SOFTWARE PROCESS IMPROVEMENT AND PRACTICE
Softw. Process Improve. Pract. 2003; **8**: 183–199 (DOI: 10.1002/spip.187)

Collaboration Practices in Global Inter-organizational Software Development Projects

Research Section

Maria Paasivaara and Casper Lassenius*,†
Helsinki University of Technology, Software Business and Engineering Institute, FIN-02015 HUT, Finland

Global interorganizational software development projects are becoming common, but their management and the creation of practices and processes to support collaboration seem to be harder than what the companies expect. In this article, we present successful collaboration practices collected in an interview study of eight globally distributed interorganizational software development projects.

On the basis of 34 semistructured interviews, we were able to identify several practices that the interviewees subjectively deemed successful. The identified collaboration practices include: milestone synchronization, frequent deliveries, and the establishment of peer-to-peer links. The need to plan for problem-solving communication was often neglected in the beginning of the project, despite its paramount importance. We identified several ways to ease related problems, such as having a dedicated person solve problems, using bulletin boards and e-mail lists or dedicated mailboxes. Successful projects had learned the value of two-way communication regarding project progress monitoring. Finally, practices helping in building and maintaining a working relationship included face-to-face meetings, distribution of organization charts, and having people travel to give all sites faces. Copyright © 2004 John Wiley & Sons, Ltd.

KEY WORDS: global software development; interorganizational software development; collaboration practices; communication

1. INTRODUCTION

Global interorganizational software development, including outsourcing, subcontracting and partnerships, is becoming increasingly common (Heeks *et al*. 2001). Advice for outsourcing and acquiring large systems or modules with well-defined requirements and/or interfaces can be found in literature (e.g. IEEE 1998). However, software projects developing genuinely novel products are often faced with uncertainty regarding, e.g. both requirements and implementation technologies, and subcontractors or partners need to be involved long before these uncertainties can be resolved. In such projects, the parties cannot receive clear requirement specifications at the beginning. Instead, close cooperation and communication between the parties is required during the whole project, as the project both builds a product and tries to understand what to build at the same time. In these kinds of projects, problems often arise since practices and processes needed for collaborating across distances and organizations

* Correspondence to: Casper Lassenius, Helsinki University of Technology, Software Business and Engineering Institute, P.O.B. 9210, FIN-02015 HUT, Finland
† E-mail: Casper.Lassenius@hut.fi

are neither well understood in theory nor typically established in practice.

In our experience, companies easily underestimate the need for specific collaboration practices and processes when running projects across distances. Often, companies seem to jump start global interorganizational projects without first planning how to work together with their partners. This can lead to quite problematic situations. Collaborating companies often have differing cultures, processes, and practices that may pose significant challenges to joint projects.

Current literature does not provide much help for managers planning globally distributed software development projects; only a few articles can be found presenting practices that are really used in case projects (e.g. Battin *et al*. 2001, Ebert and De Neve 2001, Heeks *et al*. 2001). In particular, interorganizational aspects are often missing. We believe that empirically collecting successful collaboration practices could provide useful information for managers who plan and execute global interorganizational software development projects. The objective of our study was to collect collaboration practices that really are used in interorganizational software development projects and that are perceived as useful. In this paper, we present a collection of successful practices we identified in eight globally distributed interorganizational software development projects. We focus on practices useful in interorganizational projects, but we think that most, if not all, of the practices could be useful in intraorganizational distributed projects as well.

The rest of this paper is structured in the following way: Section 2 briefly presents related literature. After that, in Section 3, we describe the research methodology and introduce the case companies and projects studied. In Section 4, we present the identified collaboration practices. Finally, we present a discussion of our results, as well as the conclusions and ideas for future work.

2. RELATED WORK

Global software development literature brings up many challenges related to collaboration practices and processes (e.g. Mockus and Herbsleb 2001). Also, some solutions and advice regarding 'best practices' are presented (e.g. Carmel 1999, Ebert and De Neve 2001).

Regarding collaboration processes, Battin *et al*. (2001) identify the challenge that collaborating sites even within one company can have differing development processes, a fact that can be very problematic when collaborating closely. Differing change-management processes between sites have also caused problems, especially in the integration phase, according to Herbsleb and Grinter (1999). One solution to these process compatibility problems could be to force all sites to use the same process. Another solution, which provides a faster way to start a project, is to let everybody use his/her own processes. However, then, it is necessary to clearly divide and specify the work (Battin *et al*. 2001). Synchronization of work (Herbsleb and Moitra 2001) and integrating work products (Battin *et al*. 2001) are other process related challenges. As a solution, Battin *et al*. (2001) suggest an incremental integration plan, which would be based on clusters and shared milestones to avoid 'big bang' integration. Incremental integration and frequent deliveries is a core practice in agile methodologies for collocated projects (Larman and Basili 2003), but its use in distributed development has not yet received much attention. Fowler (2004) and Simons (2002) recently reported their experiences in using agile methods in offshore development projects. According to Simons (2002), an iterative model seems to work well in these distributed projects, and it benefits the project also by providing increased visibility into project status.

Communication is in a central position in almost all collaboration practices and processes. However, communication is often mentioned as the biggest problem in distributed projects, since, e.g. geographical distance limits face-to-face communication, time zone differences prohibit synchronous communication, and language and cultural differences cause misunderstandings. Literature has proposed several low-level communication practices for globally distributed projects, e.g. the use of liaisons (Battin *et al*. 2001), cultural liaisons (Carmel and Agarwal 2001) or straddlers (Heeks *et al*. 2001) to share information and facilitate contacts; conference calls to resolve problems (Battin *et al*. 2001); the rotation of management to cope with cultural diversity (Ebert and De Neve 2001); and setting up a project home page summarizing project content and planning information to distribute information (Ebert and De Neve 2001).

Communication practices and patterns have also been studied in the new product development literature. Allen (1977) studied communication in geographically dispersed organizations and identified several communication patterns, e.g. the gatekeeper pattern. Gatekeepers are link persons that have several peer-to-peer links outside their own project or company. A gatekeeper disseminates information from outside to inside the project and translates it into terms relevant to the project. Gloor *et al*. (2003) have built a software tool to uncover and visualize communication patterns based on the basis of e-mail messages exchanged in a network. Their work has already revealed communication patterns and some central roles in innovation networks.

