Visualization of Defect Inflow and Resolution Cycles: Before, During and After Transfer

Ronald Jabangwe, Kai Petersen, Darja Šmite

20th Asia-Pacific Software Engineering Conference

2013 Bangkok, Thailand

Visualization of Defect Inflow and Resolution Cycles: Before, During and After Transfer

Ronald Jabangwe, Kai Petersen, Darja Šmite
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
{*ronald.jabangwe,kai.petersen,darja.smite*}*@bth.se*

***Abstract*—The link between maintenance and product quality, as well as the high cost of software maintenance, highlights the importance of efficient maintenance processes. Sustaining maintenance work efficiency in a global software development setting that involves a transfer is a challenging endeavor. Studies report on the negative effect of transfers on efficiency. However, empirical evidence on the magnitude of the change in efficiency is scarce. In this study we used a lean indicator to visualize variances in defect resolution cycles for two large products during evolution, before, during and after a transfer. Focus group meetings were also held for each product. Study results show that during and immediately after the transfer the defect inflow is higher, bottlenecks are more visible, and defect resolution cycles are longer, as compared to before the transfer. Furthermore we highlight the factors that influenced the change in defect resolution cycles before, during, and after the transfer.**

***Keywords*-Global Software Development, Software Transfer, Maintenance flow, Efficiency, Lean**

I. Introduction

Maintenance work is a costly endeavor. A large portion of the total costs associated with the software product lifecycle is associated with maintenance work [1]. Almost 50% of the costs per function can come from maintenance work and over 40% of the software engineering labor force are most likely involved in maintenance activities [2]. These statistics highlight the costly nature of, and the importance of sustaining levels of efficiency during, maintenance activities. In addition, monitoring and improving the speed of maintenance work, e.g., shorter defect resolution cycle times, can help with controlling and improving software quality [3]. Thus it is important to understand the impact that project settings or strategies have on defect resolution cycles, and hence maintenance work efficiency so as to make informed decisions for future releases.

Strategies implemented in global software development (GSD) projects have been linked with a decline in efficiency [4], [5] and delays in task completion times [6]. Software transfer, a prevalent GSD strategy, can alter the development and maintenance setting. This can negatively impact efficiency levels, and consequently make it difficult to realize the potential benefits, such as cost reduction [4].

Software transfer is the relocation of, for example, software product development responsibilities, from one location to another [5]. There are studies that report on the negative impact of transfers on efficiency [4], [5]. We use a lean indicator, proposed by Petersen [7], to visualize the magnitude of changes in defect resolution cycles, before, during and after a transfer, during software evolution. Defect resolution cycles is used as an indicator of maintenance work efficiency. Furthermore, we elicited factors that led to changes in defect resolution cycles during the three periods.

To the best of our knowledge, there is currently no such systematic investigation using a similar lean indicator to visualize the magnitude of the effect on defect resolution cycles, using multiple releases. Our work is driven by the need to better understand the implications of software transfers on maintenance work efficiency.

The remainder of this paper is organized as follows: Section II outlines related work. Section III follows with a description of the study context and the research methods. Section IV follows with results from the quantitative and qualitative analysis. A summary and discussion of the study findings is presented in Section V. Validity threats and study limitations are discussed in Section VI. Lastly, the conclusions and future work are presented in Section VII.

II. Related Work

Though GSD related empirical studies are increasing [8], there are very few GSD studies that focus entirely on software transfers. In the few studies available some studies have focused on the suitability of products for transfers and understanding factors that impact efficiency during transfers.

Transferring work to a site with employees that are unfamiliar with the product, can result in a decrease in productivity. Based on a case study that was conducted at Lucent Technologies, Mockus and Weiss [9] report that this decrease can be linked to learning curves of a complex product. Their study highlights the importance of formulating risk mitigation strategies when engaging in transfers. Mockus and Weiss found that Lucent Technologies utilized, or were considering, certain transfer approaches, such as transferring by development stage, or by maintenance phase, as ways to alleviate transfer-related issues.

In a case study conducted at a large multinational corporation, Ericsson, Šmite and Wohlin [10] found factors that hinder efficiency, such as, product complexity, lack of training for the receiving site, or limited product experience of the receiving site. Their study is based on findings from two transfer projects. In the same study, Šmite and Wohlin also provide recommendations that can mitigate these factors, such as, early transfer planning and early recruitment, training, and prolonging support from the sending site to the receiving site. In a separate study, Šmite and Wohlin [5] provide a checklist in which they group "transfer critical factors" that can contribute to a successful transfer into either product-, people-, or process-related categories. The checklist is based on findings from four transfer projects, and it can help decision-makers, that are considering or are already engaging in transfers, with identifying mitigation strategies for factors that can negatively impact efficiency.

Our study differs from the aforementioned studies in that we use an evolutionary view to investigate defect resolution cycles before, during and after a transfer. Thus our study provides a much deeper understanding of the impact of transfers on maintenance work efficiency. A similar longitudinal perspective is used by Jabangwe and Šmite [11], in which they investigate the impact of software transfers on quality. Results from the study suggest an initial decline in quality after transferring work to a site with employees that have limited or no previous product engagement. The study is done by using an evolutionary view of defect data, across multiple releases, of a large commercial product from Ericsson. The study presented in this present paper differs from the study by Jabangwe and Šmite [11] in that the evolutionary and longitudinal view is used to investigate defect resolution cycles and not quality.

