Classification of Software Transfers

Claes Wohlin
Blekinge Institute of Technology
Karlskrona, Sweden
Claes.Wohlin@bth.se

Darja Šmite
Blekinge Institute of Technology
Karlskrona, Sweden
Darja.Smite@bth.se

***Abstract* — Many companies have development sites around the globe. This inevitably means that development work may be transferred between the sites. This paper defines a classification of software transfer types; it divides transfers into three main types: full, partial and gradual transfers to describe the context of a transfer. The differences between transfer types, and hence the need for a classification, are illustrated with staffing curves for two different transfer types. The staffing curves are obtained through a combination of interviews with both high-level management and a group of experts, and an industrial case study. From the empirical work, it is concluded that the distribution of personnel differs for different types of transfer, which means that it is crucial to be clear about different classes of software transfers. If not, it is easy to underestimate the effort needed to transfer software work as well as additional costs related to the transfer as such.**

Keywords: Global software engineering, global software development, software transfers, distributed teams, offshoring, offshore insourcing, empirical study

I. INTRODUCTION

The world is evolving and global software development has become business as usual. Companies strive to make the best possible use of the global market by, for example, having sites in different countries. The latter is, among the other reasons, motivated by differences in costs, proximity to different markets and tapping into the global pool of talent and expertise around the globe. This is not something completely new, but it has accelerated in the last decade.

The perception of a manager at Ericsson, involved in software transfers, triggered a literature search related to change in development efficiency when software is transferred from one site to another. Ericsson provides products within the telecom domain, and preferably the manager wanted evidence of efficiency changes after a transfer from this domain. At the point of the literature review, three studies were identified as shown in Figure 1. The decrease in development efficiency was inline with the gut feeling of the manager, but still the numbers are hard to use due to them coming from different applications domains. The example illustrates both a potential consequence of software transfers, but also how important the context is to make the findings trustworthy and useful both in practice and for other researchers. Findings are context-dependent.

A related problem is that when transfers are reported in literature, it is hard to understand the context of the transfer as such. Thus, there is a need to have a classification of software transfers so that others who have a similar context can more easily use the findings such as those in Figure 1. Thus, even if the application domain would have been telecommunications, it is in most cases impossible to deduce how the transfer was actually conducted. More information about the cases in Figure 1 can be found in [1] for the Meta Group and the other two cases are described in [2].

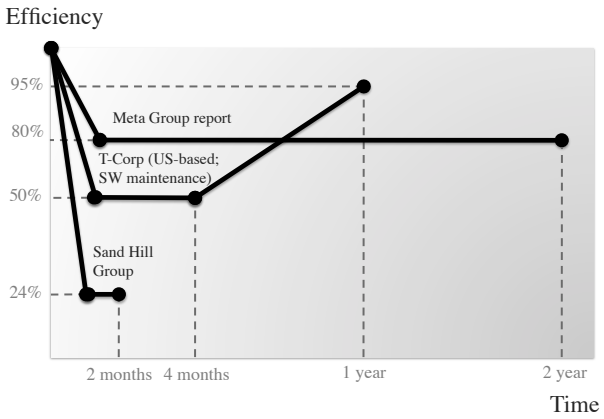

Figure 1. Illustration of efficiency decrease in software transfers.

Before classifying software transfers, it is important to define the concept of a transfer project. Here, the following definition is used:

Definition: Transfer project – A project where work is moved from one development site to another development site. Each transfer project is assumed to be planned individually, and being characterized by a start state and an end state.

The globalization and its consequences may be illustrated with an example from Ericsson. In Figure 2, it can be seen that the percentage of employees in Sweden for Ericsson was in year 2000 around 40% of 105 000 employees and in 2010 the percentage is below 20% of approximately 92 000 employees. Data for individual years may be extracted from reports available in [3]. This shows the transformation of industry, which is due to many different factors. For example, it is due to acquisitions, but also a change in terms of what the company actually produces. Today, it is estimated that 80% of the R&D at Ericsson is related to software [4], which most certainly is higher than in 2000 (actual numbers are unavailable). Thus, this illustrates how one company has both become much more global and at the same time moved into being a major software development company. This inevitably leads to that software

development is transferred between sites in some cases to target the market, in others to free up resources or develop a certain competence in a certain location.

Interestingly, relocation of software work is often associated with a turnover of the employees, since it resembles replacement of developers in one site with those in another. However, this does not have to be true; it could very well be motivated to free up resources for development of new products. To illustrate the importance of the context and hence that different types of transfers may be very different, examples of staffing patterns are used as an illustration. Staffing patterns from two different types of transfers are presented to further emphasize the need for more contextual information when it comes to software transfers. The latter is the key motivation for proposing a classification of transfers. All transfers are not of the same type, and researchers and practitioners alike must understand the differences to be able to evaluate the value of the evidence provided in specific cases.

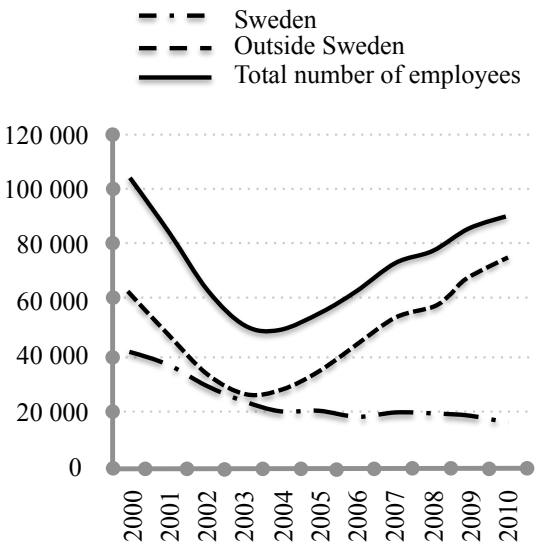

Figure 2. Number of employees in Ericsson 2000-2010 [3]

This paper contributes with a classification of different types of software transfers, and illustrates the importance of a classification by showing some different staffing scenarios from different types of transfers. The classification of transfer types is based on three years of collaborative work with Ericsson and one other company in Sweden as part of a joint research project. Thus, the classification should be viewed as a complement specific for software transfers in relation to the "normal" contextual information needed for industrial studies as described in [5]. The need for reporting guidelines in global software development has been stressed in, for example, [6]. Thus, the classification here provides an added dimension to cover in the contextual information when it comes to software transfers.