Software project management literature also discusses patterns. Coplien (1994, 1995) describes organizational and process patterns, which he thinks will help us, both to understand existing software development organizations and to build new ones. Ambler (1998) presents process patterns that are especially intended for object-oriented software development. He defines an organizational pattern as 'a pattern that describes a common management technique or a potential organizational structure' and a process pattern as 'a pattern that describes a proven, successful approach and/or series of actions for developing software'. Both Coplien and Ambler present patterns that are proven or at least expected to be successful. Some organizational patterns that Coplien (1994, 1995) presents describe communication-related problems and solutions. Coplien's patterns are carefully described: every pattern has a name, describes the problem, gives the context of the problem, explains the forces or trade-offs affecting it, gives a solution to the problems, and explains the resulting context and design rationale. Other collaboration practices or patterns, especially meant for distributed use, can be found, e.g. from the Dispersed Agile Software Development web site (cited 17.4.2003). These pages present patterns such as TravellingDevelopers and DailyConferenceCall. However, some of these patterns are 'protoPatterns', i.e. they may be just suggestions for useful practices that are not yet used as such in real life.

So far, only a few collaboration practices suitable for globally distributed software development projects are presented in literature. Moreover, the experiences of the use of these practices in real distributed projects are also scarce. We therefore believe that the collection and documentation of successful collaboration practices from globally distributed software development projects discussed in this paper is beneficial to advancing the knowledge in the field.

3. RESEARCH METHODOLOGY

The research presented in this paper is a multiple case study (Yin 1994). We chose six Finnish companies that developed software: Alpha, Beta, Gamma, Delta, Epsilon, and Zeta. Three of the companies developed software products, one developed customer-specific systems, and two developed embedded systems. We used purposeful sampling (Patton 1990) and selected companies that we knew used software subcontractors and that we expected to be experienced in interorganizational software development. All companies, except one, were large and well known in Finland. The companies were successful from a business point of view, on measures such as growth and profits.

We chose eight projects for closer study – one project each from five companies and three organizationally dispersed projects from one large company . We concentrated on projects involving subcontractors, and focused on the processes and practices used in the interface between the customer and the subcontractor(s) in parallel development situations. We purposefully selected projects that demanded constant collaboration and lots of communication between parties, e.g. due to a high degree of uncertainty, dependencies, and changing requirements. Therefore, in particular, projects developing new products and having the actual work of software development divided between participating companies were interesting for us.

Also, all the selected projects had a global distribution aspect, either inside or between the companies. All eight projects had sites or partners in two or more different countries. Four projects were distributed between continents, two of them between Europe and Asia and two between Europe and North America. In three projects, all sites were located in Europe. One customer-specific project had developers only in Finland, but the requirements came from customer sites located all over the world.

We performed 34 semistructured interviews, each lasting 2 to 3 hours. In each customer company

we interviewed, if possible, both a partnership manager responsible for software subcontracting and a process developer involved in developing the process used between customer and subcontractor. In some cases, the partnership manager and process developer was the same person. From each project, we interviewed a project manager and, if possible, also one or more team members and a representative from the subcontractor company. The cases and interviewees are presented in Table 1.

The number of interviewees differed between cases, depending, e.g. on the size of the project and how interesting the project was from the point of view of collaboration practices. In three cases, the number of project managers interviewed was more than one since some subproject managers were also interviewed. Subcontractors were interviewed only in half of the projects. In two cases, subcontractors were situated across such long distances that we were unable to travel, and in one case, the customer did not want us to interview the subcontractor. In one case, we found out that the subcontractors were not doing intensive concurrent development with the customer, making us believe that they would not provide much added value to the study.

In company Alpha, two of the projects, Alpha 1 and Alpha 2, were normal product development projects from different organizational units and project Alpha 3 was a bespoke system that was developed by a subcontractor. In company Zeta, we interviewed three persons from the subcontractor since the subcontractor had interesting internal distribution between Europe and Asia.

In the interviews, we asked the interviewees to completely describe the practices they used in their projects and the experiences they had gained. We did not have any ready-made categories of practices that we were looking for. Instead, we tried to encourage the interviewees to also tell about practices and experiences that we might not have thought of asking.

We tape recorded all interviews, transcribed them, and used ATLAS.ti for grouping and analyzing the data. ATLAS.ti is a software tool that provides support for grouping and analyzing qualitative textual data. In ATLAS.ti, we created categories that arose from the data and then collected quotations from data under these groups. The practices described here are based upon these categories describing related experiences and practices found in different projects.

All practices described in this paper have been successful in the projects they were used, according to the subjective opinions of the interviewees.

4. RESULTS

This section presents our findings on processes and practices used in global interorganizational software development projects. The most surprising result of our study was that the interviewed companies did not have any clear processes and practices that would have been commonly used in all of their interorganizational software development projects. The practices we encountered were mainly project specific and were created by trial and error at the project level. The practices identified and presented in this article might seem quite basic. However, in our experience, they are often not implemented in real-life interorganizational projects, even though, possibly, a lot of problems could be avoided by using them.

Table 1. Case project interviews

	Number of interviews per role							
Case projects	Partnership manager	Process developer	Project manager	Team member	Subcontractor	All	Industry	Geographical distribution and number of sites involved
Alpha 1	1	1	2	–	1	5	Telecom	Europe (3) and North America (1)
Alpha 2	2	1	1	–	–	4		Europe (4)
Alpha 3	–		–	–	2	2		Finland (several)
Beta	1	1	3	1	1	7	Bespoke SW	Europe (6)
Gamma	1	1	1	–	–	3	Security	Europe (2)
Delta	1		2	1	–	4	Telecom	Europe (2) & North America (1)
Epsilon	1		1	1	–	3	SW products to several industries	Europe (2) & Asia (2)
Zeta	1		1	1	3	6	Telecom	Europe (2) & Asia (1)

Table 2 presents a summary of the practices identified and the following subsections describe them in more detail. The first two practices, *synchronization of main milestones* and *frequent deliveries* are directly linked to the software development processes that the collaborating companies use. The third practice, *establishment of peer-to-peer links*, has an effect on the organizational structure and the roles in collaborating companies. The last three, *problem-solving practices, informing and monitoring practices*, and *relationship building practices*, are actually collections of several practices that are closely related and have similar goals. These three classes of practices support communication and collaboration within and between collaborating companies.

