrate the visualization to the specific organization in which it is used.

4.3. Calibration

The calibration focuses on deciding the threshold lead-time in which at least 50 % of the work items are to be completed. Thereafter, the calibrated visualization is used to identify and resolve bottlenecks in the process. For specified periods, the improvements achieved through bottleneck removal are evaluated using the post-project visualization. Given that the capability of the organization changes over time the visualization requires (re) calibration in time (see Figure 5 for an illustration of this cycle).

The calibration of the visualization should be done in a dialog with the concerned stakeholders in the organization that are able to influence the lead-time/throughput by doing something about identified bottlenecks. Before setting the thresholds, it is important to have facts about the distribution of lead-times for different process activities. This analysis helps to identify undesirable cases (requirements taking a long time). Thereafter, similar requirements in type (e.g. complexity) should be identified that are good references for comparison, and are good performers. They give a good indication of whether a desired threshold can be fulfilled. A similar principle is applied by

Table I. Example Factors Influencing Threshold Setting

Factor	Description
Process	Length of iteration cycles, time boxes, length of testing cycles
Size	Size of the work items, e.g. function points as an estimator for size
Effort	Effort needed to develop the system (e.g. person hours)
Impact	Number of systems affected by a work item increases complexity
Priority	Importance/value of the work item for the customer

Data Envelopment Analysis, identifying the best reference projects to compare to in benchmarking (cf. [22]). After having the facts, they should be used as input for discussion with practitioners, as otherwise the thresholds are not set based on facts, but might be determined by political factors (e.g. achieving a far too lenient threshold to be evaluated as a good performer) instead.

It is also important to highlight that not all requirements (e.g. low effort vs large effort requirements) aim at achieving the same threshold. Table I shows a list of example factors that we found to play a role in the studied organization as the experts in the organization suggested that the lead-times will vary for these factors. In other organizations further or different factors might be of relevance when deciding on thresholds, depending on different organizational contexts. When setting the thresholds, for example, large and complex requirements have more lenient thresholds than small/easy to develop requirements.

In order to assess whether the classification of requirements based on thresholds is realistic, two qualitative factors are of importance. One factor is acceptance of the classification by the practitioners. If they, for example, feel that the thresholds are set in a way that they are unachievable, there will be no acceptance for the visualization in the organization. A second factor is whether the way the thresholds are set provides indications for improvement potential, which is the overall aim of the visualization. If, for example, all activities only have requirements above the threshold all the time, it will not be possible to detect which activity is the bottleneck (too strict thresholds). On the other hand, if there are only requirements below the threshold, the thresholds might be set too lenient, as they do not allow detecting any bottlenecks. In order to assess whether the thresholds are realistic one could also use simulation to play through “what-if” scenarios of what happens to the time requirements reside in an activity depending on improvements achieved in different activities (cf. [8]).

4.4. Required Measures

In order to realize the approach, control points and lead-time need to be captured. A control point is defined as an exit criterion when a work item changes its state[17], e.g. a user story might change from state “in specification” to state “in design and architecture” at control point “user story review passed”. The lead-time is determined as time in state (*TIS*) based on the transition date (*TD*) for checkpoint *q* and work-item *i*, or the current date (*CD*).

$$TIS_{qi} = \begin{cases} TD_{(q+1)i} - TD_{qi} & \text{if } TD_{(q+1)i} \text{ exists} \\ CD - TD_{qi} & \text{otherwise} \end{cases}$$

Further details on control points and lead-time calculations are presented by Miranda and Bourque [17]. In addition categorical data have to be collected to allow for filtering (e.g. which product a work item belongs to).

4.5. Summarizing the Design Choices

After presenting the visualizations we provide the rationales for the design decisions made. In order to be able to detect, proactively, the bottlenecks we used different coloring schemes to categorize requirements based on the duration they are in a specific state. When requirements are in the yellow state and start approaching the red state, this indicates that they will become a bottleneck soon. Hence, the practitioner will be enabled to early identify future bottlenecks.

In order to address the system of systems situation, we propose different viewpoints to be integrated. As very many requirements are to be handled, the goal is to provide an overview of the distribution with respect to green, yellow and red requirements, which represents the aggregated view. Then if the practitioner is interested in details, there also is a detailed view where every requirement is shown (see Figure 3). Individual requirements then can also be characterized by relevant factors that lead to different thresholds. In this case different sizes are indicated through the height of the boxes representing the requirements.

Overall, the implementation made the whole system of systems used at the company browsable.

4.6. Bottleneck Resolution

Bottleneck resolution is not a part of our solution, as this work focuses on the detection of bottlenecks. Bottleneck resolution encompasses understanding the reasons for the bottlenecks, identifying improvement actions (improvement planning), and implementation of improvement actions, and following up on the effect of the improvement actions. Overall, this is in-line with general process improvement approaches. Hence, various approaches proposed in this context are of relevance, such as a light-weight approach for process improvement [26], or more specific approaches for root-cause analysis, such as Lehtinen et al.[12].

In the context of analyzing bottlenecks in the software development process, simulation has been used in an industrial context, as presented by Höst et al. [8]. The simulation allows to assess “what-if” scenarios on the effects of bottleneck resolutions.

The actual effect of bottleneck resolutions can be assessed with the post-project visualization illustrated in Section 4.2.