The remainder of the paper is organized as follows. In Section II, related work is presented. Section III describes the empirical background and research methodology. The classification scheme for software transfers is introduced in Section IV. Illustrations of staffing curves for different transfer types are presented in Section V, followed by a discussion in Section VI. Finally, Section VII concludes the paper with a summary of the major findings.

II. Related Work

The challenges and peculiarities of developing software across national, organizational, temporal and cultural boundaries [7] are well known by now, but still the actual empirical evidence related to different practices to be successful in global software engineering is scarce [7, 8]. One of the important conclusions becoming apparent from empirical observations suggests that the realization of assumed benefits enabled by globalization, such as cheaper and faster development to name a few, cannot and shall not be taken for granted [9]. The reasons behind unmet expectations are commonly associated with the increased complexity of managing cross-border collaboration [10]. Distributed development has proven to be less efficient than software projects managed entirely at one site [11]. Thus, there is a growing interest in software transfers – relocation of existing work from one site of a company to another site or even a third party vendor. Transfers of software work are also motivated by the willingness to improve the leverage of resources, since companies suffer from tight budgets and shortage in skilled people available at a competitive cost, as well as to gain proximity to a growing market.

Unfortunately, these decisions are too often than not looked at in simple economic terms – it is cheaper, and skilled labour is easier to find [2]. Nonetheless, empirical studies have demonstrated that recruitment can be challenging [12] and high levels of attrition can introduce significant difficulties on the way of achieving these expectations [9]. In addition, the process of transferring software work is associated with significant challenges in relation to knowledge transfer. In fact, a transfer can be compared with a large loss of team members, and changes to the team can be critical to performance [13], knowledge retention and thus the quality [14]. As a consequence, maintaining the same level of service after any transfer independent of locations of the sites becomes challenging. The challenges outlined mean that all lessons-learned and experiences must be highly valued, but to get the full benefits of them, the context in relation to the lessons-learned and experiences must be clear.

Empirical investigations suggest that software transfers cannot be performed overnight. One of the reasons for this is the necessity of knowledge about the software product and product domain expertise for handling software development successfully. A knowledge transfer from one site to another, however, takes considerable amount of effort and time. Previous studies have shown that the learning curve of the new site takes time due to substantial training required for those who are unfamiliar with the product [15]. While the new site is climbing up the learning curve, their efficiency tends to be lower than that of the original unit. The drop in efficiency for the organizations transferring the work can decrease down to 20% as shown in Figure 1, and may not fully recover [2]. The time for recovery varies from one year for less complex

tasks such as maintenance [2] to more than five years for knowledge-intensive work such as development [16, 17].

In our earlier qualitative observations from a case study of an in-house software product transfer from Sweden to India at Ericsson we have also observed a drop in productivity in the short-term [12], which is in accordance with other studies [16, 17] of transfers. Furthermore, slower release cycle and an increase in software defects have been observed in relation to software transfers [18]. The findings led to conclude that the main challenges in software transfers are related to finding the right people, transferring competence, maintaining on-going development, and overcoming cultural differences [12]. We identified several unforeseen risks that came into play during the transfer and required additional investments. Timely investments were likely to ease the transfer. At the same time cost reduction strategies often demand to perform transfers with minimal investments. Thus, we believe that it is highly important to explore the ways to approach transfers wisely, in particular because overcoming the identified challenges may take longer than it takes to meet the objectives on the real capability front [19].

Since literature on software transfers as such is scarce [12], the importance of capturing empirical experiences grows. Supplementing our previous findings based on qualitative interviews; we propose a classification of software transfers. The objective is to present a classification that may help when addressing different types of transfers both in research and in practice. Different types of transfers need to be addressed in different ways, and experiences from one type of transfer may not easily be used in a different context and hence the classification is introduced to help identifying similarities between transfers.

After having introduced the classification of software transfers, some examples of staffing curves are presented to illustrate some of the differences between different types of transfers. Unfortunately, it is impossible to illustrate all types of transfers, and the empirical data needed for further examples is not available from the company at the time of the study. The illustrations also show that there are indeed differences and hence it works as a validation of the need for classifications of software transfers. The staffing curves are used since they easily visualize the effect of a transfer. Traditionally staffing curves have been studied in the light of effort distribution and staffing patterns in different phases of software projects (e.g. [20]). Globalization of software organizations has motivated further investigation of the impact of organizational structure, experience and expertise on software development and product quality [13, 14].

III. Research Methodology

A. Research questions

Motivated by our earlier observations from transferring software products at Ericsson, in this paper we aim at addressing the following research questions:

RQ1: Is it possible to define a comprehensive classification with orthogonal software transfer types?

RQ2: How can staffing curves for different transfer types be used to illustrate the difference between different transfer types?

B. Empirical background

The empirical work contains two main parts:

1. The classification is based on the experiences from conducting research with Ericsson in the area of global software engineering during three years, which have given access to several different transfer projects and people having been involved in transfers.
2. The staffing curves used for illustration of the differences between two different types of software transfers are primarily based on project data and interviews.

1) Empirical basis: classification

The classification is based on the experiences from working with Ericsson in the area of global software engineering during three years. Thus, the classification reported in this paper is a result of continuous research collaboration on software product transfers between Ericsson, one of the leading companies in telecommunications worldwide, and Blekinge Institute of Technology in Sweden. The company develops a wide range of products and solutions, including generic software products offered to an open market and complex compound systems with customised versions. During the past several years Ericsson has extended its operation in Asia and transferred a number of software products between different sites. Software products have also been received by the Swedish site studied from other Ericsson sites or acquired companies. Acknowledging the challenges of software product transfers, Ericsson initiated a research project that aims at collecting and documenting experiences for organizational learning.