Table 2. Identified collaboration practices

Collaboration practice	Main finding
Synchronization of main milestones	It is not always necessary for closely collaborating companies to use the same software development process. Synchronizing the main process milestones between organizations seems to be enough.
Frequent deliveries	Using frequent deliveries and an iterative process seem to be very beneficial in distributed projects. It creates transparency, brings real checkpoints, and adds developer motivation.
Establishment of peer-to-peer links	Creating roles, assigning the roles to team members, and indicating which roles need to communicate with each other between companies. These links can be useful at several organizational levels; we identified three levels.
Problem-solving practices	Problem-solving communication was extremely important in the distributed projects studied. However, in the project-planning phase, this communication need was often neglected.
Informing and monitoring practices	Creating transparency about project progress seems to be important. Besides project managers, team members from all distributed sites also need progress information. For them, it can also be a motivating factor.
Relationship building practices	Building a good relationship between distributed teams requires at least some face-to-face contacts. Meeting at least someone face to face from collaborating distributed sites also helps.

4.1. Synchronization of Main Milestones

The first question when collaboration with partners or subcontractors really starts should be: 'What is the software development process the collaborating companies will use?' Quite often, the customer company determines the process to be used. It can simply give the subcontractor its process description saying: 'This is our process, use it'. This was the case in one of the projects we studied. However, it seems that it is not really necessary to enforce the same process. In our study, we found several successful projects in which both the customer and the subcontractor used their own development processes in their collaborative projects. Only the main phases and milestones were synchronized between the companies. In particular, in cases where the subcontractor already has a well-functioning process in place, this seems to work well. A few customer companies we interviewed said that when they notice that a subcontractor has a good and functioning own process in use, there is no reason to change that. Instead, letting the subcontractor use its own process makes starting the collaboration faster because the subcontractor does not have to learn a new process. However, the main milestones and project phases need to be synchronized. For example, companies can have the same name for main project phases, even though every company's internal processes inside these phases differ (Figure 1), as was explained by the person responsible for process development at company Alpha:

> 'When you understand that these two are the same [when comparing the process descriptions between the customer and the subcontractor], but it is stated differently, then we could agree that we jointly name this phase this. It gives a basis for having a common process which both parties have described in more detail.'

Our interviewees from projects Alpha 1 and Alpha 2 emphasized that when using an incremental process model, it is important that all participants synchronize the iteration cycles also, i.e. use the same length for iteration cycles and between deliveries, otherwise, problems will occur.

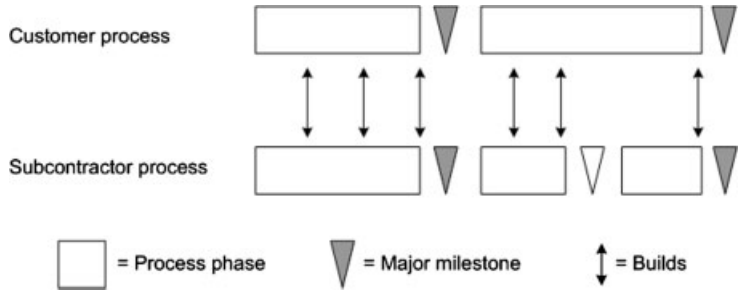

Figure 1. Synchronization of main milestones

4.2. Frequent Deliveries

The use of frequent deliveries from subcontractors and distributed sites seemed to be a very suitable practice for distributed use. Our interviewees said that when deliveries are integrated and tested right away, this gives a very good picture of how the project is progressing. Frequent deliveries normally mean that the whole project is using an iterative process model. Of course, frequent deliveries can be required from a subcontractor also when using a more traditional process model, as was done in project Beta. In such cases, the deliveries could consist, e.g. of draft versions of various documents in the early phases and code and test cases in the later ones.

The benefits of using frequent deliveries and integration are numerous. In our case companies, they prevented different sites and partners from doing long periods of independent development, which could lead to modules that are hard or impossible to integrate. Frequent delivery cycles also brought transparency of the work to all partners. When both the customer and the subcontractor used an iterative process with frequent deliveries, the subcontractor regularly delivered functioning code during the development phase, e.g. monthly or even weekly. The delivery was integrated into the system and tested. Frequent deliveries, followed by integration and testing ensured that the subcontractor was doing work that was compatible and that the requirements were understood correctly. Our interviewees had noted that frequent deliveries made it easier for the customer to monitor the real progress of the subcontractor's work. Integration and testing reports gave distributed developers instant feedback on their work, which they felt was very motivating. Moreover, when the customer saw that the subcontractor was doing good work, the customer's personnel started to trust and respect the subcontractor and its developers' know-how, which made further collaboration easier.

Half of the projects we studied, Alpha 1, Alpha 2, Beta, and Epsilon, used an iterative process model with frequent deliveries from subcontractors and distributed sites. These deliveries contained functioning code, which was integrated into a build and tested. The iteration and delivery cycles used varied between projects and also between project phases, e.g. in the beginning, they could be longer and later, in intensive phases, shorter. This was the case in projects Alpha 1 and Alpha 2. The subcontractor in project Alpha 1 described their delivery cycles:

> 'In the beginning we did three month increments. Then they told us that now we will do weekly deliveries.'

In the projects Alpha 1 and Alpha 2, during the most hectic development, weekly deliveries and builds were used. In other projects, builds were made more seldom. In project Alpha 2, two subcontractors used different iteration cycles, which caused problems when the changes did not come fast enough from the other subcontractor using longer iterations. In project Alpha 1, a similar problem occurred when one of the customer's offshore sites that delivered directly to the subcontractor had longer delivery cycles than all other sites and partners. The Finnish subcontractor in project Alpha 1 describes the situation:

> 'Company Alpha let their North American site deliver to us once every two months. We were required [to deliver] once a week. When we noticed that something was missing, we had to wait for two months. (. . .) We complained

about that, and finally they changed to this same weekly delivery cycle, but it took all too long [to do this change]. (. . .) This was one of the biggest mistakes made in this [project]. Our bug fixing average times were somewhere between two and three months [before the change]. (. . .) Our scheduling was ruined. (. . .) We got all the complaints because we were the subcontractors.'

4.3. Establishment of Peer-to-peer Links

Establishment of peer-to-peer links between collaborating companies seems to be crucial in distributed projects. Creating roles, assigning the roles to team members, and indicating which roles need to communicate with each other between companies was a successful practice described by interviewees from companies Alpha and Gamma. These companies had defined and had already taken into use part of the roles they had defined, but the work was still ongoing. Their initial experiences showed that roles made the interorganizational project structure more clear to all participating team members and helped them find the correct person to contact. These two companies had links between persons at three different organizational levels: at the management level between subcontracting managers or the like, at the project level between project managers, and at the team level between individual developers (Figure 2). Only projects Alpha 1, Alpha 2, Alpha 3, and Gamma had functioning links between roles at all these three levels. Other projects studied had different combinations of links, and all the projects had links at least between project managers from the collaborating companies.