Petersen [7] illustrated the usefulness of four lean indicators for evaluating efficiency of maintenance work. The four lean indicators are defect inflow (referred to as maintenance flow in [7]), lead-time, workload, and cumulative flow diagram. Defect inflow is a measure of inflow per unit time. Lead-time is a measure of the length in time spent in a particular state, for example, time spent designing a solution for a defect. The workload indicator is a measure of the value-adding activities in, for example, maintenance work over time. A cumulative Flow Diagram (referred to as CFD from hereon in) provides a pictorial view of the rate with which a solution, for defects for example, is identified and implemented. Petersen and Wohlin [12] empirically evaluated CFDs on their usefulness in a software engineering context.

We use one of the lean indicators, the CFD. This is because, unlike with the other indicators, with a CFD we are able to visualize more than one aspect in the maintenance flow, i.e., defect inflow, periods of bottlenecks and waiting periods or lead-times. This gives us the means to analyze defect resolution cycles from different angles. An in-depth explanation and illustration of how to interpret cumulative flow diagrams when used as a lean indicator can be found in [7].

Overall, our study differs from other studies on two fronts. First, the use of a lean indicator that visualizes defect resolution cycles to understand implications on efficiency of maintenance work. Second, we use a longitudinal and evolutionary view of the lean indicator to investigate the impact of software transfers on defect resolution cycles. We did not find any other study that conducts a similar analysis.

III. Research Methodology

A. Purpose

The purpose of this study is to explore defect resolution cycles, during product evolution, before, during and after a software transfer. The overarching research questions that drive this research are:

RQ1: *How does the efficiency of maintenance work vary in GSD projects that involve software transfers?*

RQ2: *What are the confounding factors that influence defect resolution cycles for products that evolve in a GSD context that involve software transfers?*

In our study we use the defect resolution cycle [1] as a surrogate for maintenance work. We use the CFDs to visualize the defect resolution cycle before, during and after a transfer. Unlike defect removal efficiency, which is the percentage of defects removed prior to delivery to customers or users [14], defect resolution efficiency is the speed with which reported defects are solved. Maintenance in our study context refers to system modifications that are carried out to correct and/or change system behavior through, for example, defect-fixes or enhancements [15].

B. Case Description and Context

This study is an exploratory case study that was conducted at a multinational telecommunication corporation, Ericsson. The company was selected due to the company's interest and support in the research. The case study design was conducted following the guidelines proposed by Runeson et al. [16]. We consider the study to be an embedded case study, of two products from the same company, with the unit of analysis being the product. The study subjects are the employees involved in product development, i.e. practitioners.

The two products used in the investigation are software intensive products with customers in numerous countries. For confidentiality reasons the two products are referred to as Alpha and Beta. Both products are large subsystems, software intensive products, that are part of a compound system (i.e. integrated with other products) that is developed for the telecommunication market.

[1] Defect resolution can also be measured by aggregating resolution duration times, as shown by Luijten and Visser [13]. Luijten and Visser use the measure as a proxy or indicator for maintenance efficiency.

Alpha is a complex product that requires specific technical competences. Product development commenced in 2001, and it was initially carried out by a single site in Sweden. In 2007 employees from the company's site in India were brought to Sweden for training purposes in preparation for the transfer. Based on the definition of types of transfers described by Wohlin and Šmite [17] the transfer for Alpha is regarded as a gradual transfer. This gradual transfer from the site in Sweden to the site in India took place in 2009. Alpha was regarded as a mature product.

Beta was involved in two transfers. The first transfer of development and maintenance work responsibilities, occurred between a site in Ireland (owned by another company) and the case company's site in Sweden, in 2007. At this time the product was regarded as an immature and highly complex product. In addition very little product documentation was available.

The second transfer occurred between the first quarter and the fourth quarter of 2009, from the site in Sweden to a site in India. During this period the product was moderately mature. As part of the transfer a consultant was hired to help documenting critical product aspects. This improved product documentation.

For Beta, we only studied the second transfer. This is because the experts available for focus group meeting (that is described in Section III-C2) were those involved after the first transfer. Thus we limit our observation and analysis to after the first transfer and onwards. Using the transfer type definitions described by Wohlin and Šmite [17], the transfer for Beta is regarded as a full transfer.

In 2005, the case company began shifting from using waterfall methodologies to adopting agile practices for development and maintenance activities for all its products, including Alpha and Beta.

Summary of the Context: Providing study context helps in understanding generalizability of study results and findings [18], [19]. The classification of GSD empirical studies proposed in [20], and GSD related characteristics provided in [5], is used in Table I to show how this study relates to empirical GSD studies.

C. Data Collection and Analysis

A methodological triangulation approach is used in the studies by combining quantitative and qualitative methods [16]. This approach helps with increasing validity and precision of study results.

1) Quantitative Data (RQ1): Our unit of **measure** is the number of defects. Defects used in our study are those that are linked to a problem in the source code and thus require a solution to be directly implemented into the source code. Thus we excluded any defects that did not meet this criterion, e.g., defects related to documentation problems.

At the case company, defects are prioritized based on their severity or effect on the system as either high, medium or low. High priority is given to defects that pose a risk to the system stability and may cause failure. Low priority is given to defects that do not have an effect on the performance or stability of the system. During maintenance work, high priority defects are given precedence to medium and low priority defects. High priority defects are also constrained to shorter resolution cycles than medium and low priority defects. Medium priority defects are given precedence to, and also constrained to shorter resolution cycles than, low priority defects.

To ensure a level of comparability of the maintenance flow for defects across releases, CFDs were applied to each of the three types of defects as suggested by Petersen [7].