5. VALIDATION

The goal of the validation was to evaluate the usefulness of the approach in an industrial context and to find ways of further improving the proposed visualizations. First we describe the research context in which the study was conducted. Thereafter, we describe how the research was conducted based on the technology transfer model proposed by Gorschek et al. [7] consisting of static and dynamic validation, followed by the results of both types of validation.

5.1. Research Context

The case description and context are described, as this allows for generalizing the results to a specific context. Other companies in a similar context are likely to find the results transferable to their context [24].

The process used at the company followed a systems of systems (SysoS) approach. There is no common definition of systems of systems, as the term has been defined in different domains, such as military, enterprise information systems, or education [11]. The term has been recently established in the software engineering field, where a system of systems should fulfill several of the characteristics shown in Table II. The items ticked are fully fulfilled by the studied company. Operational independence and managerial independence are partially fulfilled due to technical dependencies (e.g. in some cases a feature in one system has to wait for a feature in another system to be ready). Furthermore, program managements have to collaborate in order to be able to present a complete solution later. This is also referred to as a directed system of systems [14].

The overall architecture of the system of systems studies consisted of 12 systems. The process used at the company is shown in Figure 6. In the first step the high level requirements (HLR) for the overall SysoS were specified. Before the requirements were handed over to compound system development a so-called “Go”-decision was taken, meaning that development resources were allocated to the high level requirement. When the decision was positive, teams wrote a detailed requirements specification (DRS), which then was handed over to the concerned system(s). The requirements were then implemented for a specific development system, and they were integrated (also called system level test of last system version – LSV-test). The development was done in sprints

Table II. Systems of Systems Approach Characteristics (cf. [11])

Characteristic	Studied Company
Operational independence	(√)
Managerial independence	(√)
Integration of system into systems of systems	√
SysoS comprised of complex systems	√
System suppliers deliver systems for integration	√
Complete technical oversight of SysoS and system supply	√

Figure 6. System of Systems Development Process at Ericsson

run by agile teams (AT), referred to as AT Sprints in Figure 6. Each system could be integrated independently of another system, which provides some degree of operational and managerial independence (cf. Table II). However, the versions of two systems had to be compatible when the system of systems was integrated (Compound System Test). Each of the systems was highly complex, the largest system having more than 15 development teams. The size of the overall system of systems measured in lines of code (LOC) was 5,000,000 LOC. This fulfills the characteristics of SysoS development related to system complexity and integration. In order to make sure that the system of systems was working together in the end, an overall system structure and design was developed, referred to as the anatomy. This allowed to have oversight of the overall SysoS, also making explicit how each system in the SysoS contributes to the overall system goals.

Looking at other context elements [24] the following should be added as information: All systems are older than 5 years; the development process is incremental with projects adding increments to the code base-line on system and compound system level; within the teams and in the testing activities agile practices are used, such as: continuous integration, time-boxing with sprints, face-to-face interaction (stand-up meetings, co-located teams), requirements prioritization with product backlogs, refactoring and system improvements; the system of systems is developed globally (with some development units being located in India and some in China).

Prior to the introduction of the new visualizations the company already had cumulative flow diagrams in place to assess bottlenecks. The new solution has been designed and originated by the authors to aid the company in early bottleneck detection.

5.2. Static Validation Conduct

Prior to the static validation every participant received a presentation of the first visualization for bottleneck identification. In addition the participants had access to the visualization to try it based on their viewpoint (e.g. the system they were responsible for).

As a means for data collection a questionnaire form was used. The sampling strategy followed was convenience sampling [20] and sampling for diversity [20]. Diversity means that we would like

Table III. Respondents

Role	Responsibilities	Responses
Project Mgt.	Motivation of teams, time and resource management of projects	5
Product Mgt./Req.	Requirements road-mapping, requirements analysis and specification	1
Line Mgt.	Resource management and allocation	4
Impl./Quality Ass.	Implementation and testing/review	3
Config. Mgt.	Versioning, Traceability, Documentation	1
Program Mgt.	Responsibility on system level across several projects, drive improvements for system development	6
Techn. Mgt.	Responsible for coherent system architecture and consistency	3

Table IV. Experience in years (y) in management and development activities

Role	]0;1]y	[2;5]y	[6;∞]y
Project Mgt.	1	8	4
Product Mgt./Req.	2	7	0
Impl./Quality Ass.	1	9	7
Config. Mgt.	0	0	1
Process Impr.	4	1	0

to gather feedback from persons with a variety of experience levels, and across different roles. That is, the goal was to capture a wide spectrum of perceptions in the form of a survey. The survey was sent out to 36 persons, of which 23 answers were received. The persons were chosen by a representative of the company having good knowledge about persons and their roles. The sample was further restricted as only persons with knowledge about the visualizations could be asked who received the presentation (i.e. the 36 persons the survey was sent to). An overview of the current roles of the persons is shown in Table III. The distribution shows that the key roles within the company were covered by several persons, while product management/requirements and configuration management were only covered once.

The experience/background of the respondents is shown in Table IV. Even though in their current role few respondents were working on requirements engineering at the time of the survey, nine of them had experience in this area. From an experience point of view project management, product management/requirements, implementation/quality assurance, and process improvement were covered by several respondents, while only one person responsible for configuration management responded.