4.3.1. Creation of Role Descriptions

Companies Alpha and Gamma had already written part of the role descriptions for their projects. The process developer in company Alpha described their goals in this work:

> '. . . we would like to unify these practices for working with subcontractors: what collaborating roles we have, what roles the counter party has, and how these roles exchange information. (. . .) The roles are described as a group of tasks in the process. (. . .) The role includes being responsible for a specific set of tasks.'

The role descriptions in Alpha and Gamma included, e.g. tasks to perform, decision-making rights, responsibilities, and identified other-party contacts. The idea was to use the same roles and descriptions in all projects. At the beginning of a new project, the roles needed are chosen and persons are assigned to those roles. When starting a project, it is easier to give team members roles rather than many separate tasks. Some of the roles may not include any interorganizational communication, while some may require lots of it. Our interviewees emphasized that it is important that the roles requiring interorganizational communication have matching counterparts, both at the customer's and both at the subcontractor's side. The role descriptions require both communication as well as allowing a person to communicate by making communication a task. Our interviewees thought that it was important to clearly define communication as a part of the job for certain roles since it is an important task that takes a lot of time. Actually, we noticed that in some other companies that

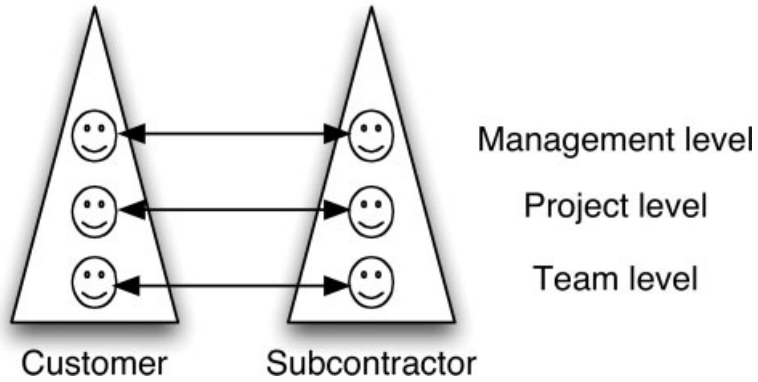

Figure 2. Peer-to-peer links between companies at several organizational levels

did not have these kind of role descriptions, a few interviewed link persons complained that communication was a task that required a considerable part of their time but their companies underestimated the time required for that and piled too much other work on them.

In addition to the customer company, the subcontractors also can make use of these role descriptions. For example, company Gamma encouraged its subcontractor to use the role descriptions it had created and documented on extranet pages.

4.3.2. Management-level Links

At the management level, collaborating companies needed peer-to-peer links between persons who could communicate about matters common to several projects (consecutive and parallel projects). According to our interviewees, communication was needed, e.g. about the customer's future project plans and the subcontractor's available resources, prices, infrastructure needs, problems in collaboration, future collaboration development activities, etc. Company Alpha had instantiated a practice of naming a subcontracting responsible from its organization for each of its subcontractors. Company Alpha also expected each subcontractor to name a corresponding person from their company to be a counter party in communication at this level. In company Gamma, which was a smaller company, a management-level peer-to-peer link was established between its product development manager and the subcontractor's corresponding manager. This link functioned particularly well in the form of weekly phone meetings between these persons.

4.3.3. Project-level Links

At the project level, the project managers in our case projects communicated on a daily basis. This peer-to-peer link seemed to function well in all projects studied.

4.3.4. Team-level Links

Development teams often have experts in collaborating companies who need to communicate with each other, e.g. persons responsible for related modules and software architects. Also developers, testers etc. may need to communicate across company borders. Encouragement of peer-to-peer communication between companies at this developer level brought contradictory opinions from our interviewees. Some commented that communication through these link persons was beneficial, but on the other hand, others were afraid that too much direct, uncontrolled communication between developers might lead to situations that are difficult to manage. The reason behind this fear was that because of direct contacts, project managers may not know what is going on and developers may agree directly on matters that affect costs, schedule or other parts of the system. Especially, when the subcontractors' personnel discuss with each other, it might cause some concern, as an interviewee from project Alpha 1 noted:

> 'Some of our subcontractors discuss at the developer level, but we try to minimize that by choosing the areas we subcontract suitably. (...) It is not to our benefit that, e.g., subcontractors create so good discussion links [to our other subcontractors] that they could centrally decide their price and start to pressure us.'

However, directing all intercompany communication through project managers may burden them or other key persons too much and it may restrict the information flow, as had happened in project Beta. Especially in smaller projects, developers from the subcontractor and the customer working on related tasks needed to communicate with each other. For example, in project Zeta, managers encouraged direct communication by introducing the developers to each other. These kinds of situations emerge when developers from collaborating companies are working with modules that have common interfaces. Another example is a situation in which a developer in the customer company is testing a module that a developer from the subcontractor has developed, like in project Zeta. These specific developers from the customer and the subcontractor, working on common interfaces or problems could be introduced and encouraged to communicate directly. The project manager of the subcontractor in project Alpha 1 stated:

> 'As a project manager my task is to create the right contacts between the developers, so that I don't slow down the work and communication. Instead, as soon as possible, the right persons will discuss directly, and I just follow that it is working and that there

> aren't problems. When I meet the customer's project manager and he introduces me to somebody [from the customer company], then I know that he works with that module and we have one person working with another [related] module, and then we arrange a meeting between these persons either through email, phone or face-to-face. Then they can solve the problem together.'

We noted that Internet chat seemed to be an efficient media for communication between developers. In two case projects, Epsilon and Zeta, developers, working in different companies and countries, communicated frequently using chat. The chat client made it easy for them to see who was present at another site. They mentioned several good properties of chat: it is inexpensive compared to the telephone, it is possible to have chat discussions open all the time, several developers can participate in the conversations, counter questions can be asked right away, foreign language speakers are easier to understand when the discussion is in written form, and from a written conversation, it is also easy to copy important paragraphs and, e.g. send them by e-mail to others. A system architect from project Epsilon used chat to discuss with developers from a foreign subsidiary and commented on its use:

> 'Chat is more practical than phone, especially with foreign partners. (. . .) It is already difficult to understand different pronunciations, not to mention the difficulties for me to even express what I want to say. (. . .) Writing emails takes a lot of time when you have to structure it and give background information. When I write some technical explanation, it takes time – it can take up to two hours to write one email when I search information and go back to code. (. . .) But, chat is more like talking. You do not have to structure things or think too carefully. Comments are very short. You can write about what you have on your mind. And the discussion just flows in the right direction.'