Measurements: At the case company a date stamp is given to a defect to indicate the state of a defect in the maintenance flow process. More details at a more granular level of the maintenance flow process at the case company can be found in [7]. With the help from a test and verification employee at the company, we extracted and used the date stamps on the defects to map the monthly states of each defect. The following aspects were then measured using the information of defect states over time:

- M1: Rate of defect inflow over time,
- M2: Rate of analyzing, designing and implementing solutions for defects rate,
- M3: Rate of implementing correctional packages solutions for deployment at customer sites.

Visualization: Analysis was done to evaluate periods of efficiency and inefficiency of maintenance work by applying CFDs to the measures to visualize the following aspects:

- Variances of defect inflow (through the aid of M1),
- Bottlenecks in the solution design activities (through the aid of M2), and
- Waiting periods (through the aid of M3).

Variances in defect inflow can either be an increase, decrease, or stable flow. Bottlenecks occur when the rate of designing and implementing solutions is lower than the rate of defect inflow. Waiting periods are indicated by the length in time it takes to release correctional packages to customers. For an example on how to read and interpret CFDs please see Petersen and Wohlin [12].

All authors held an internal discussion during which the authors shared personal observations from the CFDs. A consensus was reached on the period of efficiency and inefficiency, and these periods were marked on the CFDs. The marked CFDs were then presented in focus group meetings at the company for practitioner reflection.

2) Qualitative Data (RQ2): A separate focus group meeting was conducted for each product. A focus group meeting is a method which uses qualitative data collected from group discussions [21]. The meetings were planned and conducted following the steps described by Kontio et al. [22].

The focus group meetings were used as a platform to present the CFDs, and to gather insights not visible from

Table I: Study Context

Empirical Background		
Main Method	Embedded case study	
Sub Methods	Quantitative analysis and focus group meeting	
Empirical Focus	Empirically-based (exploratory)	
Subjects	Practitioners	
Focus of Study	Defect resolution cycle during software evolution	
GSD Background		
Collaboration Mode	Intra-organization/ Offshore insourcing	Intra-organization/ Offshore insourcing
Approach and Type of Work	Single-site execution before the transfer. In preparation for the transfer work was distributed. The transfer was a gradual transfer during which work was distributed. The transfer resulted in a single-site execution.	Single-site responsibility before the transfer. The receiving site was involved with the product before the transfer which resulted in distributed work. The transfer resulted in a single-site execution.

Table II: Focus Group Participants

** The employee is from the site located in Sweden. The others without the asterisk are from the site in India*

Product	Role	Responsibility	Period of Involvement
Alpha	Senior Test and Verification Manager*	Managing testing activities	Before and during transfer
Alpha	Senior System Tester*	Tester	Before and during transfer
Alpha	Test Coordinator	Coordinating testing activities	Before and during transfer
Alpha	Technical Support	Monitoring defect flow and providing support to customers	During and after transfer
Alpha	System Tester	Functional and non-functional testing	During and after transfer
Alpha	Architect	Designing and analyzing both architecture and source code artifacts	During and after transfer
Beta	Senior Test and Verification*	Managing testing activities	Before and during transfer
Beta	Senior System Tester*	Managing testing activities	Before and during transfer
Beta	Senior Manager	Managing of architecture and development activities	During and after transfer
Beta	Senior Manager	Managing of architecture and development activities	During and after transfer
Beta	Architect	Designing and analyzing architecture	During and after transfer
Beta	Senior Software Engineer	Software Developer	During and after transfer

the quantitative data pertaining to events or factors that may have had a confounding effect on defect resolution cycles. Each meeting lasted for approximately an hour.

The focus group participants were either involved in the maintenance process, or had knowledge of the products' evolution and history. Product and project managers for both products helped with selecting suitable participants.

Each focus group consisted of employees in Sweden and employees in India. The focus group meetings were held at the site in Sweden. Hence, employees from the site in India took part in the discussions through video conference calls. This created a platform that consisted of both face-to-face and virtual focus group meetings. For Alpha, participants in the focus group meeting were those involved with the product before, during and after the transfer. For Beta, the participants were those involved with the product after the first transfer and, during and after the second transfer. The participants are listed in Table II.

Participants in the meetings for each product shared similar opinions on their reflections. The reflections were documented in real-time in the form of written notes. Analysis of the notes was performed in four steps. In the first step we identified reflections from the meetings on the measures that were visualized in the CFDs. In the second step we separated the reflections based on their relation to the transfer period, i.e., before, during and after a transfer. In the third step, we examined the reflections to identify factors that the participants linked to patterns and trends in defect inflow, bottlenecks and waiting periods. In the fourth step, we characterized the factors by the appropriate type of transfer critical factor, i.e., product, people or process, that are described by Šmite and Wohlin [5].

The final result was a table that captures the type of critical factor, and a description of the factors (as described by the focus group participants), and the effect the factors had on defect resolution cycles, before, during and after a transfer as observed from the CFDs. The table was an "m-by-3" matrix; where m is the number of factors and three are the transfer periods, i.e., before, during or after a transfer. The analysis approach we followed, i.e., of grouping and categorizing the reflections, is similar to the thematic analysis approach [23].

IV. Results and Analysis

A. Product Evolution

Defect data extracted from the company's defect-tracking system for Alpha spanned nine years, between 2004 and 2012. During that time period a total of nine releases were made available to the market. Defect data for Beta spanned five years, between 2008 and 2012, and seven releases were made available to the market. This data highlights the evolution of the products, (i.e., software releases over time), of the two software products. Such a wide range of data helps with visualizing variances in defect resolution cycles over time, and also helps to understand the extent of the impact of transfers on efficiency during defect resolution activities, during evolution.

