Collaboration in Global Software Projects at Siemens: An Experience Report

Matthew Bass
Siemens Corporate Research, Inc
Princeton, NJ
matthew.bass@siemens.com

James D. Herbsleb
Institute for Software Research International
Carnegie Mellon University
Pittsburgh, PA
herbsleb@andrew.cmu.edu

Christian Lescher
Siemens AG, Corporate Technology
Munich, Germany
christian.lescher@siemens.com

Abstract

As a globally operating company with about 30,000 software engineers worldwide, Siemens has accumulated a wide variety of experiences in global development. Many individuals and organizations have adjusted their practices to deal with the challenges related to the geographic distribution of the development effort. From a corporate perspective, Siemens has accumulated a rich base of knowledge about global development and how to approach it successfully. The Siemens Software Initiative - a company-wide improvement program for software development at Siemens - has worked on collecting this widely-distributed knowledge and synthesizing it in a form accessible to the wider software development community. In this paper, the approach as well as key learnings in people and communication-related aspects of collaboration are summarized.

1. Introduction

Software development at Siemens is increasingly a globally-distributed undertaking. A variety of motivations, including cost competitiveness, ability to use the most appropriate resources regardless of location, and co-location with important markets and customers are driving this move. Competitiveness now and in the future requires a world-class competence in this new global development paradigm. Global software development (GSD) also introduces big challenges in software development. Communication and coordination are significantly harder to manage when a project is distributed over multiple geographic sites sometimes spanning multiple countries or even continents [2][3][4].

Siemens is a globally operating company with about 30,000 software engineers and spends ca. 3 billion Euros annually on software development costs for our software-based systems, plants, and services. The range of these products is very broad and have a worldwide market, including e.g. automotive systems, building technologies, communication systems, just to name a few.

To meet the needs of the Siemens Business Groups, the Siemens Software Initiative – a company-wide improvement program for software development at Siemens, with an international network, including representatives in the various regions – has started a project to continuously improve collaboration in global software development Siemens-wide and to promote best practice sharing within Siemens. The Siemens Software Initiative addresses multiple aspects of collaboration (see Figure 1). Collaboration in this sense involves people aspects such as communication, teambuilding and competency management as well as engineering aspects like architectures, development processes and tools.

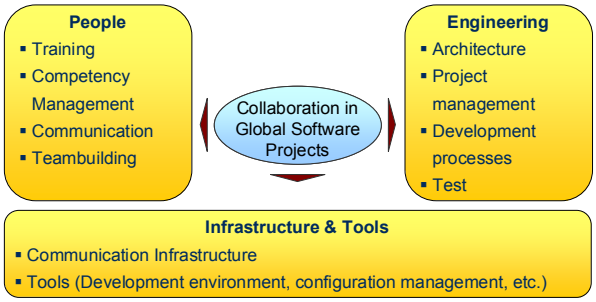

Figure 1. Aspects of Collaboration in Global Software Projects

Because of its size and diversity, Siemens has accumulated a wide variety of experiences in global development. Many individuals and organizations have adjusted their practices to deal with the challenges related to the geographic distribution of the development effort. From a corporate perspective, Siemens has ac-

International Conference on Global Software Engineering(ICGSE 2007)
0-7695-2920-8/07 $25.00 © 2007 IEEE

cumulated a rich base of knowledge about global development and how to approach it successfully, and wanted to take advantage of it in a way that would be useful to others in the community.

In a joint cooperation with Siemens Corporate Research, and Carnegie Mellon University, the Siemens Software Initiative has been working on collecting this widely-distributed knowledge and synthesizing it in a form accessible to the wider software development community. This knowledge has been captured in the form of specific "best practices" (see Figure 2). To date, 18 best practices have been identified across 4 Siemens Business Groups, 8 of these practices are presented in this paper.

This experience report will summarize selected practices identified by the Siemens Software Initiative, focusing mainly on people- and communication-related aspects of collaboration.

2. The Approach

For the purposes of this effort a "best practice" is a practice that was successful in the context of one or more multi-site projects in dealing with specific GSD related issues. Below we describe the approach we used in collecting these practices. The resulting best practices were presented and discussed in Siemens-internal workshops with participants of many different business groups and confirmed to be useful.

The data for each best practice was collected using semi-structured interviews. During the interview we collected the information summarized in the template shown in Figure 2. Even though in this paper we are using brief and informal descriptions of best practices, this template represents the underlying structure of results. The best practice template was adapted from a widely-used template for design patterns [6].