Published results to date comprise of a case study [21], a multi-case study [22], both of which offer observations from software transfers from Sweden to India and describe the process of transferring software work from one site to another and challenges associated with the transfers. Further, we have developed a strategy for conducting software transfers that offers a list of factors alleviating transition [22], and quantitatively evaluated the effect of a transfer on software release frequency and quality [18].

2) Empirical basis: staffing curves

Earlier findings suggest that each transfer undergoes a defined set of activities that is usually planned when a transfer is initiated [12, 20] — from onsite work to recruitment and training of the receiving site, formal handover followed by a trial operation when the receiving site demonstrates their capabilities and finally into offsite work. These findings were used for further investigation of project staffing during and after a transfer.

The primary source of observations used in the staffing curves comes from documentation of a software transfer project and additional interviews with experts, which supplement the qualitative observations of the same project

reported in [12]. To understand the common project staffing associated with transfers we have conducted two non-structured group interviews with five experts at Ericsson (referred later to as expert group), i.e. people having had leading roles in software transfer projects. Furthermore, a semi-structured 1.5-hour long interview with the Ericsson development unit manager being responsible for the development unit including the sites in both Sweden and India.

The transfer staffing curves presented have emerged from qualitative inquiry of experience. Perceived staffing situations and their effect on the success of a transfer were modeled and discussed with the experts in the first group interview. The initial staffing curves without preconceived limitations (timeframes, phases, or activities) were drawn on a whiteboard by the researchers who moderated the discussion. Each change in the personnel headcount was mapped to a particular event or activity. The expert group specified duration of each activity by obtaining consensus and providing a rationale behind the answers.

An interview with the development unit manager was then performed to obtain the high-level management perspective. The manager was kindly asked to draw a common and a desired staffing curve using the previously defined phases and activities, and specifying their duration.

The modeled curves were combined and discussed during the second group interview with the experts. While there was a large consistency with the shape of the curves and thus personnel dynamics, several differences were identified. The major differences between the two proposed staffing situations were related to duration of activities – the development unit manager's view favoured a longer transfer time. The development unit manager clarified that these differences are motivated by the natural willingness to reduce the transfer costs by stressing the deadlines. At the same time, the manager suggested that there is a trade-off between the short duration and its consequences. These findings are discussed in more detail in the forthcoming sections.

To improve the validity of the modeled staffing curves, we performed a comparison with the staffing situation in a project through a case study. Although the curve is based on employment reports in the studied transfer project, and it represents another type of transfer project, the main phases, reasons for change and change directions were supported. The differences can be explained by the difference in the transfer conditions — while the modeled curves represent a transfer from scratch (full exchange of the employees), the work in the case study was transferred to a team that had been already involved in several product activities. This is further elaborated in Sections IV and V. The differences in transfer conditions help in understanding the dynamics of employees in different types of transfers, and hence supporting the need for a classification of software transfers.

IV. Classification of Software Transfers

Based on the research collaboration with Ericsson, insights have been gained into different types of transfers, and also how these different types of transfers need to be planned differently. The motivation for the classification is based in the different needs for different types of transfers. The experience from the collaboration together with a comparative analysis of different transfer projects and discussions with the expert group helped us to formulate different types of software transfers. Three main types of transfers were identified: full transfers, partial transfers and gradual transfers (see Figure 3).

To fully understand the difference between the three main types, there is a need to define the transfer object that makes it necessary to plan handover of the work differently. Here, we have chosen to use the words system/product and entity to capture the concept of a transfer object. Thus, entity is defined as follows.

Definition: Entity – An entity is a part of a system (e.g. a subsystem, module or component) or a part of a system development process (e.g. phase or activity) that requires continuous coordination of work among people working on at least one other entity.

A collection of entities can make up parts of or a full self-contained system or development process, where self-contained refers to that continuous coordination is not needed. The latter may, for example, be the case when having well-standardized interfaces between two systems or parts of a system.

For illustration purposes, we assume having a system consisting of three entities. Thus, the system as a whole is assumed to be self-contained in the sense that it has well-defined interfaces and hence no continuous coordination with other sites are needed if the system is developed in one site. On the other hand, if entities are spread across sites then continuous coordination is needed (which is in accordance with the definition of an entity).

The transfer types illustrated in Figure 3 are defined as follows:

1. Full transfer — prescribes full relocation of a system/product from scratch, i.e. moving from co-located development in one site to co-located development in another site. This type of transfer is shown in the upper left of Figure 3 (type 1). The figure illustrates how the system consisting of three entities is moved from Site A to Site B. No development is kept at Site A. Thus, the start state of the transfer project is all development at Site A and the end state is all development at Site B.
2. Partial transfer — prescribes relocation on the entity level, i.e. at least one entity (but not all at once) is transferred from one site to another site. The partial transfer type is shown in the upper right of Figure 3. The figure illustrates a scenario where one of the entities is moved from Site A to Site B. Thus, in the given example, the start state of the transfer project is all development at Site A and the end state is that Entity 1 is developed at Site B and Entities 2 and 3 are still developed at Site A.

   Partial transfers may be different. In the example above, the start state is all development at Site A, and the end state is reached after moving one entity. Other partial transfers may include moving more than one entity, or

starting from a different start state than having everything in one site, or ending up in a different start state, for example having all development in Site B. The key characteristic is that the transfer project is planned for transferring one or more entities, although not all entities at once. The latter is a full transfer (type 1).

A partial transfer implies that there are no plans initially for transferring all development, for example from Site A to Site B.

3. Gradual transfer — means that all entities are planned to be transferred from one site to another site, but it is done in steps. In Figure 3, this is illustrated by having the start state with all development in Site A and the end state being all development in Site B. However, in a gradual transfer there are intermediate steps, where the development is shared across sites.