Observations of the CFDs and the reflections from the focus group meetings for the aforementioned data are presented in the next sections.

B. Alpha

Observations: Figure 1a, 1b and 1c shows the CFDs for the high, medium and low priority defects, respectively, for Alpha.

After a period during which there is a continuous stable low rate of defect inflow, the rate of defect inflow increases. Notably, this increase begins from the point that employees from Alpha's Site-B get involved with the product and after Alpha's R1 release. The continuous stable low rate of defect inflow is visible in all priority defects, i.e., indicated by 1A in Figure 1a for the high priority defects, indicated by 1B in Figure 1b for the medium priority defects, and indicated by 1C in Figure 1c for the low priority defects. Increase in defect inflow before the transfer is more visible in the high priority defects indicated by 2A in Figure 1a, and in the low priority defects indicated by 2C in Figure 1c, than in the medium priority defects.

During the transfer the rate of defect inflow increases for all defects. This is indicated by 3A in Figure 1a for the high priority defects, and indicated by 2B in Figure 1b for the medium priority defects, and 3C in Figure 1c for the low priority defects. Interestingly bottlenecks in the solution design activities emerge during the transfer for the high priority defects indicated by 4A in Figure 1a, and for the low priority defects indicated by 4C in Figure 1c. It is important to note that, though bottlenecks are visible for the medium priority defects during the transfer, there are bottlenecks visible in the CFD for the medium priority defects across all releases, before, during and after the transfer. In the case for the high and the low priority defects, the bottlenecks appear after Alpha's release R5 is made available to the market.

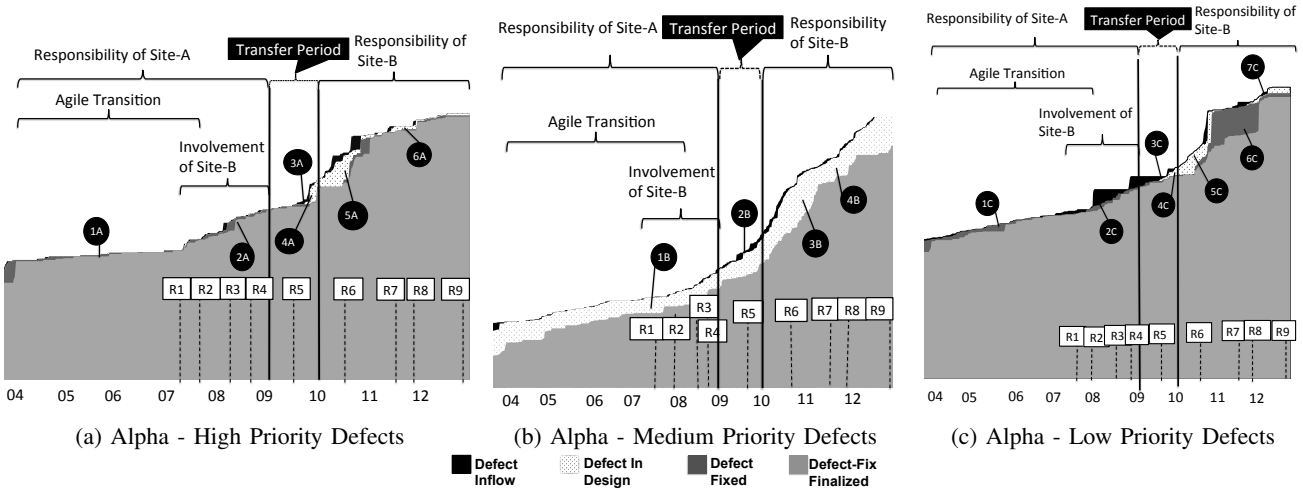
There are sharp increases in defect inflow soon after the transfer period, and bottlenecks in the solution design activities that are more prominent than during the transfer, for the high priority defects indicated by 5A in Figure 1a and the low priority defects indicated by 5C in Figure 1c. This period is also around the time a new product release, R6, is made available to the market, which is six months after the end of the transfer period. After this release, there are visible delays in the finalization of correctional packages for defects of low priority, indicated by 6C in Figure 1c.

Approximately, a year after the transfer, defect inflow begin to stabilize, and there are improvements in the solution design activities as bottlenecks are less apparent, and the rate of finalizing correctional packages get faster. This period is indicated by 6A in Figure 1a for the high priority defects, and indicated by 7C in Figure 1c for the low priority defects.

Overall, bottlenecks that emerge for high and low priority defects during and after the transfer, as well as the delay in finalizing correctional packages indicate periods that experienced a decrease in efficiency in resolving defects. It is important to note that medium priority defects show a different trend in defect resolution cycles across releases in comparison to the high and the low priority defects. This difference is further elaborated in the reflection section. The medium priority defects have a prolonged and prevalent bottleneck in the solution design activities across all releases, before, during and after the transfer. In addition the decrease in efficiency during the transfer is much more apparent in the high and low priority defects, than the medium priority defects. This difference highlights the usefulness of taking into consideration the severity of the defects when analyzing maintenance activities, because the flow of the maintenance process can be influenced by the type of defects being reported, i.e., the effect of the defects to the system's stability.

Reflections: Participants in the focus group meeting for Alpha linked the increase in defect inflow just before the transfer to Alpha's new market release, R1, and the introduction of new employees from Alpha's Site-B that were brought in for training in preparation for the transfer. The new employees were still learning the ripple effects of making changes in the complex product. Ripple effects are concerned with the impact that certain changes may have on other areas of the system [24]. According to the focus group meeting participants, there was also insufficient documentation to help the new employees learn the complex product.