4.4. Problem-solving Practices

In all studied projects, problems and questions requiring timely solutions constantly appeared. Issues like this are particularly common in distributed software development projects (Carmel and Agarwal 2001). Few of the projects had planned for this kind of communication beforehand. In several of the studied projects, the lack of agreed-upon channels for problem-solving communication led to delays in solving the problem, which in turn caused delays at the project level. We found that the lack of suitable and defined communication channels caused project members to spend a lot of time just trying to find somebody to help them, wasting both time and energy. In addition, the barrier for the subcontractor's personnel to contact the customer was often high, despite the fact that the problems were both severe and in need of quick resolution. Problems that were brought up late were often both difficult and expensive to solve. Some of the case projects had taken some very useful problem-solving practices into use already at the beginning of the project, whereas other projects created new practices using trial and error during the project.

Next, we will describe three useful practices that we identified related to problem solving: *solution provider, bulletin boards and e-mail lists*, and *problem e-mail box*. Direct contact between developers, as discussed in Section 4.3.4 was also beneficial for problem solving.

4.4.1. Solution Provider

In two of our case projects, Beta and Epsilon, no formal problem-solving role existed at the beginning of the project, but after some time, developers learned, by asking around, that a specific individual, in both cases, a system architect, knew the answers to many questions. Subsequently, most of this person's time was spent on answering questions, using mainly e-mail, phone, or chat. A system architect from project Beta commented on his new 'role':

> 'It just gradually happened that I became ''a link person''. I have a lot of experience since I have been [here] so long. I have also been interested in the big picture of the project. But it is sometimes very hard. Because there are so many tasks, sometimes tens of tasks at the same time. And then I have my own work, the code I should develop. (. . .) My phone rings 40–50 times a day, but at the same time I should code thousands of lines. It is difficult to run from task to task, when you

> cannot concentrate. (. . .) Approximately half of my time goes to this kind of communication through phone or email, I'm like a help desk.'

In a way, this 'solution provider' liked his new role since he was a social person. However, he would have liked management to recognize his new position and understand that because of this new role, he did not have as much time to do normal development work as he used to.

These 'solution providers' either answered questions by themselves or found the answers from elsewhere in their organizations. Both projects that had 'solution providers' were very satisfied since it was easier to ask questions and the answers came quite quickly. However, it took quite a lot of these individuals' time to answer the questions. They reported that more than half of their working time was spent on answering questions and this was not taken into account when delegating other tasks to them. Therefore, in these two projects, this practice was successful, partly because these two individuals were very responsible and were eager to help as much as they could to make the project succeed.

4.4.2. Bulletin Boards and E-mail Lists

Another practice that was useful for problem solving was the use of discussion lists or bulletin boards. They were useful, e.g. for finding experts on some specific technology in a large project to answer a specific question. In two larger projects, Alpha 1 and Epsilon, bulletin boards focusing on specific technological topics were successfully used. In a smaller project, Gamma, project-wide mailing lists were used for the same purpose. According to our interviewees, in an e-mail or a bulletin board, questions asked need to be explained very carefully, otherwise, the readers do not understand the questions and they have to send several mails or messages asking clarifying questions before the question is understood correctly and can be answered.

4.4.3. Problem E-mailbox

In project Beta, an e-mailbox for problems was used in the early phases of the project. This project was distributed between 6 sites and had over 50 team members, most of whom did not know each other beforehand. In the beginning, it was extremely difficult to find the correct persons to ask questions. Therefore, this project decided to take a 'problem box' into use. The problem box was implemented as an e-mail address to which questions were sent. A person responsible for the problem box forwarded the questions to the persons she believed had the needed knowledge to answer. During the early phases of the project, this practice functioned very well. In the later phases of the project, the problem box was removed, partly because direct contacts were established and partly because it was regarded as an attempt by the customer's technical project manager to control communication a bit too much.

4.5. Informing and Monitoring Practices

Customers normally remembered to monitor how the subcontractor's work was progressing, even though it was difficult in the projects in which only time sheets were used. In interviews, we noticed that not only did project managers need progress information but the subcontractors' personnel and other distant sites also hoped to get information about the status of the project. The interviews showed that in addition to helping the personnel at distant sites to accomplish their tasks, this information also helped to motivate them, e.g. to adhere to the schedule since they understood why it was important. Subcontractors commented that in parallel development situations in which the work of various sites and partners was strongly interconnected, it was important to have status information flow not only from the subcontractor to the customer but also in the other direction.

Customers often forgot to inform the subcontractor about decisions and changes made or about new documents produced. The subcontractor in project Alpha 1 even commented that the customer gave them almost no documents unless they asked for it. The subcontractor further commented that it is very difficult to ask for documents that you don't know exist! In project Alpha 1, the lack of information concerned even the division of different tasks and modules between partners. The subcontractor in this project did not get enough information on how the responsibilities for creating certain modules were divided. The personnel at the subcontractor had some knowledge about the modules assigned to them and expected that probably someone else was doing the rest. This knowledge was partly wrong since there were modules that were needed between the parts the customer was developing and the

parts the subcontractor was developing, but nobody was taking care of them. The subcontractor's project manager was quite nervous when he found this out:

> 'We expected that they [the missing modules] would come from somewhere else, until we found the truth. (. . .) You never know whether some matter is forgotten or whether it is just that they [customer] are not telling us about it. You just don't want to ask about things for many years, when all you get is counter questions, such as 'How are YOU meeting the deadlines?' – meaning that it is none of our business. And then we get feedback that we should carry the responsibility for the whole project. (. . .) It is hard to be a subcontractor!'

Besides progress information, subcontractors also expected feedback on their work, e.g. about quality. They hoped to get comments also when they were doing something right, not only when things went wrong. This was also a motivating factor. We found that subcontractors rarely were given information on the current state of the project, and in some cases, they did not even receive feedback on their work. In many projects we studied, this communication need was totally disregarded.

We identified three useful practices for informing and monitoring: weekly meetings, progress reports, and traveling steering group.