   This type of transfer is often used when there is a need to build up competence in the receiving site or there is a need to recruit people at the receiving site (Site B) before handing over full responsibility for the product. The actual pace of the transfer may also depend on the intentions at the sending site (Site A), i.e. whether the objective is to downsize or start some new development at the sending site. The gradual transfer is illustrated at the bottom of Figure 3 (type 3).

A key difference between partial and gradual transfers is the objective of the transfer. If the objective is to move all development from Site A to Site B taking several steps, the plan is to make a gradual transfer. However, if the full relocation happens gradually but without initial intention, it means the company executed a series of partial transfers. To summarize, partial and gradual transfers can be organized in different ways:

a. Transferring parts of a product or associated activities from scratch, i.e. moving from co-located into distributed development;

b. Scaling up in one site by transferring additional parts or associated activities, i.e. changing the division of work within distributed development;

c. Transferring the remaining parts of a product or associated activities to one site, i.e. moving from distributed to co-located development.

The difference between partial and gradual relates to the start state and end state of the transfer project. In a gradual transfer the whole transfer from Site A to Site is planned. In a partial transfer, only one step is planned and then other changes may be decided later.

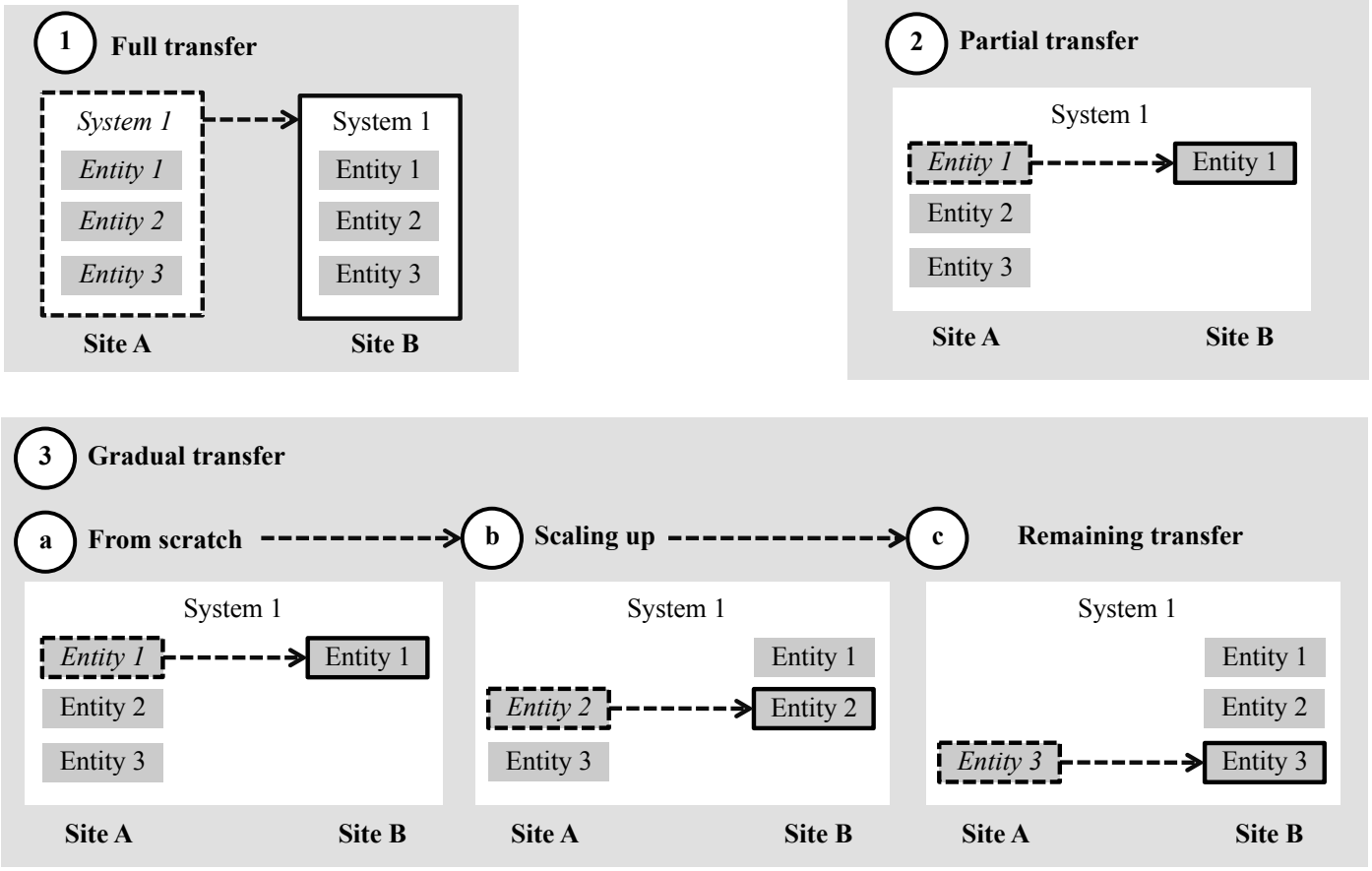

Figure 3. Types of software transfer

A special transfer case is backsourcing or returning previously transferred parts of a product or associated activities to a site where the development has previously been. Backsourcing can be done in any of the three types of transfer types.

The classification of transfer types are defined in response to RQ1, and the types are based on the experiences gained in a three-year collaborative project between industry and academia. The transfer types are orthogonal and capture, to the best of our knowledge, the situations that may occur when transferring software development from one site to another site.

V. STAFFING ILLUSTRATIONS

The classification in Section IV is based on the conjecture that there are indeed differences between the different types of transfers. Thus, it is assumed that the differences are important to take into account when comparing "your case" to a published study if being a practitioner, or when synthesizing evidence in relation to software transfers if you are a researcher. To obtain a first indication of the differences, it was decided to study staffing curves for some of the different types of software transfers, and hence illustrate that there is a strong need that it is clearly documented which type of transfer it is when describing the context of an industrial study.