For increases in defect inflow that occur during the transfer period, the participants linked them to the increase in new customers. As the number of customers increased, and the number of end users increased, previously unknown issues were discovered. They linked the bottlenecks to the new release for Alpha, R5, that came with new complex functionality. This new release made maintenance activities



(a) Alpha - High Priority Defects (b) Alpha - Medium Priority Defects (c) Alpha - Low Priority Defects

Figure 1: CFD for Alpha

more challenging for employees from Alpha's Site-B who were still learning the product. The participants also pointed out that the complexity of the product architecture and insufficient product documentation made the handover or transfer difficult to execute.

For the increase in defect inflow and bottlenecks in the solution design activities, after the transfer, the participants linked them to an increase in new requirements and functionality implemented in releases R6 and R7, shown in Figure 1c. This occurred at a time the employees were still climbing the learning curve. On top of that, due to attrition at Site-B, there were changes in the design and maintenance team after the transfer, which resulted in new employees being introduced in the team. This considerably decreased the competence within the maintenance teams and amplified the negative effect on defect resolution cycles, and hence efficiency. The consequence was bottlenecks in the solution design activities and delays in finalizing correctional packages as the new employees needed time to climb up the learning curve of the complex product. As a result of the bottlenecks there were increases in escalations from customers pertaining to product related issues.

With regards to the prolonged or prevalent bottlenecks in the solution and design activities visible in the medium priority defects, across releases, the participants indicated that there was less pressure to formulate fixes for defects of medium priority in comparison to the high priority defects. The higher priority defects were given precedence and constrained to shorter resolution cycles.

Overall, the confounding factors that had an impact on defect resolution cycles for Alpha are: (1) unfamiliarity or inexperience of employees with the product, (2) learning curve of the complex product for the new employees and insufficient product documentation, (3) implementation of complex product functionality, and (4) unstable maintenance team after the transfer, and (5) low precedence/priority for certain maintenance work.

C. Beta

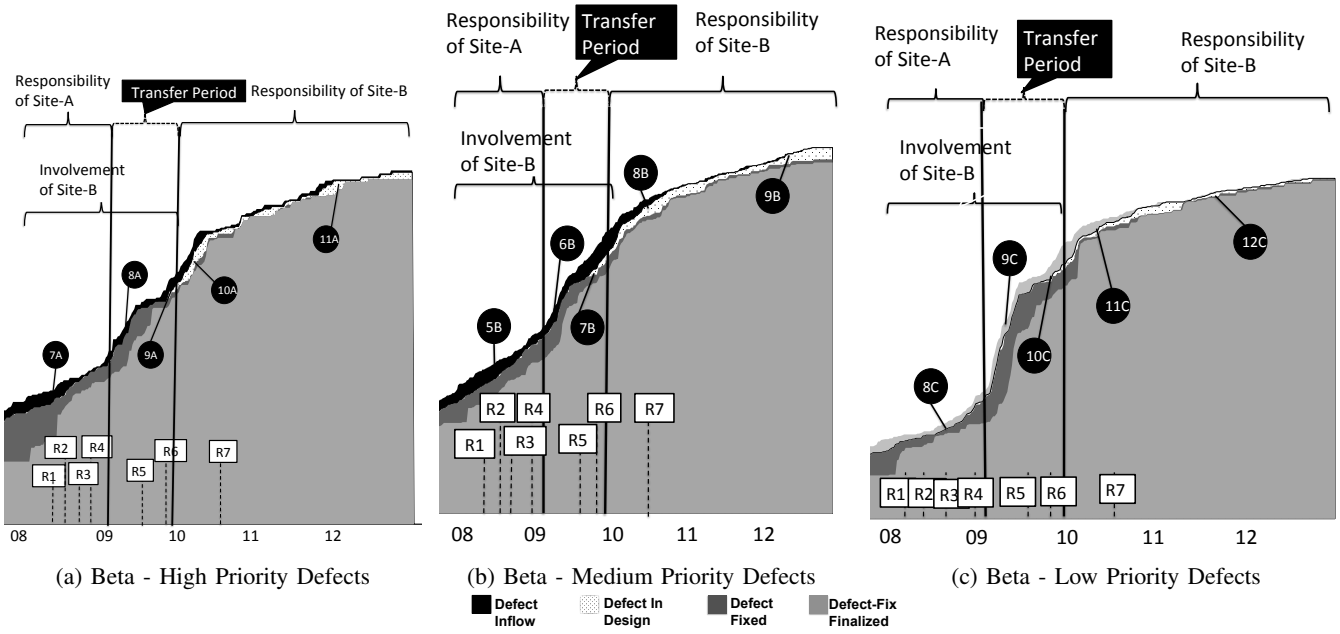
Observations: Figure 2a, Figure 2b and Figure 2c shows the CFDs for the high, medium and low priority defects, respectively, for Beta.

Before the transfer for Beta there are increases in the rate of defect inflow and delays in the finalization of correctional packages in the high priority defects indicated by 7A in Figure 2a, and the medium priority defects indicated by 5B in Figure 2b, and the low priority defects indicated by 8C in Figure 2c.

After Beta's R4 release, the rate of defect inflow increases and it is faster than before the release and there are still delays in finalizing correctional packages. This period is during the transfer and it is indicated by 8A in Figure 2a for the high priority defects, and by 6B in Figure 2b for the medium priority defects, and by 9C in Figure 2c for the low priority defects.