The questionnaire used to collect feedback was structured as follows:

- Personal Information: This part captured the current role and system which the respondent is working on.
- Presentation: This question checked whether the presentation helped the respondent in understanding the visualizations.
- Usefulness: Three different opinion questions were asked regarding the usefulness of the visualizations in improving a) lead-times, b) throughput, and c) software quality. The answers to the questions were given on Likert scales. Furthermore, the respondents were asked to provide free-text answers motivating the rating on the scale.
- Acceptance: In the last question it was asked whether the respondent would like to use the visualizations on his/her product. The answer was also to be provided on a Likert scale in combination with a free-text answer.

The questionnaire can be found in appendix A.

The responses of the survey were analyzed through descriptive statistics. The qualitative data (free text answers) was summarized by coding [28]. The text was codified (i.e. codes are assigned to text snippets) according to the first level codes stated in Table V.

For example, the code “bottleneck” would indicate that the respondent talked about bottlenecks. In connection with the other codes we could identify whether a positive aspect, a benefit (B) or a liability (L) related to bottlenecks was mentioned. In addition to the defined first level codes second level codes were identified. The second level codes were identified using the following steps:

Table V. Coding Scheme for Qualitative Analysis

Code	Description
B	The code indicates that the statement/text refers to a benefit resulting from the introduction of the visualizations.
L	The code indicates that the statement/text refers to a liability/drawback expected by the introduction of the visualizations.
C	The code indicates that the statement/text refers to a condition/pre-requisite that has to be fulfilled in order to benefit from the visualizations.
L	The code refers to B/L/C with respect to lead-times.
T	The code refers to B/L/C with respect to throughput.
Q	The code refers to B/L/C with respect to quality (in terms of defects).

- S1: Identify a statement and create a code for the statement and log the code.
- S2: Identify the next statement and determine whether the code already exist. If so, log using the existing code. Otherwise, create a new code and log it.
- S3: Repeat step S2 until all statements are coded.

In total, 18 low level codes have been identified.

5.3. Dynamic Validation Conduct

The quantitative data was based on control points described earlier. The control points were captured in a corporate database. The existing control points showed which state a work item (in this case high level requirement) was in. In the organization specific persons have been assigned responsibility of changing states at control points. The responsibilities were assigned in a way that people responsible for specific control points had an interest in the data, which motivated data providers keeping their data up-to-date. For example, test managers were responsible to set control points when testing had been successfully completed. The test manager was interested in doing so as the data provided a good overview of the work in progress, and hence provided transparency to the test manager.

The analyzed data has been collected for a duration of 15 months after roll-out of the visualizations to reflect the improvement of the situation with respect to post-project visualization focusing on assessing improvements achieved. In the analysis of the data, factors considered as crucial in the context of the studied organization were used to provide an analysis from different angles, that is:

- Requirements were grouped based on effort (labeled as small, medium, large). The groups are defined in intervals based on estimated technical work hours.
- Requirements were grouped based on complexity. That is, requirement belonging to a single system were in one group, and requirements belonging to a system of systems consisting of a set of systems were in another group.
- All requirements in the studied period were assessed without grouping them according to characteristics (effort and complexity).

These factors were used for calibration of the visualization and a preliminary analysis is presented by Petersen [21]. In addition to the aggregated analysis (all requirements) the post-project visualization was also used to analyze the performance by factor as the factors seem to play an important role with respect to whether bottlenecks are fixed (e.g. it might be harder to remove a bottleneck for very complex requirements in comparison to very simple ones).

5.4. Results of Static Validation

The results of the static validation are first presented based on the ratings given by the respondents, as shown in Figure 7. The rating shows that the strongest agreement is given for the statement that the visualization for bottleneck identification helps in improving lead-time. With respect to throughput the majority thinks that throughput will be improved, but there is also a high number that are not expecting a change here. Finally, with respect to quality the responses are divided into two halves, one half expecting improvements, while the other half thinks that the visualizations do not help in improving quality.

Figure 7. Rating of Expected Improvements for Lead-time, Throughput, and Quality

5.4.1. Benefits The most commonly mentioned benefit (11 respondents) was that the visualization is able to determine possible bottlenecks with a neat visualization and filtering mechanism. As one respondent stated, the solution “*certainly gives an overview of areas where accumulation is there or movement is not up to the mark, or there is batch behavior*” (note from the author: batch behavior refers to having unmanageable peaks of workload, which should rather be distributed more evenly in time).

Another common benefit mentioned by 9 respondents was the support given by the visualizations in planning and prioritizing work. One aspect raised was that the number of things to be worked on can be better planned, as was pointed out by one respondent saying that “*this can also help us determine the optimal number of things that is good to leave in the system, instead of clogging the system with too many things (for example, not taking on new requirements when there is a sufficient number already being worked on).*” The visualizations also help prioritizing as activities (e.g. test) “*would have a better clarity in terms of what would be delivered to them, they can do the right privatization and planning accordingly.*” According to a respondent the visualization also helps to pace the development flow end to end, saying “*the visualizations show the real picture in black and white to internal as well as external stakeholders from system perspective. Hence, it gives a push to all stakeholders involved to move requirements further with the right pace.*”

Furthermore, 8 respondents acknowledged that the visualizations help in taking improvement actions to increase lead-time as well as throughput. Different types of actions are mentioned, i.e. reactive actions (e.g. when things already missed the lead-time threshold), proactive and preventive actions (e.g. when things marked as yellow approach the lead-time threshold).