We established a set of criteria to help identify appropriate best practices. First, we needed to interview at least two people for each candidate practices. Typically we interviewed the originator of the practice, and at least one additional "user" of the practice that was not the originator. Second, there needed to be "adequate" experience with the practice. While we had no concrete definition of "adequate" we considered practices that either had a long history with a larger project or a practice that had been used on multiple projects. Additionally, we selected practices that carried a substantial benefit, addressed a common problem of global development, and could reasonably be applied by a significant range of Siemens development teams.

Best Practices Template

Practice Name and Classification: Every practice should have a descriptive and unique name that helps in identifying and referring to it. Additionally, the practice should be classified according to a classification such as the one described below. This classification helps in identifying the use of the practice.

Intent: This section should describe the goal behind the practice and the reason for using it. It resembles the problem part of the practice.

Motivation: This section provides a scenario consisting of a problem and a context in which this practice can be used. By relating the problem and the context, this section shows when this practice is used.

Prerequisites and limitations: This section describes situations in which this practice is usable, describing the prerequisites that must be in place in order for the practice to be useful, and any limitations that have been observed. It represents the context part of the practice.

Benefits and other consequences: This section describes the benefits, side effects, and trade offs caused by using this practice.

Implementation: This section describes the implementation of the practice, and represents the solution part of the practice. It provides the techniques used in implementing this practice, and suggests ways for this implementation.

Experiences: This section includes a description of the experiences that Siemens has had with this practice as well as experience-based comments about similar or related practices. It also provides contact information for projects and people who have made use of this practice.

Related Practices: This section includes other practices that have some relation with this practice, so that they can be used along with this practice, or instead of this practice. It also includes the differences this practice has with similar practices.

Sample Artifacts: A collection of relevant examples, such as templates, plans, processes, minutes, tools, and anything else that would be helpful for allowing someone else to understand and adopt the practice.

Figure 2. Best Practice Template.

Potential best practices were identified in several ways, including presentations at company-wide global software development workshops, internal reputation for having effective practices for addressing specific problems, and nominations by managers and executives. There were several candidate practices that were anticipated to be useful, but lacked the experience cur-

rently to be included in the first round of best practice collection.

The following three sections contain a description of people and communication related practices that have been identified at Siemens in the areas of maintaining cross-site relationship, selecting the right communication media, as well as training and teambuilding.

3. Maintaining Cross-Site Relationship

The establishment of relationships amongst members of teams from different geographic locations greatly aids in the coordination. There are many ways to establish these relationships, however, each potentially having different costs and are effective in different ways. Below we summarize three such practices used within the projects interviewed.

Practices that have shown their effectiveness at Siemens include:

- Onsite management visits
- Cross-site delegation
- Unfiltered communication

Onsite management visits are used to monitor the status of the project, ensure progress and address issues for remote project sites. Starting with the project start up the project manager visits the remote sites once every 6 - 8 weeks. They have detailed status meeting during these visits as well as meetings to address other topics (e.g. budget or schedule issues, risk management, planning for the next release, or logistical issues such as sending test equipment or dealing with customers). During these visits sub-project leaders also visit with various team members and have technical exchanges and presentations. In addition to the project manager, the line manager visits the sites a couple of times a year. The frequency is about every 6 weeks at the beginning of the project and changes to every 8 weeks as the project progresses. During these status meetings every feature coordinator[1] presents the detailed status on their feature. In between these onsite visits each feature coordinator gets a weekly status report from the relevant team members. This is usually done via a teleconference in the case where people are geographically distributed. Onsite management visits allow for exchanges that would not otherwise be able to take place: A major portion of the exchange occurs outside of the meetings and needs to be allowed for. Therefore it is important to build free and social time into the agenda.

Cross-site delegation is another way to establish personal relationships and to achieve a better integration of multiple geographically distributed teams. It is basically the delegation of individuals from a central site to a remote site (or vice versa) and helps to establish communication across sites that can be useful if cross-site information is needed at any point in the project (e.g. finding particular expertise).

Cross-site delegation could happen in many different ways for different reasons. A member of a remote site could come to a central site, or a member of a central site could go to a remote site for a defined period of time. The delegate to a remote site could take any number of roles (e.g. developer, feature coordinator, sub-project lead, or site manager) depending on the need. This is often a career step for the delegate (typically a high-potential candidate). The delegate to the remote sites is involved in all meetings at the remote site and becomes a contact person for the project lead as well as for the line manager. Typical length of a delegation is 1 – 2 years. On return, the delegate usually becomes a key person for cross-site collaboration.