Apart from the increase in defect inflow and delays in finalizing correctional packages during the transfer period, bottlenecks in the solution design activities also emerge, in all three types of Beta's defects. The bottlenecks emerge after Beta's R6 release. The bottlenecks are indicated by 9A in Figure 2a for the high priority defects, and by 7B in Figure 2c for the medium priority defects, and by 10C in Figure 2c for the low priority defects.

After the transfer period, similarly to during the transfer, there is an increase in defect inflow, bottlenecks in the solution design activities, and there are delays in finalizing correctional packages. These aspects are visible in all three types of the defects. The increase in defect inflow and the bottlenecks are more apparent in the high priority defects indicated by 10A in Figure 2a, and for the medium priority defects indicated by 8B in Figure 2b, than those visible in the low priority defects indicated by 11C in Figure 2c.



(a) Beta - High Priority Defects (b) Beta - Medium Priority Defects (c) Beta - Low Priority Defects

Figure 2: CFD for Beta

After the last release for Beta, R7, approximately a year after the transfer period, though the rate of defect inflow gets slower and the rate of finalizing correctional packages is faster, bottlenecks in the solution design activities remain apparent after the transfer period. These bottlenecks are indicated by 11A in Figure 2a for the high priority defects, and by 9B in Figure 2b for the medium priority defects, and by 12C in Figure 2c for the low priority defects.

Overall, the delays in finalizing correctional packages before, during and the transfer, and the bottlenecks during and after the transfer highlight areas for improving efficiency during maintenance work for Beta. Unlike for Alpha, the three defect types for Beta show similar trend in defect resolution cycles.

Reflections: For Beta, focus group meeting participants linked the increase in defect inflow to increases in new requirements. Participants linked the delays in finalizing correctional packages to that there was a low competency within the maintenance team. There were new employees from Beta's Site-B that had little or no previous engagement with the product that had joined maintenance activities. In addition, Beta's R4 release was a major release that made maintenance work even more challenging for the inexperienced employees. The participants pointed out that the increase in defect, and the delays in finalizing correctional packages, before the transfer can also be linked to the major re-architecturing activities that was taking place at the time the new employees joined. The major re-architecturing activities that was taking place made the learning and understanding of the complex product difficult for the employees from Beta's Site-B.

During the transfer, there was an increase in the number of customers due to the major release, R4, just before the transfer period. Participants from the focus group meeting for Beta linked this aspect to the increase in defect inflow. Additionally, as the transfer was taking place, there were more new employees from Beta's Site-B that were involved in the handover. These new employees needed time to build-up competencies. However, during this period new complex functionality was added with releases R5 and R6 for Beta, and as a result slowed down the learning process of the maintenance teams. In addition to making maintenance work more challenging for the new employees, the new requirements slowed down the learning process. The participants linked these aspects to the bottlenecks and the delays in finalizing correctional packages that occur during the transfer.

Participants of the Beta focus group meeting linked the bottlenecks that occur after the transfer to that employees at Site-B employees were still climbing the learning and building up competencies for the complex product. They also pointed out that, a year after the transfer, the product was in a servicing stage, and it was transitioning into the phase-out stage. As a result, maintenance activities for the product were not prioritized. The servicing stage followed by the phase-out stage, are the last two stages of a products evolution leading to the close down of the product [1]. This period is indicated by 11A in Figure 2a, by 9B in Figure 2b, and by 11C in Figure 2c. During this period there were no new functionality added to the product and minimum effort was assigned to maintenance activities.

D. Cross case comparison

Defect data for Beta has some similarities and differences in comparison to that of Alpha. There are no periods of a continuous stable low rate of defect inflow before the transfer for Beta that is visible in the data for Alpha. There are delays in the finalization of correctional packages before, during and after transfer for Beta, but not for Alpha. However, for both products, during and after the transfer the rate of defect inflow increases and bottlenecks in the solution design activities emerge.

Another interesting aspect is that the rate of defect inflow is higher during the transfer period than before or after the transfer period in both cases. In addition, bottlenecks are much more visible after the transfer period.

Overall, apart from a few differences in the quantitative data, generally the factors that had an impact on defect resolution cycles for Beta are similar to those that had an impact for Alpha. One factor that was identified in the case for Alpha but not for Beta is low precedence that was assigned to certain maintenance work. Two factors that had an impact for Beta but were not brought-up for Alpha are: major re-architecturing activities and software evolution life-cycle stage of the product.

V. Summary and Discussion

Our findings show that there is a negative effect on defect resolution cycles during the transfer period and months after the transfer period as indicated by the bottlenecks and delays in releasing/finalizing correctional packages. These are indicators of a decrease in efficiency. Table III shows factors that were obtained through reflections during the focus group meetings, for Alpha and/or Beta, that had an impact on the measures M1, M2 and M3 (for description of the measures, see Section III-C1). Data in the table is sorted by the categories for transfer critical factors that are described by Šmite and Wohlin [5].

Our findings confirm and illustrate how some of the factors identified in [5] have an effect on defect resolution cycles, and hence maintenance work efficiency. This supports posits reported by Carmel and Tjia [4] and by Šmite and Wohlin [5], and findings from Mockus and Weiss [9]. Both products used in our study experienced periods of bottlenecks, increased inflow of defects and lengthening of waiting periods. Table III, that summarizes our findings, also suggests that there are more product related factors that have an impact on defect resolution cycles, than people- and process related factors. This highlights the importance of taking note of product factors when planning transfers.