Other lead-time and throughput related benefits mentioned by fewer (1 to 4) respondents were that the visualization helps in tracking progress and provides the big picture, shows more detail than the visualizations previously used (cumulative flow diagrams), and clarifies the contribution everyone makes to the product (e.g. due to filtering on team level).

The previously presented benefits have been mentioned in relation to lead-time as well as throughput.

With respect to quality, many respondents (7) could not make a connection between the information shown in the visualization and the quality of the product in terms of defects. They stated as a reason that the visualizations do not show failure/fault statistics. This also explains that the subjective rating on the Likert scale showed that people disagreed on the positive effect of the visualizations on quality. However, those seeing a positive effect provided different explanations. Low input quality to testing will show in longer time needed to run the tests (e.g. due to breaking test builds). Furthermore, ongoing defect fixing in development teams can become visible as a

bottleneck as they take time away from other development activities. One respondent was pointing out that by having the visualizations in place *“the pressure is put on early phases to think within time limits. This creates pressure, which helps in quick turn around times. Early starts mean early fault detection and time for rework when required.”*

5.4.2. Liabilities One liability mentioned by a respondent was that the visualizations generate pressure in the organization, stating that *“if not properly managed it [the visualizations] can create more pressure and stress in the organization. Stressed people do not think clear. To reach the goals people can do bad short cuts to satisfy management/leadership.”* This statement is in clear conflict with the statement presented in the previous section, saying that pressure generates quick turn around times and helps in early fault detection.

Another liability mentioned by 3 respondents was that the visualizations do not show the flow of requirements in a continuous manner, as do cumulative flow diagrams. Here it is important to mention that the measurements (control points and time in state presented earlier) provide all the information needed to create cumulative flow diagrams, which then can be used to provide a different view of the flow of requirements, as is discussed by Petersen and Wohlin [25].

Two respondents stated that the categorical data (i.e. having three categories – green, yellow, and red based on thresholds for time in state) could be improved by having a continuous time scale showing exactly how long a requirement has been in a specific phase. However, no good mechanism for visualization for that purpose has been identified as of yet.

5.4.3. Conditions As one respondent pointed out the visualizations *“will not lead to improvements by themselves.”* Conditions have been identified by the respondents that are considered to be important in order to utilize the full potential of the visualizations.

In order to have a positive effect the visualizations should be used to improve and not to punish, one respondent saying that *“it is important though that we act on findings in an analytical way and not in a punishing way. Instead carefully look into what is the reason/background for a specific situation and start to focus on improvement in that area, not start to punish the area where the problems become visible”*. The respondent added that it is important to consider that *“the problems can be created within other areas than where they pop up.”*

The same respondent was also saying that often in practice work is done in a lack-of-time mode (e.g. due to pressure with respect to release dead-lines), though in order to continuously improve there needs to be more time to reflect on what the data is showing.

Another condition mentioned was that knowledge about the visualizations needs to be shared on all levels of the organization, including e.g. program management, project management, and so forth.

5.5. Results of Dynamic Validation

5.5.1. Calibration. Before the visualization was used in live operation it had to be calibrated. As has been shown in the investigation in our previous publication [21] sowed differences in lead-times with respect to categorization into small, medium, and large requirements showed differences, while the complexity in terms of how many systems are involved in developing the requirement did not. As a consequence the calibration of the visualization led to three different threshold durations for small, medium, and large requirements. It should be pointed out that this was the baseline setting, that was based on a limited number of data points.

The calibration was discussed with decision makers in order to decide on a threshold duration that was perceived as realistic, but challenging. This was done in a working group set up for threshold setting, as well as in informal discussions run by the person in the organization being formally responsible for the visualization. The final decision for the threshold duration was agreed on by all stakeholders considered as being critical stakeholders (system managers/technical perspective, program managers, line managers, project managers).

Figure 8. Example of Identifying Bottlenecks with Visualization

5.5.2. *Bottleneck Identification (In-project)*. After having calibrated the visualizations the first visualization was used to identify bottlenecks. Figure 8 shows the visualization for one specific release for a system of systems.

Four activities are shown, “RS/US ongoing”, “Design and FT”, “LSV”, and “Release Project”. On the left we indicated the number of requirements that are fine (light grey) in relation to the total number in the activity. For example, for phase RS/US ongoing, only 1 out of 31 requirements (3%) is in control given that the requirement is below the lowest threshold value. 6 out of 31 requirements approach the threshold maximum threshold, 24 out of 31 are out of control, being above the maximum threshold. Bottlenecks are identified considering the relative distribution of requirements.

Ranking the activities with regard to where the main bottlenecks are, “RS/US ongoing” is on rank one, “LSV” second, and “Design and FT” third, and “Release Project” fourth. RS/US ongoing and LSV are ranked one and two, respectively, given that in RS/US ongoing only 1 out of 31 requirements is in control, while in LSV 2 out of 21 requirements are in control. In comparison, “Design and FT” is not considered as being as significant given that 12 out of 30 requirements in this activity are in control. In the activity “Release Project” almost all requirements are in control (10 out of 11).