A. Idealistic view

A perfect handover of work would mean that the work was done in one site one day and the next day the work is done at another site. This is, of course, completely unrealistic in most cases, and, in particular, when it comes to knowledge-intensive work, such as software development. In Figure 4, a perfect handover is illustrated by the dashed staffing curves. The challenge here comes from the nature of software activities due to which newly employed software engineers or engineers being unfamiliar with the product are unable to continue the development of a software product with the same speed and accuracy from the first day. Thus, there is a transfer phase, in which the original employees are still involved in the project, and the employees at the other site are also involved in the project, resulting in the direct transfer costs of cumulative resources. Thus, it is important to set realistic goals regarding the transfer and the corresponding transfer curve. The solid lines in Figure 4 illustrate an idealistic view when it comes to transferring software development between sites. The idealistic view may act as a vision of what we want to achieve. The transfer can be described by using the following concepts:

- The staffing before and after the handover
- The point where the two staffing curves meet
- How fast one curve goes down and another goes up
- How early the new employees are involved
- How long the old ones are kept.

Handover is here referred to as the time of transfer of the formal responsibility for the software product being transferred.

In the beginning of our investigation we asked an expert group to illustrate perceived target staffing curves, i.e. the staffing patterns they perceive is the target to aim for in order to achieve an as fast and cheap transfer as possible. The curves in Figure 4 illustrate a transfer of type 1 according to the definitions above, i.e. a full transfer.

The curves show a short transfer time, fast growing build-up at the receiving site and fast reduction of employees at the sending site. Note that in this pattern the curves meet before the handover at the rate of 65%. These curves were recognized as idealistic. In particular, in the following sections we explore curves based on actual experiences of the expert group and the development unit manager.

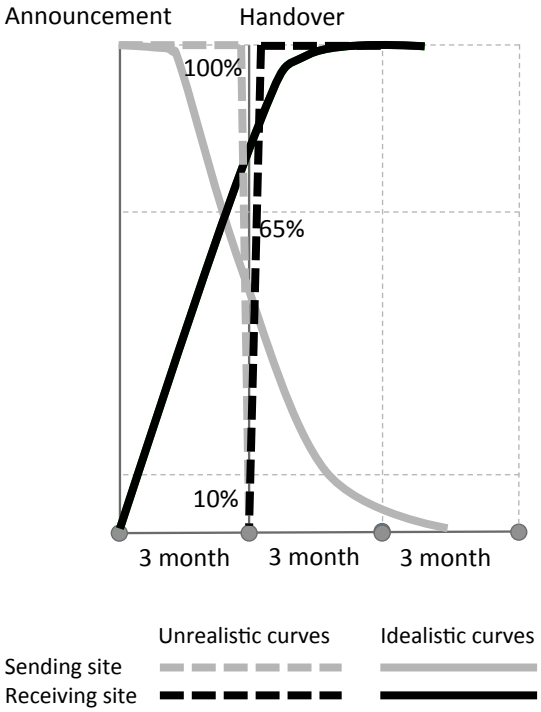

Figure 4. Cheap and fast transfers

B. Full Transfer

The challenges, related to project staffing during and after the transfer, were discussed with the expert group and the development unit manager at Ericsson and several best practices were identified. Since transfers are in many cases initiated from scratch (having 100% of employees in the sending site at the moment of announcement), the practices were developed for such transfers that require full exchange of employees (transfer of type 1). The following are the lessons learned and suggestions for the recommended staffing patterns and the underlying reasoning behind these suggestions.

Duration of a transfer: Transfers require time and it is important to plan the cut-off day when all resources at the receiving site are employed and trained. The total recommended duration of a transfer is 18-20 months including the prolongation of few experienced employees, preferably relocated to the receiving site, for the last six months. The actual length of the transfer is of course highly dependent on the size of the product being transferred.

Additional mentors versus prolongation of original resources: A transfer has a major impact on on-going development, since employees from the sending site are involved in both current work and training of the new people taking over the responsibility. This means that pulling out resources too soon can have a devastating effect on the product development. It is thus recommended to either dedicate additional resources for training and mentoring purposes or reduce the size of the delivery planned during and directly after the transfer.

Recruitment: Recruitment (if needed) of employees in the new location can be challenging and shall be planned in advance not to endanger handover deadlines. On the one hand, this may indicate that the announcement needs to be done earlier, but on the other hand a too early announcement may create negative consequences in terms of key people moving away from the software product to be transferred. This is a delicate balance! It is also important to have the receiving resources in place, while performing the training. Although this is not critical in the first months, while the product documentation and transfer plans are being prepared, active ramp-up of employees is expected to take place well before the handover day. It is advisable to promote people if possible within the organization to have the key resources in the beginning of the transfer. The remaining 20-30% of the required employees have a smaller impact and can be recruited gradually.

Scaling down at the sending site: Downsizing of the development of the product at the sending site can be fast. However, it is recommended to be organized after a trial operation, when the receiving site has demonstrated their capabilities. Thus, the sending resources are still available if fire fighting is required. It should be stressed that downsizing here does not imply laying people off. The personnel are primarily freed up to work on other products.

Support: Several employees shall be allocated to support the product development after the handover to ensure a smooth transfer. Dependent on the criticality of the product, it is advisable to relocate several key experts with the product to ensure the continuity of the product expertise. However, this requires significant investment. Although it was seen as unrealistic to reduce the length of support without any negative impact on the product, the number of employees from the sending site remaining with the product is seen as an important potential improvement, i.e. the target is to have fewer people from the sending site in support of the further development of the software product. One of the possible suggestions prescribes adopting more efficient training approaches for the ramp-up of competence at the receiving site.

Based on these lessons learned the curves in Figure 5 were modelled based on experiences of the industrial participants involved as recommended staffing patterns including the potential improvement (illustrated with a dashed line denoted target improvement). The curves illustrate a transfer of type 1. To summarize, a full transfer from scratch starts six to nine months prior to the handover for employing the necessary people at the new location and managing to provide the necessary training. The curves meet at the level of 80% of the initial employment at the time of the handover with the substantial amount of experienced resources and new resources to handle the on-going development with the forthcoming trial operation. At the end of the trial period most of the original employees are then released. Dependent on the success of the trial, the organization can decide on the reasonable amount of experienced resources to be kept in the project for an extended period of time until the final cut-off.