Unfiltered communication is similar to the *Onsite management visits*.

Specific about this practice is that the line manager meets directly with the developers from the remote sites to become aware of and help solve the issues that the developers are experiencing. The line manager visits the sites once or twice a month and meets with the developers there. There are no explicit arrangements with the local managers regarding the agenda. The developers set the agenda and discuss any issue that is hampering their work. Topics can range from issues such as slow LAN connection, too much work, or hindrance to effective working. The line manager views his job as the person who deals with these issues so the developer can continue to be productive. The line manager typically brings 1 – 4 people with him depending on the needs of the project. These might be feature coordinators, sub-project leads, or technical people. The main benefit of this practice is that the line manager has overall product visibility and so can keep the needs of the product (across projects) in mind. The manager is in a position to understand what the issues are and to rectify these issues. It also becomes possible to identify and make connections across sites. If there is a need on one site the manager is in a position to identify the appropriate person from another site to help out. This creates a sense of team and improves morale with the developers. Furthermore, this makes it

[1] Project member, who takes over the responsibility for a feature during the whole development lifecycle

clear that they are important and their opinion and needs matter.

A common practice in case of a supplier-relationship to the remote site is also to establish a *Supplier Manager* to manage the relationship with the remote site: The customer provides a supplier manager, who is primarily located at the customer site, to manage the relationship with the offshore partner. The supplier manager is not directly involved in the technical decisions (except for oversight and review), but serves as an information conduit and keeps track of progress at the remote site.

4. Selecting the Right Communication Media

Awareness across sites is often an issue [2]. Knowing who is working on what, knowing who to contact in case of a question, or finding the status of tasks across sites can be difficult. A couple of the best practices collected made effective use of technology such as wikis to help overcome these difficulties. In this section we summarize two such practices.

Two specific practices that have demonstrated their effectiveness at Siemens are:

- Distributed pair programming
- Urgent request

Distributed pair programming is an application sharing-based approach, where pairs of geographically distributed developers practice virtual pair programming using NetMeeting or other collaboration software [7]. This practice is particularly helpful in case of component code with important dependencies on code developed at another site. In order to avoid delay, the developer asks for an instant review by a developer at the other site who understands the relevant code in depth. The two developers jointly review the code, perhaps making a few changes, and both are satisfied that the code is free of problems. The benefit of using distributed pair programming is that potential conflicts can be eliminated very quickly, saving test and fix time later. It appears to enable development of a single component across sites, which is ordinarily extremely difficult given the density of interdependencies in intra-component technical work. As a prerequisite, the distributed pair programming practice depends on personal and reciprocal relationships among the developers. It may not be appropriate early in a project before these relationships have had a chance to develop. The "collaboration maturity" of a team must be very high [10]. Furthermore, developers must be willing to undertake the pair programming work either as an unscheduled interruption of their ongoing work, or agreeing to schedule a session with very little lead time. The technique is likely to work only for developers who perceive benefit for themselves – either because they won't have to fix the code later, or because they want to receive as well as give technical assistance.

While not precisely pair programming, the tools and practices can be useful in other circumstances. For example, they have proven useful where a tester wants to show a developer the execution of a test case, especially when the developer has been unable to reproduce the bug. Developers can share the application being developed with each other, show them how it works, what the interface looks like.

Urgent request is a broadcast mechanism for requesting urgent information for a project from a volunteer group with specific knowledge. This practice aims at promoting unplanned communication in case that a member of a project has an urgent need for information or advice about a particular technology, tool, or product, and would benefit from quick response.

The primary prerequisite for the urgent request practice is that some distribution mechanism must exist or be created. It is necessary to have in place a network of motivated volunteers with a wide variety of technical expertise from around the company. Various business units may be able to meet this prerequisite in a variety of ways. Some organizations may have networks of internal consultants who could form the core of the expertise network. Others may have wikis or distribution lists that are directed towards groups with expertise in particular technologies, markets, products, standards, and so forth. It is important that this mechanism is used for urgent requests only, since the broadcast mechanism necessarily goes to a fairly large number of people. Receiving a significant number of non-urgent requests would be substantially demotivating to the volunteers. Volunteers also find it demotivating if they only receive requests for which they do not have the correct expertise to provide assistance. Some mechanism for targeting requests is therefore advisable.