The reason for the bottleneck in “RS/US ongoing” was that some systems concerned with a requirement did not provide a statement of compliance saying that they understand the requirement and that it can enter design and function test. The reason for the bottleneck in the LSV phase was that some systems were waiting for other systems to be verified on the system level, so that they can now be integrated on the system of systems level. The visualization in Figure 8 also shows progress information, e.g. it shows that a total of 11 (all requirements in release project) out of 93 requirements in total (31 in RS/US ongoing, 30 in design and FT, 21 in LSV, and 11 in Release Project) have been verified. We can also see that 15 (1 + 12 + 2) requirements are green out of 82 (31 + 30 + 21) that are currently in development (currently in development includes the phases RS/US ongoing, design and FT, and LSV).

5.5.3. *Evaluating Improvements with Respect to Bottlenecks (Post-project)*. Figure 9 shows the results of the post-project visualization evaluating achieved improvements. The figure shows the result for all requirements, requirements belonging to single systems (54% of all requirements) in comparison to SysoS (45% of all requirements), and small (53% of all requirements)/medium (30%)/large requirements (17%). In the sub-figures we also state the throughput in each time period

Figure 9. Results Shown through post-project visualization. The numbers below the subfigures show the % of requirements that were finalized in Q1, Q2, etc. from the total number of requirements finalized in the overall time period.

as the percentage of requirements being completed (100% is all requirements completed from the first quarter (Q1) in the first year to (and including) Q1 in the following year.

Figure 9(a) shows the results of the visualization for all requirements. An improvement trend is visible from Q1 in the first year to Q1 in the following year. Commitment level is reached in Q3. In Q4 there is a reduction in performance with respect to lead-time. Q1 shows the best result so far, achieving commitment level. Q2 and Q4 have the highest throughput.

Figure 9(b) indicates that single system requirements achieved the threshold in all time periods. Q3 and Q4 showed a reduction in throughput while still meeting the threshold. The situation improves again in Q1 of the following year. The highest throughput is achieved in Q4.

Figure 9(c) illustrates the situation for requirements belonging to a SysoS. In Q1 the situation is classified as alarming based on the visualization. A continuous improvement trend in Q2, Q3 and Q1 the following year is visible. The exception is Q4 where the lead-time increases significantly. The highest throughput is achieved in Q4.

Figures 9(d), 9(e), and 9(f) show the situation for requirements of different efforts (expressed in terms of technical person hours). As mentioned earlier depending on estimated effort different thresholds have been set. A stable situation is shown for small requirements with minor changes between time periods, the most noticeable increase in lead-times being in Q4. The highest throughput for small requirements is achieved in Q2 and Q4. For medium effort requirements an increase in lead-time is visible while the situation improved in Q1, first time reaching committed level. For large requirements the situation is continuously improving with a clearly visible reduction in Q4.

In Section 6.2 we discuss the results of the visualization, and also present the main root-causes identified in the company, as well as the improvement actions taken.

6. ANALYSIS AND DISCUSSION

6.1. Static Validation

There was a high agreement between respondents that the visualizations help driving improvements with regard to lead-time and throughput. None of the respondents disagreed with that. With respect to quality the situation was different as some respondents say the visualizations help to improve, while others disagreed (see Figure 7). Those that said there is an improvement saw a connection between poor quality and disturbance in development flow. Those that did not see that connection stated that the visualization does not contain quality information, such as fault statistics. Overall, the connection to quality can only be made by taking time to reflect on the data identifying root-causes for undesired behavior (see Section 5.4.3), while bottlenecks are immediately evident from the visualizations. A similar observation has been made in another study where one of the hindering factors identified was that software process improvement activities are perceived as getting in the way of the “real” (i.e. operational development) work [19], leading to a lack of time needed for reflection.

The results also showed that concerns regarding the visualizations were mainly related to how they are used. In particular, it was highlighted that the visualizations should not be used to generate pressure or to put blame on teams or people.

6.2. Dynamic Validation

Visualizations: We highlight some observations from the quantitative analysis shown in Figure 9, and state potential reasons explaining change in performance. Given that many confounding factors are present in a large organization [31] the reasons stated might only partially contribute to changes in lead-time and throughput over time.

In all figures it is apparent that there is a reduction in performance with respect to lead-times in Q4. The reason for the reduction is summer vacation in Sweden where there is only a small number of persons working during July. This leads to increased lead-times due to that things are finalized in Q4 that would otherwise have been finalized in Q3, which also explains the higher throughput in Q4.

Another interesting observation is that requirements related to single systems are finalized within the desired thresholds easily, as in almost all periods (except Q3 and Q4) the company was close to achieve stretched status in the KPI. Requirements belonging to a SysoS are further away from achieving the threshold, which was recognized as a challenge, and hence the company initiated improvements to resolve bottlenecks.

Root-Causes: From previous root-cause analysis investigating reasons for bottlenecks we know that three main issues have been the reason for bottlenecks in development flow, namely 1) overloading the development organization with work by allocating too many tasks for the given resources, and 2) dependencies between systems and lack of compatibility (e.g. if one system is upgraded to a new version, it can not be integrated with another system due to that this system is not upgraded yet), and 3) barriers in communication.