The full transfer curve based on the experiences of the development unit manager and the expert group may be viewed as a baseline curve, since it covers the simplest case to describe, i.e. full transfer.

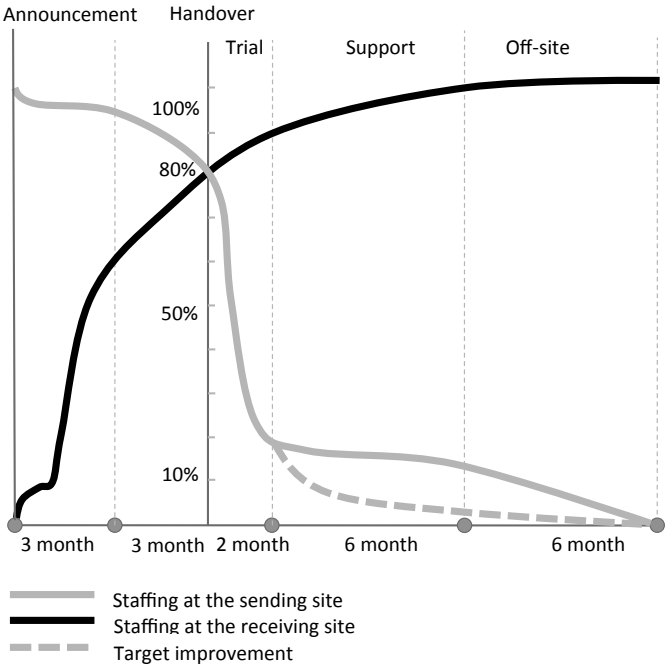

Figure 5. Recommended staffing and targeted improvements

C. Partial transfer to co-located

The next illustration is from a case study of a partial transfer. The situation in the beginning is that two sites are jointly developing a software product (start state) and the intention is now that the parts remaining in site A should be transferred to the other site (site B) (end state). The curve in Figure 6 shows the staffing in the project. This is not a gradual transfer since this was not originally planned when moving into distributed development between the sites.

Due to the problems of finding the right people with the right competence [12], the personnel needed, ended up being recruited quite close to the formal handover. In this transfer project, the main build up in the receiving site was done only three months before the handover of the responsibility, which was regarded as rather late. It should be also noted that the total amount of resources at the end of the transfer decreased by 20% in comparison with that at the beginning. This is mainly due to major changes in product development, which required additional resources, were implemented before the transfer.

The sending site is located in Sweden and the receiving site in India. It should be noted that at the time of the announcement 55% of the employees working on the software product were located in Sweden and 45% in India. The situation relates to a transfer of type 2c, i.e. it is a partial transfer since it was not planned originally to transfer the whole product and the full product is at the receiving site after the transfer. In Figure 6, 100% refers to the number of employees at the time of the announcement of the transfer. Thus, the sum of percentages during the transfer may be higher than 100% due to that more people are involved in the work than at the time of the announcement.

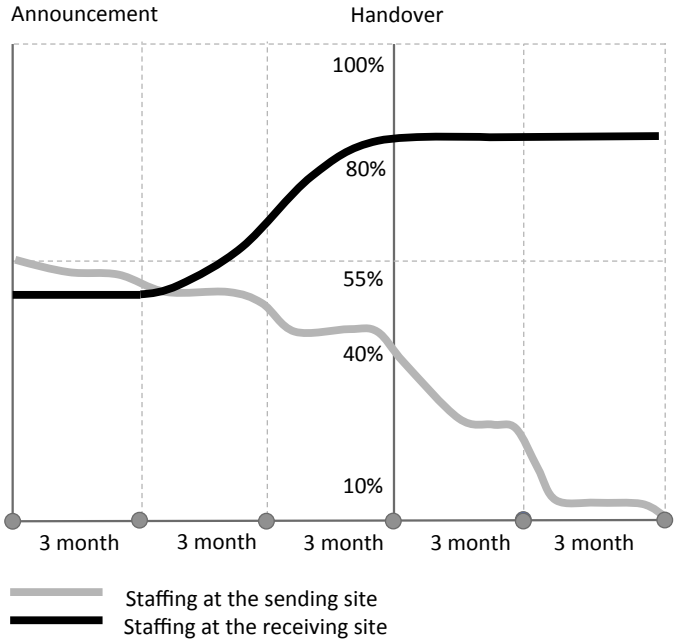

Figure 6. Actual staffing in a partial transfer

The curve suggests that shortly after the transfer was announced, several employees were relocated to other projects and the process of freeing up the sending resources and handing over responsibilities continued gradually for nine months. After the formal handover the receiving site demonstrated their capability to handle the work with the product independently in a trial operation. During this time most of the resources in the sending site were pulled out of the project. The remaining 10% of the employees at the sending site supported the project for another three months to ensure a smooth transfer and help the receiving site in case of emergency or unforeseen situations. This can be characterized as a pattern of gradual decrease in the number of experienced employees.

D. Summary staffing curves

As can be seen from the staffing curves in Figures 5 and 6, they are quite different, which really is not a surprise. However, it illustrates that there is a need to distinguish between different software transfer types. It is clearly insufficient to describe a software transfer in a research paper without providing more details along the lines of the classification proposed here. It should be remembered that Figure 5 is based on qualitative data and Figure 6 on quantitative data, and hence not making them fully comparable.

VI. Discussion

In addition to only illustrating that there is a difference in the staffing curves; the curves provide some opportunities to reflect on the differences in staffing in the two different scenarios: full transfer vs. partial transfer.

A. Transfer costs

The staffing curves described in this paper can also be used to quantify the direct personnel costs of transferring a product from one site to another. It is obvious that the costs during the project grow as the cumulative staffing curve suggests. However, the differences between the cases illustrated in Figure 6 in relation to the case in Figure 5 must be kept in mind. In Figure 6, a transfer to one site from having worked jointly between the sites is illustrated. While in Figure 5, a full transfer from one site to another site is illustrated (type 1 transfer). These curves show how staffing patterns can be illustrated in a software transfer situation, and hence answering RQ2.