4.5.1. Weekly Meetings

Physical meetings are difficult to arrange in distributed projects because of geographical distances. Time differences complicate matters further. In large projects, there are so many participants that it is difficult for all of them to be at the meeting at the same time and on the other hand, not all of them could be interested in every detail of the project, as some of our interviewees commented. On the other hand, all team members, both from the customer and the subcontractor, need information on project progress, what tasks to perform next, and on changes. They also need feedback on their work. In a small project, all distributed sites could have one common teleconference. In a larger project, smaller meetings can take place at every site or team, and team leaders or project managers can have common video-/teleconferences afterward, which was one of the practices used. In three of the case projects, Alpha 1, Beta, and Epsilon, different kinds of weekly meetings were used. For example, project Alpha 1 used site-specific meetings followed by teleconferences between team leaders and project managers. All projects using weekly meetings considered them to be useful. They all preferred teleconferences over videoconference since they felt that the picture in videoconferences was of low quality and, therefore, did not bring added value. Incompatible videoconferencing equipment decreased the value further.

4.5.2. Progress Reports

The customer needs to monitor the progress in the subcontractor's teams, but doing this is often very difficult since, e.g. the number of code lines or hours used in coding does not give very useful information. Lack of sufficient progress monitoring can cause negative surprises. Two case projects, Alpha 1 and Zeta, used quite detailed progress reports. In project Zeta, they were delivered every week, and in Alpha 1, they were delivered every month since in this project, weekly meetings partly compensated for reports. The subcontractor wrote weekly or monthly progress reports that included information, e.g. about tasks accomplished, open questions, problems, and future estimations about task- and problem-solving schedules. The customer gave feedback on every issue in the report. When the customer knew the situation in the subcontractor's teams, it could react quickly when the project was not going in the right direction and could also help in resolving problematic situations. Feedback received from the customer also gave the subcontractor confirmation that the correct tasks were being performed. Getting feedback also motivated team members.

4.5.3. Traveling Steering Group

Project Alpha 1 had a 'traveling steering group' that consisted of members from all sites and partners. Since all sites were involved, most important decisions could be made in the meetings and none of the partners were forgotten. Because the subcontractors could participate, their point of view and worries were also accounted for. In addition to decision-making, these meetings gave participants a good overview of the whole project, which they could convey to their own teams. The fact that the meeting location changed forced everyone in the steering group to visit all sites at least once and nobody had

to travel every time. This gave the team members at every site a possibility to meet representatives from all other sites. The frequency of meetings needed seems to depend on several factors. It seems that the level of uncertainty and the degree of requirement stability influence the meeting frequency the most.

4.6. Relationship building practices

When starting a distributed interorganizational project, it is quite common for many of the persons participating to not know each other beforehand, as was also the case in the projects we studied. In distributed software development projects, relationship building is especially important for collaboration to succeed. Relationship building actually takes place during all collaboration and communication between parties. However, in a distributed project, communication is more difficult than in a collocated project. It is normally easier to communicate with a person that you have met at least once. Therefore, when starting a distributed project with several partners and sites that have no common history, having all involved personnel meet face-to-face would be an optimal solution. Early face-to-face meetings also facilitate later electronic communication. For this kind of situations, literature suggests common kickoff meetings for the whole project (Carmel and Agarwal 2001). However, in practice, this is often not cost efficient. Actually, none of our case projects arranged a common kickoff meeting that would have included personnel from the subcontractors. This was mainly due to cost reasons since the projects were large and the distances were long. A couple of projects arranged kickoff meetings for their internal staff but did not invite subcontractors or people from all distributed sites.

Next, we describe two simple practices that we noticed were good for relationship building in cases where meeting everybody is not seen as a viable alternative: give faces and organization chart. Besides these two, a practice presented earlier, frequent deliveries, was also beneficial when building trust and a good relationship between distant colleagues since being able to produce functioning code in the early phase of the development builds trust between parties on the basis of their know-how.

4.6.1. Give Faces

We noticed in our study that distant sites and subcontractors were easily forgotten, and, e.g. their questions were not regarded as important and urgent to answer as the questions from colleagues nearby. It seems to be much easier to disregard questions or deliveries coming from unknown persons. A common kickoff meeting is certainly beneficial, if arranged. Instead of a kickoff, our case companies, using trial and error, arranged other possibilities to meet, often in response to problems.

Arranging for everybody to meet at least somebody from all other sites that they would be collaborating with seemed to be a well-working practice in our case projects. This gave the distant sites and companies 'faces', i.e. they were no longer unknown and easily disregarded partners when team members knew at least someone from each site. 'Giving faces' seemed to be the major benefit of various face-to-face arrangements that our case projects did.

These face-to-face meetings took different forms. Some examples were as follows: a system architect or some other key person traveled to the subcontractor's site to train them. The subcontractor's key persons were invited to the customer's site for training or for a short collocated working period. A practice presented earlier, traveling steering group, was also beneficial in giving faces to different sites, by arranging meetings at all sites.

In project Epsilon, testers were reluctant to test the code delivered by a subcontractor from a distant country. They preferred to test their mates' work first. The situation improved significantly after mutual visits. The project manager from Epsilon commented:

> 'We had difficulties to get our acceptance testing people to understand that we are in the same boat [with our subcontractors] and it is no use being enemies. (. . .) [The reason] might be that when these developers get a delivery and it is not functioning perfectly well, and they know that it is not made by their friends here, but by someone living in Turkey who they think is trying to do it as cheap as possible. (. . .) And that was the reason why it [testing] was delayed here, because it was not motivating. (. . .) [In this project] we learned a lot (. . .) about communication and how much

it actually helps to see those [subcontractor's] faces. It was difficult to believe it beforehand!'

In project Beta, a common cruising trip was organized in the middle of the project when the project was facing difficulties. Before this trip, many of the workers from different sites had never met each other. After the trip, communication and collaboration improved according to the interviewees. The subcontractor in project Beta commented on this trip:

'The trip was useful, because there we saw persons we had never met, but with whom we had been exchanging email every day. We also met the end-customer for this project for the first time.'

4.6.2. Organization Chart

An organization chart covering the whole interorganizational project was missing in most of our case projects. This was quite surprising since it is quite an easy thing to do, but it can help a lot. Such a chart makes it easier to find the correct persons to contact when questions emerge. In project Gamma, a simple web page with information about project personnel, including names, roles, photos, and contact information was regarded as very useful.

In project Alpha 1, the customer did not want to give the subcontractor its internal organization chart, which caused difficulties to the subcontractor. The subcontractor's project manager tried to solve the problem by creating his own organization chart of the customer by adding new persons and their contact information when he met them.