Improvements: These issues have been resolved by implementing a pull-system where the development organization can pick requirements from a prioritized requirements list when resources are free (see Kanban [10] for further information regarding pull systems). In addition, technical solutions for increased compatibility between system versions have been implemented. To address communication different roles have become part of a team (more cross-functionality). It is also apparent that large requirements improved most with respect to lead-time in comparison to small and medium requirements. The reason for that was that even though the requirement is classified as large, requirements were broken down into smaller parts reducing the effort needed to implement them. This was an initiative started before the first quarter considered in our investigation.

Improvement trends/patterns in throughput are not visible with Q4 having the highest throughput overall due to summer vacation (see Figure 9(a)). One explanation is that the organization focuses on continuous work and integration (an agile practice implemented in the organization) avoiding

to overload the organization at one point in time, and that the improvements implemented require more time to become visible in the data.

6.3. Overall Implications

From the qualitative as well as quantitative data analysis the following lessons learned have been identified:

Useful as a bottleneck detector: The static validation has shown the visualizations are perceived as a useful bottleneck detector (see Sections 5.4 and 6.1). The dynamic validation has shown that shifts in lead-time and throughput are detected. For specific types of requirements (SysoS and large requirements) clear improvement trends have been identified (see Sections 5.5 and 6.2).

Understand factors leading to variance in lead-time/throughput: We found that lead-time improved differently depending on estimated effort and whether requirements belong to a single system or SysoS. These factors have been identified by experts in the studied organization and are therefore context dependent. In order to support organizations taking the most relevant factors for their context into consideration further studies are needed determining which factors lead to variance in lead-time/throughput. Similar investigations have been done in the past, but have been focusing on software productivity instead of lead-time [22]). Overall, knowledge about relevant factors is important for calibration (different thresholds for different types of work items) and evaluation of improvements using the second visualization (post-project).

Visualization needs to benefit practitioners: During the static validation practitioners highlighted that metrics should not be used as a punishment system, but rather should aim at improving (see Section 5.4.3). We believe that in order for practitioners to use the visualizations for improving they need to directly benefit from the data in their daily work, which is supported by literature as well [6]. Data providers need to understand how the metrics/visualizations can be useful in their daily work as otherwise they are not willing to invest time in providing high quality data. We designed the visualizations in a way that by taking viewpoints through filtering in consideration different stakeholders, depending on the viewpoints they are interested in (product, project, release, activity), are motivated to submit and maintain data of high quality. In our study, practitioners were regularly updating the data, which is an indication that the visualizations are perceived as useful.

Management shall avoid generating pressure with visualizations: In the static validation respondents disagreed on whether generating pressure through the visualizations has a positive (see Section 5.4.1) or negative (see Section 5.4.2) effect. We have no evidence which alternative is better. However, aggregated evidence presented by Beecham et al. [3] suggests that pressure is a de-motivator. Instead, the visualizations should serve to highlight empowerment and responsibility by showing and highlighting contributions (e.g. what a team contributes to a product), recognition of good work and contribution, as well as trust and respect [3]. Another risk we see with generating pressure is that the visualizations are perceived as negative, and not as a support for improvement.

Setting thresholds strict or lenient: As discussed in Section 4.3, there is a risk that calibration is too lenient/strict due to factors for lead-time and throughput variance are not well understood. Threshold setting is important to be able to detect lack of performance, the thresholds being competitive and realistic and agreed with stakeholders. Goals that are too ambitious and unrealistic could be considered a common de-motivator for software engineers [3]. However, lenient thresholds still show movement in performance as long as they are not set too lenient (i.e. all requirements easily achieving stretched). Overall, this raises a new research questions, in particular how time constraints affect the performance of software engineers. This has been investigated on an individual level in inspections [15, 23], however, in future work evidence should be generated on organizational levels as well. That is, one could investigate the performance of an organization with lenient versus more strict thresholds.

Scalability: Given the high number of requirements in large-scale development, scalability of the visualization has to be discussed. In the case company, the requirements were distributed between systems, which reduced the overall number of requirements to be managed per system. Furthermore, the requirements were on an abstract level, and are containers for more detailed requirements. Overall, the requirements should be verifiable units of functionality. These are further broken down,

and implemented. As soon as a verifiable requirement changes state, this is to be documented by the persons responsible for keeping track of the progress. As an example, the tool screenshot of the visualization in Figure 3 showed over 60 requirements at a time in one phase (RS/US ongoing in the example of the Figure), which was an easy to situation to visualize and manage. The boxes, as described, also indicate the requirements IDs, and the persons responsible for the systems knew, just by looking at the ID, what each requirement was about. Overall, for the size of the system, people did not feel overwhelmed by the amount of information provided.

6.4. *Validity threats*

Threats to validity are grouped into four categories, namely external validity, internal validity, construct validity, and reliability [29].

External validity refers to the ability of generalizing the findings to different contexts. Results of empirical studies are valid for the contexts in which they were studied [32]. In order to generalize, we have taken care of describing the context in a detailed manner to aid generalizability. The context here is large-scale system of systems development where incremental and agile practices are used.

The application of the visualization is not constrained to a particular organization, but modifications might be necessary. For example, each organization has to identify filters and viewpoints relevant for their context and calibrate the visualization depending on its goals and capabilities. Future replication in very different contexts (e.g. different domain, scale, development process, etc.) is needed to further strengthen the generality of the results.