While manufacturing thinking may lead one to believe that the benefits of relocating the work from a high cost to a low cost location can be estimated through a simple comparison of development costs (as all too often measured by the hourly rate), we emphasize the importance of avoiding underestimation of the value and effect of knowledge and experience on the delivered software product, confirming related studies that have identified the link between experience, expertise and product quality [13, 14]. To illustrate this we denote an ideal view of a transfer in Figure 4 and compare it with the more realistic view based on industrial experience in Figure 5. We presented the reasons for e.g. not shortening the duration of a transfer and the necessity for keeping the experts as a backup. Although these are costly transfer practices, they aim to mitigate the risks of failure of overall product development. It is also worth mentioning that the higher the levels of client-specific and product-specific knowledge requirements, the higher are the transfer costs [23]. Thus, before transferring a software product, organizations ought to estimate the acceptable decrease in productivity for the coming deliveries and take measures to ensure the service quality by e.g. prolonging involvement of the experienced developers or relocating some of them with the product. Organizations may also choose which transfer type best suit the current situation, and perhaps start small, scale up as the knowledge and experience grows, and transfer the remaining parts with the responsibility, when the quality and efficiency are not in danger.

B. Effort distribution

The effort distribution becomes different depending on the type of transfer. When summing the number of employees in each site depicted in Figure 6, we can see that the maximum effort is around 130%, i.e. when building up in the receiving site and freeing up personnel in the sending site. The maximum is reached before the official handover of responsibility. This is reasonable since the receiving site is

familiar with the software product. This also means that the personnel at the sending site can be phased out quite soon after having the personnel in place at the receiving site. In this case, there are no major challenges when it comes to product knowledge, although the receiving site is not familiar with all parts of the product. However, given that the product previously was handled across sites the receiving site has sufficient knowledge as well as good personal contacts with personnel at the sending site.

In the case of a full transfer as illustrated in Figure 5, the situation becomes different. There is a major challenge in ramping up the personnel in the receiving site quickly. It may be argued that more time is needed to ramp up, but if announcing it earlier then there is a risk of personnel at the sending site moving out before there is personnel available at the receiving site. Thus, it becomes a delicate balance of when to announce a transfer. If it is too early then personnel may move to other projects before the personnel is available at the receiving site, and if it is too late it will be very hard to get the needed resources in place in the receiving site. The full transfer creates a higher maximum of personnel. In this case (Figure 5), the maximum is at 160% at the handover. This is due to that personnel in the sending site cannot leave until the personnel at the receiving site has sufficient knowledge of the product being transferred to them. The potential improvement is indicated in Figure 5 with the dashed line. The curves in Figure 5 illustrate the effect in terms of effort required to perform a transfer of type 1.

The different types of transfers in combination with ensuring transfer of knowledge about the software product generate different effort distributions. This must be taken into account when transferring software products whether being full transfers from scratch, partial transfers or gradual transfers. Thus, once again stressing the need to handle different types of software transfers differently.

C. Future research directions

Our investigation is only one step towards a better understanding the impact of different types of transfers and of the consequences of software transfers and how project staffing affects the post-transfer performance. For future research we suggest collecting more empirical data regarding the effect of different types of transfers on the amount of delivered functionality and its quality during and after the transfer. To foster further research we put forward the following conjectures that emerged from our investigation, which are directly related to the different types of software transfers:

Conjecture #1: Early discontinuity of product expertise negatively affects the quality of the software delivered by the receiving site and thus increases the costs of non-quality.

While earlier disengagement of the developers from the sending site has been recognized as the potential improvement, it might appear unrealistic. Thus, more research is necessary in relation to the possibility of better training before the handover and to minimize the mentoring efforts after the handover. Furthermore, it is important to provide incentives for individuals to contribute to a successful transfer.

Conjecture #2: Involvement of key experts in training and mentoring pays off.

While prolongation of well-paid staff may seem unreasonable and counterintuitive, we feel that this might save costs in the long term through more effective training results. Thus, cost-benefit analysis and comparison of the different types of transfers may help understanding what contributes to the success of a transfer best.

Conjecture #3: Transfer of people with the software product pays off.

If the receiving site has little or no prior experience with the software product being transferred then it is most likely worth the extra costs of having some key people being transferred with the software product. This could, for example, be done based on short term contracts (months or years depending on the size and complexity of the software product being transferred) to ensure an as smooth transfer as possible.

D. Summary

It is clear from the staffing curves that different types of software transfers must be handled in different ways. This is an area for future research and the classification presented in Section IV forms the backbone of understanding the differences.

VII. CONCLUSIONS

In this paper we defined a number of transfer types (as response to RQ1) and discussed the effect of staffing on transferring software work from one location to another. In particular, we aimed at capturing different staffing patterns (in relation to the classification of different transfer types) and understanding their effect on the execution and success of the transfer. In response to RQ2 we illustrated two different transfer types based on a qualitative inquiry and a case study, and as a way of illustrating the need for the classification. The staffing curves are modeled in Figures 4-6, which provide an insight into the staffing patterns. In result, we derived recommendations in relation to the dynamics of employees in the sending and the receiving sites, which are different for different transfer types. The empirical observations based on experiences from executing software transfers between a Swedish site and an Indian site within Ericsson motivate the following general conclusions:

- Transfers are different, and hence it is important to be able to classify different types of transfer. This is important for both researchers and practitioners to being able to understand similarities and differences, and hence to be able to make more effective use of the findings from different studies of software transfers.
- Transfer of software work takes time. Thus substantial effort by experienced staff shall be planned for training and mentoring activities. In addition, keep the expectations of immediate productivity after the handover low as the most likely less experienced (at least regarding the specific software product) employees climb their learning curve.