One convenient implementation is a web form that is readily accessed. It relieves the user from having to remember the correct distribution list name, and provides appropriate cautions that it is for urgent requests only, and will be sent to many people and will be visible on the web. If question topic is readily identifiable as belonging to a particular category of information, it will be sent to the appropriate people. If it is not clear that it matches any of the categories, it will be sent to all members of all categories.

The difficult part of the implementation is not setting up the tools, but rather recruiting a suitable set of volunteers. This has been accomplished in the original organization by recruiting volunteers willing to receive e-mail in a number of specific topic areas. E-mail requests stressing the value of this service, the need to make expertise widely available, and encouraging voluntary participation may be successful in many organizations. Management in the existing organization urged participation early on, finds that this is no longer necessary, as the practice has become self-sustaining.

The urgent request functionality requires a culture in which people are willing to share information, and help each other out. The typical urgent request reply takes only a few minutes, and therefore has very little cost for the person providing help. As long as the number of requests is not too large (say, 5 a week or less), it does not impose a significant burden on volunteers. It takes only a few seconds to determine if one has the expertise to help when receiving a message. Experience has been that requests are used judiciously, and volunteers have generally not withdrawn from the lists. Everyone with experience with the Urgent Request systems agrees that archiving the replies would not be helpful. The requests are so different that it is extremely rare for the same request to arise twice. There is too much chance of old, outdated information surviving in the archive, and too small a likelihood that the archive would be useful. In fact, time spent searching the archive would be likely to be wasted and demotivating. In the organization in which the urgent request mechanism has been implemented, they report that they get an average of 7 offers of help in the first 3 hours, half of which are helpful. One side effect is that questions asked and answered sometimes extend the internal social network. People with related expertise identify each other, and continued fruitful interaction about technical issues is sometimes a very useful by-product.

5. Training and Teambuilding

Kicking off a project using geographically distributed teams can be problematic. It can take a long time to transfer technical or domain knowledge, knowledge about the processes or infrastructure to be used on the project, or knowledge about the individuals involved in the project across sites. Several practices identified are aimed at speeding up that process. Three of these practices are:

- Tailored training
- Co-located analysis phase
- Goal implementation planning workshop

Tailored training is used to train the project in technologies that they are not sufficiently familiar with, and develop a common understanding for how to apply these technologies to this project, while bringing project members from various sites together in one place for the duration of the training. This is particularly helpful, when new technologies are being introduced in a globally distributed project.

During the analysis phase the team takes part in several training courses. The training program includes standard trainings (e.g. UML training), specific trainings (e.g. technology background for the project) as well as tailored trainings. The tailored trainings are designed to teach the application of the standard training contents to the specific project, e.g. UML as it pertains to this particular system. This can be designed for example using a simplified version of the target system, using the real tool infrastructure of the project. This enables the attendees to begin to think about applying these approaches to the problems that they focus on. As experience has shown, discussions occur about how to do X or Y in their project, and to some extent the design process already begins during the class time.

As a major benefit out of this practice, team members develop a common understanding how they are going to apply particular technologies to the project. In this way, the team members are able to begin to discuss particular design decisions during these courses because the training used the system to be built as a context for the training. This practice also develops personal relationships between individuals across sites.

Co-located analysis phase is a practice in which team leads from both the remote and the central site are co-located during the analysis phase to jointly develop the functional specifications. It is a well-proven practice to bring together teams that have not previously worked together, develop adequate working relationships, familiarity with the system to be developed, and an understanding of the specific areas of responsibility. This is particularly helpful, if the teams on the project are located in various development sites around the world and they have never worked together before, or partly are not familiar with the system, or in case of significant changes in the system, e.g. a new architecture. For the co-located analysis phase of the project, key members come together at one site (e.g. for a duration of 3-4 months). During this time the team takes the high level architecture and requirements (defined prior to this phase) and develops functional specifications for their respective functional areas in mixed teams of functional experts from multiple sites. The primary prerequisite for this practice is that the high-level architecture has been defined and is fairly stable.

As the activities focus on developing a functional specification for the functional areas of the architecture these areas need to be fairly well defined. This practice helps to build personal relationships, develop a common understanding, and achieve a high acceptance of the architecture within the team, since they built it together.