The main prerequisite for an organization to implement the visualizations presented in this paper has to use of state-gate models, where one keeps track of when a requirement changes status in the life cycle. An organization, before using the approach, hence has to decide on points in the process where the states change. Which points to choose is up to the organizations and depends on their processes. If working in a highly iterative way, one could also, for example, define a state “50% done”. In this study the approach was applied to requirements. However, it could also apply to other artifacts (e.g. defects to assess bottlenecks in the defect resolution lifecycle), or any other process.

Internal validity is concerned with casual relationships and the ability to make inferences[32]. In this case a threat to internal validity is whether the improvements in lead-time can be attributed to the visualizations. Given the large scale of the organization and many improvements going on in parallel to the visualization implementation, we are not able to make an inference between improvements achieved and the use of the visualization. However, we have demonstrated the ability of the visualization to identify and assess the organization’s bottlenecks based on feedback and actual data.

Construct validity is concerned with obtaining the right measures for the concept being studied [32]. One threat is that the outcome is highly dependent on the selection of people. Hence, we selected the persons providing feedback in the static validation according to diversity in roles/viewpoints. A practitioner in the organization supported us in the selection of people to survey. Another risk is that the presence of the researcher influences the response given by practitioners. However, this risk is reduced given that the first author is partially employed at the studied company. Questions might be misunderstood by the practitioners. To reduce this threat the survey was reviewed and piloted prior to sending the survey. With respect to the quantitative data there is a threat that the data is incorrect due to lack of updates or mistakes made by the practitioners in entering the data. The risk is reduced as the data used in the visualization is continuously used in the operative work by the practitioners, hence they have a stake in keeping the data up-to-date and correct.

Conclusion validity/reliability is concerned with replicability of the study, i.e. when repeating the study the same result should be obtained [32]. One risk to conclusion validity are confounding factors not controllable or identifiable by the researcher. Confounding factors are important in interpreting the results (e.g. with respect to reasons for changes in the visualizations). In order to reduce the risk we verified our interpretation with one practitioner at the company being responsible for driving the implementation of the visualizations at the company. The practitioner has a good overview of ongoing improvement activities in the organization, and hence was a good source to

verify our interpretations of the data, and to confirm whether the trends identified in the visualization are in line with her observations. Overall, the practitioner agreed with our observations indicating that this validity threat is reduced (but not eliminated).

7. CONCLUSION

This paper presents and evaluates visualizations for assessment of bottlenecks in very complex system of systems development with many requirements being developed in parallel. The visualizations have been designed to cater for the need of system of systems developed. The evaluation of the visualizations was based on the technology transfer model by Gorschek et al. [7] who propose to conduct a static validation (collect feedback prior to roll-out of a solution) followed by a dynamic validation (use of the solution).

The result of the static validation showed that practitioners agreed on the usefulness of the visualization to improve throughput and lead-times, while practitioners had different views on whether quality is also improved. Some practitioners could not connect the visualizations to quality as no fault data is included in the analysis. However, quality problems might indirectly show in the form of bottlenecks and hence might be discovered when reflecting on the data. Problems related to the visualizations were mostly related to how they should be used. We conclude that the visualizations need to be used to improve and not to punish or generate pressure, and that sufficient time has to be taken to reflect on the results shown by the visualizations (root-cause analysis).

In the static validation we analyzed data in time including 15 month (5 quarters) starting after the roll-out of the visualizations being used in the company's scorecard. The results showed that the visualizations were able to identify bottlenecks and assess whether improvements in lead-time and throughput have been achieved. Improvements in lead-time were visible for some types of requirements (system of systems and large requirements) while no clear improvement trend was visible for single system requirements, and small/medium requirements. Hence, it is important to filter the data according to factors (here single system vs. system of systems development and estimated effort) in order to gain a richer understanding of trends in the data.

Overall, we conclude that based on the perception by practitioners and the analysis of quantitative data in time the visualization is useful in identifying bottlenecks in software development.

The visualizations need to be further evaluated in different contexts (e.g. domains, medium/small-scale development, and different development processes).

8. ACKNOWLEDGEMENTS

We would like to thank all the participants in the study who participated in the research. Furthermore, we thank the anonymous reviewers for valuable comments that helped in improving the paper. This work has been supported by Ericsson AB and ELLIIT, the Strategic Area for ICT research, funded by the Swedish Government.

REFERENCES

1. D. J. Anderson. *Agile management for software engineering: applying the theory of constraints for business results*. Prentice Hall PTR, Upper Saddle River, N.J., 2003.
2. D. Arditi, O. B. Tokdemir, and K. Suh. Challenges in line-of-balance scheduling. *Journal of Construction Engineering and Management*, 128(6):545–556, 2002.
3. S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp. Motivation in software engineering: A systematic literature review. *Information & Software Technology*, 50(9-10):860–878, 2008.
4. J. Colombi, B. C. Cohee, and C. W. Turner. Interoperability test and evaluation: A system of systems field study. *The Journal of Defense Software Engineering*, 21(11):10–14, 2008.
5. H. Erdogmus. Tracking progress through earned value. *IEEE Software*, 27(5):2–7, 2010.
6. H. D. Frederiksen and L. Mathiassen. Information-centric assessment of software metrics practices. *IEEE Transactions on Engineering Management*, 52(3):350–363, 2005.