- Continuity of product expertise is crucial. Thus, it is not recommended to pull out the experienced employees early in a transfer, although the handling of personnel will depend of the actual type of transfer. Consider relocating several key people with the product in order to ensure the effective knowledge transfer as not everything can be documented.

As stated above, not all transfers are the same. We have illustrated two different cases in relation to the classification in the paper. First, a full transfer (type 1 transfer) was described, which could work as a baseline as it is the hardest case to master, although easiest to describe. Secondly, we illustrated a transfer of a software product moving from joint development across two sites to being developed only in one site (type 2 transfer). This creates certain challenges, but it is less challenging in comparison to the first case in which a software product is fully transferred from being developed in one site to being developed in another site. The first case puts high requirements on planning knowledge transfer and potentially also to transfer people together with the product.

Finally, we stress that there is a trade-off between staffing investments during and after the transfer and ability to maintain the same level of service in terms of productivity and quality of the product development. In fact, our empirical observations suggest that success of a transfer depends on smart decisions regarding transfer project staffing. This leads us to conclude that companies unable to foresee the challenges of transferring knowledge-intensive work or those unprepared to invest into the costly transfer activities may fail to maintain the desired level of services and thus fully exploit the expected benefits. Given these challenges with software transfers, it is crucial to be able to distinguish different software transfers from each other. Thus, the classification of software transfers types provided here form an important basis to understand and ultimately master software transfers successfully.

ACKNOWLEDGMENT

We would like to thank Ericsson employees for their active participation and support of this research.

Ericsson Software Research and the Swedish Knowledge Foundation, under the grants 2009/0249 and 2010/0311, fund the research.

REFERENCES

[1] S. Overby, "The Hidden Costs of Offshore Outsourcing", CIO Magazine, September 2003.

[2] E. Carmel and P. Tjia, "Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce", Cambridge University Press, NY, 2005.

[3] Ericsson Annual Reports. http://www.ericsson.com/thecompany/investors/financial-reports/annual-reports (visited May 30, 2012).

[4] Swedish Industry Has a Software Soul, Swedsoft, 2010; http://www.swedsoft.se/Mjukvaran_%C3%A4r_sj%C3%A4len_i_svensk_industri.pdf (in Swedish) (visited May 30, 2012).

[5] K. Petersen and C. Wohlin, "Context in Industrial Software Engineering Research", Proc. 3rd Int. Symp. on Empirical Software Engineering and Measurement, 2009, pp. 401-404.

[6] D. Smite, C. Wohlin, R. Feldt and T. Gorschek, "Reporting Empirical Research in Global Software Engineering: a Classification Scheme", Proc. IEEE Int. Conf. on Global Software Engineering, 2008, pp. 173-181.

[7] D. Šmite, C. Wohlin, R. Feldt, and T. Gorschek, "Empirical Evidence in Global Software Engineering: A Systematic Review", Journal of Empirical Software Engineering, 15(1), 2010, pp. 91-118.

[8] D. Šmite and C. Wohlin, "A Whisper of Evidence in Global Software Engineering", IEEE Software, 28(4), pp. 15-18, 2011

[9] E. O. Conchuir, H. Holmström, P. J. Ågerfalk and B. Fitzgerald, "Exploring the Assumed Benefits of Global Software Development", Proc. IEEE Int. Conf. on Global Software Engineering, 2006, pp. 159-168.

[10] D. W. Karolak, "Global Software Development: Managing Virtual Teams and Environments", IEEE Computer Society, 1998.

[11] J. D. Herbsleb and A. Mockus, "An Empirical Study of Speed and Communication in Globally Distributed Software Development", IEEE Transactions on Software Engineering, 29(6), 2003, pp. 481-494.

[12] D. Šmite and C. Wohlin, "Software Product Transfers: Lessons Learned from a Case Study", Proc. IEEE Int. Conf. on Global Software Engineering, 2010, pp. 97-105.

[13] S. L. Pfleeger, "Software Metrics: Progress after 25 Years?" IEEE Software, 2008, pp. 32-34.

[14] N. Nagappan, B. Murphy and V. R. Basili, "The Influence of Organizational Structure on Software Quality: An Empirical Case Study. Organization", Proc. of the Int. Conf. on Software Engineering, 2008, pp. 521-530.

[15] A. Mockus and D. M. Weiss, "Globalization by Chunking: A Quantitative Approach", IEEE Software 18(2), 2001, pp. 30-37.

[16] R. Kommeren and P. Parviainen, "Philips Experiences in Global Distributed Software Development", Empirical Software Engineering, 12(6), 2007, pp. 647-660.

[17] A. Boden, B. Nett and V. Wulf, "Coordination Practices in Distributed Software Development of Small Enterprises", Proc. IEEE 2nd Int. Conf. on Global Software Engineering, 2007, pp. 235-246.

[18] R. Jabangwe and D. Šmite, "An Exploratory Study of Software Evolution and Quality: Before, During and After a Transfer", Proc. IEEE Int. Conf. on Global Software Engineering, IEEE Computer Society, 2012, pp. 41-50.

[19] A. Banerjee, and S. A. Williams, "Using Offshore Analytics to Identify Determinants of Value-added Outsourcing", Strategic Outsourcing: An International Journal, 2(1), 2009, pp. 68-79.

[20] F. Dong, M. Li, J. Li, Y. Yang and Q. Wang, "Effect of Staffing Pattern on Software Project: An Empirical Analysis", Proc. of the Third Int. Symp. on Empirical Software Engineering and Measurement, 2009, pp. 23-33.

[21] D. Šmite and C. Wohlin, "Lessons Learned from Transferring Software Products to India". In: Journal of Software: Evolution and Process, Vol. 24, No. 6, pp. 605-623, 2012.

[22] D. Šmite and C. Wohlin, "Strategies Facilitating Software Product Transfers", IEEE Software, 28(5), pp. 60-66, 2011.

[23] J. Dibbern, J. Winkler, and A. Heinzl, "Explaining Variations in Client Extra Costs between Software Projects Offshored to India", MIS Quarterly, 32(2), 2007, pp. 333-366.