Goal implementation planning workshop is a practice that has successfully been used for establishing a cross-site team and plan to achieve objectives that are given to the project. If management gives a product objectives such as "lower defect density" that requires cross site cooperation to execute, cross site teams will be established to develop a plan for how they would achieve these goals. During a semi annual meeting goals are given to the project. Teams are formed for each set of goals. These teams are staffed with at least one person from each site. These teams are responsible for developing a plan for how the goals are going to be achieved. As part of this plan they define site goals and measures. These teams typically meet face to face during the workshop and then interact periodically via teleconference as needed. Each team member is responsible for developing the plan for their site. They are not responsible for execution only for the plan development. This practice creates communication across sites, and may be useful if particular expertise is needed down the road.

6. Next steps

This activity and the associated report are a good first step at identifying and synthesizing the knowledge throughout the company that has been gained over more than 20 years of global software development within Siemens. It is recognized that all of the practices that have been identified to date as well as the practices that have not yet been documented work well in a given context, and may not be appropriate for all projects. While we have made an attempt to capture the context and identified prerequisites, we feel that this is not enough to determine the suitability of a given practice for a particular GSD project. Furthermore, in several cases the practices are addressing similar concerns, and some mechanism for choosing amongst them is needed.

In order to do this a project must:

- Recognize specific areas where collaborating as needed is likely to be risky
- Identify the suite of practices that are likely to address the specific areas of risk
- Understand the appropriate aspects of their context that are going to influence the cost or effectiveness of the available practices
- Have a means for selecting the most suitable practices and perhaps identifying contingency practices case they are needed
- Have some mechanism to monitor the effectiveness of these practices during the execution of the project

The Siemens Software Initiative and Carnegie Mellon are currently involved in an activity aimed at identifying and piloting an approach to accomplish the items listed above and more fully leverage the knowledge gained from the best practice collection activities. We hope to follow this report with additional reports describing the experienced gained as we progress with these activities.

7. References

[1] R. Sangwan, M. Bass, N. Mullick, D.J. Paulish, J. Kazmeier. *Global Software Development Handbook*, Auerbach Publications, 2006.

[2] J.D. Herbsleb, A. Mockus, T.A. Finholt, and R.E. Grinter. An empirical study of global software development: Distance and speed. In *Proceedings, International Conference on Software Engineering*, Toronto, Canada, May 15-18, 2001 pp. 81-90.

[3] R.E. Kraut, and L.A. Streeter, *Coordination in Software Development*. Communications of the ACM, 1995. **38**(3): p. 69-81.

[4] D.E. Damian, and D. Zowghi, *Requirements Engineering challenges in multi-site software development organizations.* Requirements Engineering Journal, 2003. **8**: p. 149-160.

[5] J.D. Herbsleb, D.J. Paulish, and M. Bass. Global software development at Siemens: experience from nine projects. In *Proceedings of the International Conference in Software Engineering (ICSE '05)*, May 15-21, 2005, St. Louis, Missouri, pp. 524-533.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software,* Addison-Wesley, 1995.

[7] M. Kircher, P. Jain, A. Corsaro, and D. Levine. Distributed eXtreme Programming. In *Proceedings of the Second International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP2001)*, pp. 66-71, Cagliari, Italy, May 2001.

[8] H. Holmstrom, E.O. Conchuir, P.J. Agerfalk, B. Fitzgerald, Global Software Development Challenges: A Cast Study

on Temporal, Geograhical and Socio-Cultural Distance. In *Processing of the 2006 IEEE International Conference on Global Software Engineering (ICGSE '06)*, Florianopolis, Brazil, 16-19 October 2006, pp. 3-11.

[9] S. Zhang, M. Tremaine, J. Fjermestad, A. Milewski, P. O'Sullivan, Delegation in Virtual Teams: the Moderating Effects of Team Maturity and Team Distance. In *Processing of the 2006 IEEE International Conference on Global Software Engineering (ICGSE '06)*, Florianopolis, Brazil, 16-19 October 2006, pp. 62-66.

[10] M. Heiss, S. Lasser. Collaboration Maturity and the Offshoring Cost Barrier: The Trade-Off between Flexibility in Team Composition and Cross-Site Communication Effort in Geographically Distributed Development Projects. In *Proceedings of the IEEE, International Professional Communication Conference (IPCC 2005),* Limerick, Ireland, 10-13 July 2005, Thread: Engineering Management (W10D).

[11] F. Lanubile, D. Damian, and H.L. Oppenheimer. Global software development: Technical, organizational, and social challenges. *SIGSOFT Software Engineering Notes*, 28(6):2–2, 2003.