The factors that were pointed out, during the focus group meetings, that had an impact on defect resolution cycles before, during and after a transfer, from at least one of the cases were: learning curves, competency within teams, and product complexity. Notably, flat learning curves due to product complexity, i.e., architecture complexity, were pointed out during the focus group meeting, for both of the two products in our study. Furthermore, the size of maintenance work, as complex functionality is added across releases, can affect efficiency during evolution [15]. This highlights the importance of knowledge transfer activities for the execution of a successful transfer.

Participants of the focus group meeting for Beta pointed out that to shorten the learning curve during the transfer employees at Beta's Site-B analyzed the complex source code and held meetings to pass on knowledge between team members. They indicated that this practice was efficient in improving competency with in teams at Beta's Site-B. This highlights the importance of source code readability as it has influence on the efficiency of maintenance work. In addition, given that over 50% of maintenance time is spent on reading source code [25], research on source code readability would greatly benefit companies engaging in transfers.

Both transfer and non-transfer related events, such as transitioning to agile practices, were highlighted in the cumulative flow diagrams during the focus group meetings. However, we did not obtain any indications from the discussions that such non-transfer-related events had an impact on defect resolution cycles. Other confounding factors that can be found in GSD literature, such as culture [4], were not brought-up as factors of relevance. However, this does not infer that the factors did not have any effect.

Mockus and Weiss [9] observed that learning curves can negatively impact productivity at the receiving site. Our findings are consistent with their findings. We found that learning curves had an impact on defect resolution cycles of high priority, i.e., high severity defects, for both products. Learning curves resulted in the increase in defect inflow and bottlenecks in solution design activities for Alpha. For Beta they had an influence on defect inflow, emergence or strengthening of bottlenecks in solution design activities and slowing down of the implementation and finalization of correctional packages, after the transfer. In addition, from our study findings it is discernible that the implementation of new requirements and complex functionality whilst employees were still climbing the learning curve, for both Alpha and Beta, amplified the effect on defect resolution cycles.

Šmite and Wohlin [5] identified transfer critical factors, i.e., product-, people- and process-related factors that can impact efficiency before, during and after transfer. Using a lean indicator to visualize the magnitude of the effect of transfers on defect resolution cycles, our study corroborates with their study. Some of the factors that can be linked to transfer critical factors that we found to have an effect on defect resolution cycles are: product complexity, product documentation, and experience with the product at the receiving site.

Table III: Confounding Factors

Categories	Critical Factors	Impact	Impact Period	Transfer Related	Product
People	Lack of necessary knowledge/low competency within team	Increase in defect inflow; Bottlenecks in solution designing activities; Delays in finalizing correctional packages $*$	Before, during and after the transfer	✓	Alpha and Beta
People	Unstable maintenance team	Bottlenecks in solution designing activities; Delays in finalizing correctional packages	After the transfer	✓	Alpha
Process	Low precedence/priority for the maintenance work	Bottlenecks in solution designing activities $**$	Before, during and after the transfer		Alpha
Product	Learning curve of complex product and insufficient product documentation	Same as $*$	Before, during and after the transfer	✓	Alpha and Beta
Product	New product release with complex functionality	Same as $*$	Before, during and after the transfer		Alpha and Beta
Product	Major re-architecturing activities	Same as $*$	Before the transfer		Beta
Product	Servicing stage of the software evolution lifecycle	Same as $**$	After the transfer		Beta

VI. Validity Threats

To reduce biased interpretation of the cumulative flow diagrams, all authors were involved in the analysis of the diagrams before the focus group meetings. The authors also held a meeting during which a consensus was reached on the interpretation of the cumulative flow diagrams.

Alignment of defects to the appropriate releases can be a challenging endeavor. Errors in the process can negatively affect study results. A test and verification manager at the case company helped with the defect extraction, alignment of the defects to releases, as well as identifying defects that were related to a problem in the source code.

Periods of varying increases in factors, such as customers and product complexity, were pointed out by the focus group participants. However, data to confirm the subjective opinions was not available at the time of this study. An analysis of the data would have helped to understand objectively the extent of the effect of the factors on defect resolution cycles, before, during and after a transfer. Due to that we were unable to obtain exact sizes for each release for each product, we were unable to normalize the data. This would have helped with understanding the extent of effect that varying sizes of releases had on defect inflow and resolution cycles.

The products used in this study should not be perceived as a complete representative of all products that are transferred in GSD projects. Both products used in this study were developed at the same company, thus results are specific to the case company. Detailed description of the study context (see Section III, and summary Table I), and the similarities and the differences between the two products used (see Section IV-D) improves external validity of our study [19]. The confounding factors that impact defect resolution cycles (see Section V) are likely to have an impact when transferring maintenance work for large and complex systems similar those that are used in our study. Besides, study findings are intended to provide valuable empirical input to support decision-making when planning transfers in GSD settings, by identifying confounding factors that can impact defect resolution cycles during such endeavors.

VII. Conclusion

The objective of this study is to investigate how defect resolution cycles vary across releases in a global software development (GSD) context that involves a software transfer. The investigation is done using two large software intensive products from a large multinational corporation. The defect-resolution cycle is used as a surrogate for maintenance efficiency. An evolutionary view is used to visualize defect resolution cycles across releases for the products. This visualization is done using a lean indicator, cumulative flow diagram, which is based on lean principles. In each cumulative flow diagram we highlight major events that are both transfer related and non-transfer related. In addition, focus group meetings were conducted to gather opinions on

defect resolution cycles and to identify confounding effects on efficiency.

Our study suggests that companies that engage in software transfers should give attention to the impact that product-, people- and process-related factors can have on efficiency, and consider risk mitigation strategies.