4.7. Summary of the Results

Table 3 summarizes the practices found in the case projects and presented in this article. In the table, we also summarize the benefits of using each of these practices and suggest the project phase in which each practice could be most useful.

5. DISCUSSION AND CONCLUSIONS

In this article, we presented collaboration practices and processes collected from globally distributed interorganizational software development projects. The most surprising result was the low state of practice regarding collaboration practices used in the case companies, all of them successful in their field. Actually, none of the case companies had clear practices that were commonly used in all their interorganizational projects. The practices we encountered were mainly project specific and created by project-level trial and error. For example, the case companies mainly did not agree upon communication practices in the beginning of their projects, a fact that later caused problems. The practices found and presented in this paper may seem very basic. However, we believe that in real-life projects, a lot of problems could be avoided by their systematic use.

5.1. Impact on Process

We believe that all the practices presented in this article have an impact on the software development process used. First of all, choosing a suitable process model for a project distributed between a customer and one or more software subcontractors is not trivial since all parties may have their own processes. We noticed that the customer organization often tended to enforce its process on subcontractors. However, this is not necessarily easy to implement. Starting to use a new process requires a lot of work and training, which of course requires additional time, and often is a constraining factor in these kinds of projects. Moreover, many subcontractors already have defined working software processes. When working with several different customers, these subcontractors might end up having to use different processes depending on the customer they are working with. Some of the customers in our case projects had noticed that when a subcontractor has a functioning process of its own, there is no need to change it. Instead, the customer and the subcontractor should sit down and compare their process descriptions and discuss the terms used and tasks done inside the process phases. This way, they can deepen their understanding of each other's processes and decide which main phases and milestones they should have in common and to what degree they can use their own processes. In general, when the main steps were synchronized, each company's internal tasks and processes could differ.

The second practice, frequent deliveries, also has a direct impact on the process used. The use of frequent deliveries seemed to be very suitable for distributed use since it provided transparency on

Table 3. Summary of the collaboration practices found

Collaboration practice	Description of the practice (and the number of projects in which identified)	Benefits of the practice	Process phase in which useful
Synchronization of main milestones	– If collaborating companies have good processes of their own, they do not have to change to a single common process but can instead focus on synchronizing the main milestones (4/8)	– Collaborating companies can use their own processes – Faster project start – Easier when collaborating with several companies	– Decided in the beginning of the collaboration – Benefits the whole process
Frequent deliveries	– Frequent deliveries of code, and integration and testing was very suitable for distributed use (4/8)	– Transparency of the progress – Early checks ensure that all parties have understood tasks correctly	– Especially useful during the software development phase
Establishment of peer-to-peer links	– Communication link persons between companies – Established at all organizational levels: subcontracting managers, project managers, and developers (functioning links at all levels: 4/8, functioning links at one or more levels: 8/8)	– Communication easier later on in the project – Communication increases between companies at all organizational levels	– Designed in the beginning of the collaboration – Benefits all project phases
Problem-solving practices	– A number of different practices: • solution provider (2/8) • bulletin boards/mailing lists (3/8) • problem e-mailbox (1/8)	– Encourages to ask questions – Problems get solved right away, not after the problems get really bad	– Designed in the beginning of the collaboration – Useful especially in the implementation phase
Informing and monitoring practices	– A number of different practices: • weekly meetings (3/8) • progress reports (2/8) • traveling steering group (1/8)	– Creates transparency of the progress in the projects – Prevents mistakes – Motivates all participants	– Designed in the beginning of the collaboration – Benefits all project phases, especially implementation, integration, and testing
Relationship building practices	– All face-to-face meetings help in building a good cooperative relationship – Also, developers need some face-to-face contacts to distant sites – Practices discussed: • give faces (5/8) • organization chart (3/8)	– Collaboration and initiating a contact is easier when you personally know at least someone from other collaborating sites	– Takes time especially in the beginning of the collaboration – Benefits the collaboration thereafter

work done at distributed sites and ensured that all sites were doing compatible work. Frequent deliveries can be used even when the software development process model used is the normal waterfall model since, even then, work in all phases can be done in small steps and delivered frequently, as was done in one of our case projects. An incremental process model with several iterations is of course an even more natural choice and seems to fit in distributed environments extremely well.

The third practice, establishment of peer-to-peer links has an effect on the organizational structure and roles in collaborating companies. By creating roles and by linking collaborating roles across companies, the interorganizational communication at different organizational levels especially will improve.

Finally, the last three groups of practices concentrated on communication: problem solving, informing and monitoring and relationship building practices. The most surprising finding was the huge need for problem-solving communication in these projects. This communication need was rarely thought about when planning the projects. Instead, solutions to arrange for this kind of communication emerged during the project. It seems that managers

should take this communication type into account and plan practices to arrange for efficient problem-solving communication already in the planning phase of a distributed project. Of course, part of this communication can be avoided, e.g. by careful planning, but it cannot be totally removed. The communication practices presented are just some examples of successful practices that companies can use in distributed interorganizational projects. Communication is an important and challenging aspect of distributed environment. Therefore, when planning processes for distributed projects, communication practices should be taken into account.

5.2. Managerial Implications

The ideal point in time for deciding which practices are needed and designing their use is in the beginning of the project, before the real work starts. Of course, practices can, and should, be revised, added, and also left out during the project, according to the situation. In this section, we discuss when, i.e. in what kind of projects, each of the practices identified in our study could be used.

Synchronization of the main milestones is a practice that companies that will be collaborating closely should adopt early, when designing their collaboration process. How much synchronization is needed depends, e.g. on how closely the companies will be collaborating and by how much their current processes differ.

'Frequent deliveries' seems to be a practice that suits distributed projects very well, and gives several benefits. Therefore, we think that it could well be used beneficially in most interorganizationally distributed software development projects. In particular, projects that suffer from uncertainties, e.g. in the form of changing requirements, will benefit from it. The length of the iteration cycles should be chosen carefully according to the project in question. The successful range for iteration length in our case projects was between one week and one month.

Creating peer-to-peer links at three levels between collaborating organizations is a practice that enhances communication. Besides naming the links, it is also important to encourage them to communicate. This practice is needed especially when the collaborating organizations are large and links do not form naturally.

We presented several problem-solving practices. It is important to remember that the need for problem solving exists in all projects and to design at least one practice when starting the project. Such practices are needed especially when partners do not know each other well beforehand, when the project is large, and/or when there is lots of uncertainty in the project. In particular, when responsiveness is important, we think that it is a good idea for the customer organization to define problem-solving roles, e.g. to name a person or two, whose primary responsibility is to make sure that both internal and external questions are answered quickly enough. When the subcontractor's developers know that there is an individual whose job it is to answer their questions and they know who he or she is, it is much easier to initiate contact. The use of 'solution providers' ensures that questions are answered quickly and that the work progresses in the right direction. Since answering questions and finding the solutions takes time, the solution provider should not have too many other duties.