7. T. Gorschek, P. Garre, S. Larsson, and C. Wohlin. A model for technology transfer in practice. *IEEE Software*, 23(6):88–95, 2006.
8. M. Höst, B. Regnell, J. N. och Dag, J. Nedstam, and C. Nyberg. Exploring bottlenecks in market-driven requirements management processes with discrete event simulation. *Journal of Systems and Software*, 59(3):323–332, 2001.
9. R. S. Kaplan and D. P. Norton. *The balanced scorecard: measures that drive performance*. Harvard Business School Press, Boston, Mass., 2008.
10. C. Ladas. *Scrumban: and other essays on Kanban System for Lean Software development*. Modus Cooperandi Press, Seattle, WA, 2008.
11. J. A. Lane and R. Valerdi. Synthesizing sos concepts for use in cost estimation. *Systems Engineering*, 10(4):297–307, 2007.
12. T. O. A. Lehtinen, M. Mäntylä, and J. Vanhanen. Development and evaluation of a lightweight root cause analysis method (arca method) - field studies at four software companies. *Information & Software Technology*, 53(10):1045–1061, 2011.
13. G. Lewis, E. Morris, P. Place, S. Simanta, D. Smith, and L. Wrage. Engineering systems of systems. In *Proceedings of the IEEE International Systems Conference (SysCon 2008)*, 2008.
14. M. W. Maier. Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284, 1998.
15. M. Mäntylä, K. Petersen, and D. Pfahl. How many individuals to use in a qa task with fixed total effort? In *Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2012)*, Lund, Sweden, pages 311–314, 2012.
16. H. L. McManus. Product development value stream mapping (pdvsm) manual. Technical report, Center for Technology, Policy, and Industrial Development, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, USA, September 2005.
17. E. Miranda and P. Bourque. Agile monitoring using the line of balance. *Journal of Systems and Software*, 83(7):1205–1215, 2010.
18. S. Mujtaba, R. Feldt, and K. Petersen. Waste and lead time reduction in a software product customization process with value stream maps. In *Proceedings of the 21st Australian Software Engineering Conference (ASWEC 2010)*, pages 139–148, 2010.
19. M. Niazi. An exploratory study of software process improvement implementation risks. *Journal of Software Maintenance and Evolution: Research and Practice*, in print, 2011.
20. M. Q. Patton. *Qualitative research and evaluation methods*. Sage Publications, Thousand Oaks, 2002.
21. K. Petersen. An empirical study of lead-times in incremental and agile software development. In *Proceedings of the International Conference on Software Process (ICSP 2010)*, pages 345–356, 2010.
22. K. Petersen. Measuring and predicting software productivity: A systematic map and review. *Information and Software Technology*, 53(4):317–343, 2011.
23. K. Petersen, K. Rönkkö, and C. Wohlin. The impact of time controlled reading on software inspection effectiveness and efficiency: a controlled experiment. In *Proceedings of the Second International Symposium on Empirical Software Engineering and Measurement (ESEM 2008)*, Kaiserslautern, Germany, pages 139–148, 2008.
24. K. Petersen and C. Wohlin. Context in industrial software engineering research. In *Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement (ESEM 2009)*, pages 401–404, 2009.
25. K. Petersen and C. Wohlin. Measuring the flow in lean software development. *Software: Practice and Experience*, 41(9):975–996, 2011.
26. F. Pettersson, M. Ivarsson, T. Gorschek, and P. Öhman. A practitioner’s guide to light weight software process assessment and improvement planning. *Journal of Systems and Software*, 81(6):972–995, 2008.
27. K. H. Pries and J. M. Quigley. *Scrum project management*. CRC Press, Boca Raton, FL, 2011.
28. P. Runeson and M. Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 14(2):131–164, 2009.
29. P. Runeson, M. Höst, A. Rainer, and B. Regnell. *Case Study Research in Software Engineering – Guidelines and Examples*. Wiley, 2012.
30. Z. A. Sarraj. Formal development of line-of-balance. *Journal of Construction Engineering and Management*, 116(4):689–704, 1990.
31. M. Unterkalmsteiner, T. Gorschek, A. K. M. M. Islam, C. K. Cheng, R. B. Permadi, and R. Feldt. Evaluation and measurement of software process improvement - a systematic literature review. *IEEE Trans. Software Eng.*, 38(2):398–424, 2012.
32. C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Springer, 2012.

A. QUESTIONNAIRE

Q1: Current (primary) role/product:

Q2: In which areas of software development have you worked in and for how long (in years)?

Options: Requirements, Product management, design and implementation, quality assurance, project management, driver software process improvement

Q3: In your opinion, do you believe that the presentation helped you to understand the TiQ viewer (name of indicator 1 in the company) in a good way? (Yes/No)

☐ Yes ☐ No

Q4: The use of the TiQ viewer will help in improving our lead-times. Please explain your answer (free-text).

☐ Don't Agree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Fully Agree

Q5: The use of the TiQ viewer will help in improving our lead-times. Please explain your answer (free-text).

☐ Don't Agree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Fully Agree

Q6: The use of the TiQ viewer will help in improving our quality. Please explain your answer (free-text).

☐ Don't Agree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Fully Agree

Q7: I would like to see the TiQ viewer implemented and used in my product. Please explain your answer (free-text).

☐ Don't Agree ☐ ☐ ☐ ☐ ☐ ☐ ☐ Fully Agree