For future work statistical reasoning and/or machine learning techniques could be considered in the investigation to further to further increase the understanding the effects on quality of software transfers. In addition, further investigation could be done by aggregating factors found in this study and those reported in literature (e.g. trust) and distributing a questionnaire to practitioners in which, based on their subjective opinion, they can rank the factors according to their effect on efficiency. This would help to further understand the relevance of the factors, and how the effect of the factors is perceived in industry in comparison to other factors.

ACKNOWLEDGMENT

This work is partially funded by the Swedish Knowledge Foundation in Sweden under the grant 20120200 (2013-2016) and Ericsson Software Research.

REFERENCES

[1] K. H. Bennett and V. T. Rajlich, "Software maintenance and evolution: A roadmap," in *Proceedings of the conference on future of software engineering*, June 2000, pp. 73–87.

[2] C. Jones and O. Bonsignour, *The economics of software quality*. Boston, MA: Addison Wesley Publishing, 2011.

[3] D. Bijlsma, M. A. Ferreira, B. Luijten, and J. Visser, "Faster issue resolution with higher technical quality of software," *Software quality control*, vol. 20, no. 2, pp. 265–285, June 2012.

[4] E. Carmel and P. Tjia, *Offshoring information technology: Sourcing and outsourcing to a global workforce*. Cambridge, UK: Cambridge University Press, 2005.

[5] D. Šmite and C. Wohlin, "Strategies facilitating software product transfers," *IEEE software*, vol. 28, no. 5, pp. 60–66, October 2011.

[6] J. D. Herbsleb, A. Mockus, T. A. Finholt, and R. E. Grinter, "An empirical study of global software development: Distance and speed," in *Proceedings of the 23rd international conference on software engineering*, 2001, pp. 81–90.

[7] K. Petersen, "A palette of lean indicators to detect waste in software maintenance: A case study," in *Agile processes in software engineering and extreme programming*, ser. Lecture notes in business information processing, 2012, vol. 111, pp. 108–122.

[8] J. Verner, O. Brereton, B. Kitchenham, M. Turner, and M. Niazi, "Systematic literature reviews in global software development: A tertiary study," in *Proceedings of the 16th international conference on evaluation assessment in software engineering*, 2012, pp. 2–11.

[9] A. Mockus and D. M. Weiss, "Globalization by chunking: A quantitative approach," *IEEE software*, vol. 18, pp. 30–37, March 2001.

[10] D. Šmite and C. Wohlin, "Lessons learned from transferring software products to india," *Journal of software: Evolution and process*, vol. 24, no. 6, pp. 605–623, October 2012.

[11] R. Jabangwe and D. Šmite, "An exploratory study of software evolution and quality: Before, during and after a transfer," in *Proceedings of the 7th IEEE international conference on global software engineering*, August 2012, pp. 41–50.

[12] K. Petersen and C. Wohlin, "Measuring the flow in lean software development," *Software: Practice and experience*, vol. 41, no. 9, pp. 975–996, August 2011.

[13] B. Luijten and J. Visser, "Faster defect resolution with higher technical quality software," in *Proceeding of the 4th international workshop on system quality and maintainability*, 2010, pp. 11–20.

[14] C. Jones, "Measuring defect potentials and defect removal efficiency," *Crosstalk: The Journal of defense software engineering*, vol. 21, no. 6, pp. 11–13, June 2008.

[15] B. A. Kitchenham, G. H. Travassos, A. von Mayrhauser, F. Niessink, N. F. Schneidewind, J. Singer, S. Takada, R. Vehvilainen, and H. Yang, "Towards an ontology of software maintenance," *Journal of software maintenance and evolution: Research and practice*, vol. 11, no. 6, pp. 365–389, December 1999.

[16] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case study research in software engineering*. New Jersey, USA: John Wiley & Sons, 2012.

[17] C. Wohlin and D. Šmite, "Classification of software transfers," in *Proceedings of the 19th Asia-Pacific software engineering conference*, vol. 1, 2012, pp. 828–837.

[18] B. Kitchenham, S. Pfleeger, L. Pickard, P. Jones, D. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Transactions on software engineering*, vol. 28, no. 8, pp. 721–734, August 2002.

[19] K. Petersen and C. Wohlin, "Context in industrial software engineering research," in *Proceedings of the 3rd international symposium on empirical software engineering and measurement*, 2009, pp. 401–404.

[20] D. Šmite, C. Wohlin, R. Feldt, and T. Gorschek, "Reporting empirical research in global software engineering: A classification scheme," in *Proceedings of the 3rd international conference on global software engineering*, August 2008, pp. 173–181.

[21] L. D. Morgan, "Focus groups," in *The SAGE encyclopedia of qualitative research methods, vol 1 and 2*. Sage Publications, 2008, pp. 352–354.

[22] J. Kontio, L. Lehtola, and J. Bragge, "Using the focus group method in software engineering: Obtaining practitioner and user experiences," in *Proceedings of the international symposium on empirical software engineering*, August 2004, pp. 271–280.

[23] L. Ayres, "Thematic coding and analysis," in *The SAGE encyclopedia of qualitative research methods, vol 1 and 2*. Sage Publications, 2008, pp. 867–868.

[24] S. Yau, J. Collofello, and T. MacGregor, "Ripple effect analysis of software maintenance," in *Proceedings of the 2nd IEEE international computer software and applications conference*, 1978, pp. 60–65.

[25] J. Foster, "Cost factors in software maintenance," Ph.D. dissertation, University of Durham, School of Engineering and Computer Science, Durham, UK, 1993.