Several informing practices were also presented. Even though some projects used several practices at the same time, we believe that in most projects any one practice would bring much benefit. Weekly meetings, face-to-face or through conference calls, is probably the best form when a project faces a lot of uncertainties. More stable projects could either have meetings more seldom or make do with only progress reports.

Relationship building practices are most important for projects with new partners and/or requiring quite constant collaboration. Making an organization chart of all persons contributing to a distributed project, and putting it, e.g. on the project extranet, where everybody in the project can easily find it could be a good idea in all projects. This chart could include names, roles, contact information, and, preferably, also photos and some personal information, e.g. about hobbies. 'Giving faces' is important, especially between persons and sites that are starting their collaboration.

5.3. Limitations

This paper presented a collection of successful processes and practices that were used in the eight case projects studied. The number of case projects was quite small and the practices presented were used only in part of the projects. Therefore, this study can only give an idea of what kind

of practices can be useful in globally distributed software development projects. Studying another set of projects would probably have resulted in a bit different collection. Therefore, we cannot say that these are the 'best practices', but only that they were useful according to the subjective opinion of the people we interviewed in those particular cases. When studying more projects, additional useful practices will certainly be identified.

The sample comprises only companies based in Finland. This can give some bias, e.g. due to cultural factors. However, all these Finnish companies also had offices in other countries and all studied projects, except one, were distributed in at least two countries. Therefore, international aspects were at least to some degree included in the results, though we did not explicitly focus on them.

We reported practices that were successful in the projects in which they were used. Of course, the success and usefulness of each practice depends on the specific project type in which it is used. For this study, we chose projects that demanded constant collaboration between the customer and the subcontractor and also between geographically dispersed sites since all parties were developing the same new software at the same time and everything could not be specified precisely beforehand. For other kinds of projects, other practices might prove to be more useful. Moreover, the projects studied were all interorganizational. Therefore, the practices identified are particularly suitable for these kinds of projects. However, we believe that most of the practices, such as frequent deliveries as well as all communication-related practices, could be used in distributed intraorganizational projects as well.

All the practices presented were subjectively determined as successful by the interviewees. We were not able to use any other measure of success, e.g. measuring the performance of a project would not have provided much added value since many other factors affect the performance, and all projects were one of a kind. Therefore, the best and easiest success measure seemed to be to rely on the subjective expert opinions of the interviewees.

5.4. Future Research

In the future, we plan to study more case projects and collect successful practices, concentrating especially on communication practices and the software development process used. First, our plan is to add some more similar case studies to find out more practices that can be useful but were not used in this very limited sample. Second, we plan to collect more detailed data about some of the most interesting and probably also important practices and their use in different situations, e.g. about frequent deliveries and problem-solving communication. Third, on the basis of the additional case studies, the classification of practices, which was presented in this paper, can be improved and then used as a base for quantitative studies. These quantitative studies could bring more knowledge, e.g. about the frequency of use of each of the practices and their effect on project success, measured, e.g. by software quality or by decrease in costs or by time to market.

The future research should also study what kind of projects and which project phases each of the practices should be chosen for. Moreover, we hope that this piece of research can help in developing communication tools that can better support the collaboration practices identified.

REFERENCES

Allen T. 1977. *Managing the Flow of Technology: Technology Transfer and the Dissemination of Technological Information within the R&D Organization*. MIT Press: placeCambridge, MA.

Ambler S. 1998. *Process Patterns, Building Large-Scale Systems Using Object Technology*. Cambridge University Press: Cambridge, UK.

Battin R, Crocker R, Kreidler J, Subramanian K. 2001. Leveraging resources in global software development. *IEEE Software* **18**(2): 70–77.

Carmel E. 1999. *Global Software Teams – Collaborating Across Borders and Time Zones*. Prentice Hall: Upper Saddle River, NJ.

Carmel E, Agarwal R. 2001. Tactical approaches for alleviating distance in global software development. *IEEE Software* **18**(2): 22–29.

Coplien J. 1994. A development process generative pattern language. Proceedings of PloP/94, Monticello, August 1994, 1–34.

Coplien J. 1995. A generative development-process pattern language. In *Pattern Languages of Program Design*, Coplien J, Schmidt D (eds.). Addison-Wesley: New York, pp 183–238.

Dispersed Agile Software Development and Dispersed eXtreme Programming web site. Cited 17.4.2003. http://www.fastnloose.com/cgi-bin/wiki.pl/dad.

Ebert C, De Neve P. 2001. Surviving global software development. *IEEE Software* **18**(2): 62–69.

Fowler M. 2004. *Using Agile Software Process with Offshore Development*. Cited 7.1.2004. http://martinfowler.com/articles/agileOffshore.html.

Gloor P, Laubacher R, Dynes S, Zhao Y. 2003. Visualization of communication patterns in collaborative innovation networks: analysis of some W3C working groups. Proceedings of ACM CKIM International Conference on Information and Knowledge Management, New Orleans, LA, November 3–8 2003, 56–60.

Heeks R, Krisna S, Nicholsen B, Sahay S. 2001. Synching or sinking: global software outsourcing relationships. *IEEE Software* **18**(2): 54–60.

Herbsleb J, Grinter R. 1999. Architectures, coordination, and distance: Conway's law and beyond. *IEEE Software* **16**(5): 63–70.

Herbsleb J, Moitra D. 2001. Global software development. *IEEE Software* **18**(2): 16–20.

IEEE. 1998. *IEEE Recommended Practice for Software Acquisition*. IEEE Std-1062. Institute of Electrical and Electronics Engineers, Inc.

Larman C, Basili V. 2003. Iterative and incremental development: a brief history. *Computer* **36**(6): 47–56.

Mockus A, Herbsleb J. 2001. Challenges of global software development. Proceedings of the Seventh International Software Metrics Symposium, London,UK, April 4–6 2001, 182–184.

Patton MQ. 1990. *Qualitative Research and Evaluation Methods*. Sage Publications: Newbury Park, CA.

Simons M. 2002. Internationally agile. InformIT, March 15.

Yin RK. 1994. *Case Study Research, Designs and Methods*. Sage Publications: Thousand Oaks, CA.