COPING WITH SOCIAL COMPLEXITY IN DISTRIBUTED SOFTWARE DEVELOPMENT PROJECTS

Jarkko Pyysäinen, Maria Paasivaara, Casper Lassenius
Helsinki University of Technology, Software Business and Engineering Institute
P.O. Box 9600
FIN-02015 HUT
Finland

ABSTRACT
The aim of this paper is to provide an insight into the nature and potential sources of social complexity in distributed software development projects based on case studies conducted in eight Finnish companies. Social complexity is here introduced as a term referring to the degree of the uncertainty that a social system encounters in its communicative operations with its environment. The paper provides a typology of distributed software development project types in the light of their characteristic levels of social complexity and outlines some successful practices in responding to this complexity in the most challenging types.

INTRODUCTION
Imagine a situation where trust in a distributed inter-organizational product development project is totally absent. What would this kind of disastrous situation look like? Nobody could predict the schedule or quality of the deliveries, the goals of the project would be constantly changing, and nobody could give you assistance with the uncertain questions of development work… Fortunately, such desperate situations are usually avoided, even though inter-organizational distributed projects (networks) have become increasingly popular in software development [4]. Such networks may include, e.g., several subcontractors or partners working concurrently with customers across distances and relying primarily on communication technologies instead of face-to-face meetings. Work across teams and companies is rather interdependent than independent and the need to orchestrate the work across the whole network is often great. This kind of coordination of work in networks creates new challenges and demands: coordination is simply not possible based on such traditional features as direct face-to-face feedback, common experiences, similarity of backgrounds and co-located decision making [5]. Instead, alternative ways for facilitating cooperation and communication must be utilized.

One way to conceptualise the effects of this kind of a situation for the participants of the network is to say that they must act in circumstances of increased social complexity. Here social complexity can be understood according to Luhmann [7,8,9], who defines it as the degree of the uncertainty that a social system encounters in its communicative operations with its environment. In other words, a social system of a customer who is doing concurrent and highly interdependent development of an innovative, novel product with a subcontractor, would manifest high degree of social complexity. In such a situation the changes in the environment are unpredictable but still highly relevant for the operation of the social system in question. In the field of organizational behavior a growing attention has been paid to the role of trust as a
mechanism for facilitating cooperation and reducing social complexity in complicated environments. A growing body of literature demonstrates the important benefits of trust for organizations and their members [2,6,11]. Even more importantly, trust is seen as a necessary element in facilitating the functioning of networked organizations [5,10,14] and global software outsourcing relationships [3,12].

While on the one hand trust building seems to be a promising mechanism for overcoming many difficulties related to distributed software development, it may on the other hand be precisely the distributed organizational contexts that constrain the development of trust between companies and teams [e.g. 4] and thereby hinder the coping with social complexity. Previous literature on software development has suggested contradictory solutions to the challenges created by the need to coordinate development work collaboratively: one line of reasoning emphasizes that much communication is needed to overcome the challenges of coordination, whereas the other line stresses that communication is merely a sign of inadequate planning and standardization of development tools and methods [3]. However, in the light of our approach these contradictory suggestions may be understood as reflecting adequate solutions in situations of varying social complexity.

The aim of this paper is to provide an insight of the nature and potential sources of social complexity in distributed software development projects based on case studies conducted in eight Finnish companies. The paper provides a typology of project types in the light of their characteristic levels of social complexity and outlines some successful practices in responding to this complexity in the most challenging types.

THEORETICAL BACKGROUND

Social complexity

The theoretical background is based on Luhmann’s [7,8,9] theory of trust as a mechanism for reducing social complexity. Luhmann’s theory deals with societal phenomena at a very abstract level, and for the purposes of this study our intention is to apply principles of his theory to the organizational level. Social complexity is conceived as the relation between a social system and its environment. The more there are possibilities to represent and communicate consequences of and alternatives for action within a social system, the higher the level of social complexity of the given social system. In this sense an organization whose members are able to reflect and communicate a wide variety of possible consequences of their actions in relation to their environment (e.g., other organizations), manifests a high level of social complexity. On the other hand, reduction of complexity can only be performed within the system, by means of structurally restricting its reactions to the environment. Accordingly, greater complexity of the social system allows better ability to react to changes in the environment. [7, pp. 9-13] In this sense social complexity refers to the predictability of events and the communicative acts used to reduce the uncertainty and risk within a social system.

In the case of distributed software projects the needs of social systems (e.g. customers) to establish communicative point-to-point correspondence with their environment (e.g. subcontractors) vary greatly depending on, e.g., the nature of uncertainty of the task in question. Social complexity seems to be a useful concept in understanding the meaning of these differences. The choice to follow Luhmann and conceptualise distributed projects as systems of communicative behaviors also allows us to consider central software development and project management issues (specifications, roles, project managing, development process) as essentially
communicative processes – and so as elements related to the level of social complexity.

**Familiarity, confidence and trust as mechanisms to respond to social complexity**

Luhmann [8,9] has presented a useful distinction of the basic modes of responding to social complexity. The conditions of familiarity, confidence and trust all represent qualitatively different modes of asserting expectations and of reducing social complexity. For example trusting behaviour allows more alternatives for action in the face of risk and uncertainty – in certain sense trust maintains the rich variety for action created by increasing complexity but at the same time reducing the complexity itself by providing flexible guidelines for action and evaluation. However, in order to be persistent trusting behaviour presupposes that the previous stages of familiarity and confidence do not lapse into disappointments. Should the expectations lapse into disappointments reduction of social complexity would happen in the form of distrust. *Familiarity* is the first necessary condition for one’s capability to form stable orientations towards the environment. Familiarity is also a necessary antecedent of the development of trust, because no stable expectations can be formed towards the strange, which remains mentally uncontrollable. Familiarity by itself may be a strong source of trust, but it provides only very narrow frames for action since in such cases trust tends to be equated with the familiar (e.g., friend, relative) and distrust with the unfamiliar (e.g., foreigner, competitor). The second condition is the stage of *confidence*, which depends in turn upon a certain amount of familiarity of the target. Confidence is based upon expectations of normal practices, standard operations and reliable norms. Confident expectations mean that no alternative ways of doing things are actively thought about; instead, a certain scenario is taken for granted. Finally, *trust* is the stage where open negotiation, active search for alternatives and reflection of the assumptions guiding one’s action become the primary mode of responding to uncertainty and risk. In this way trusting mode allows one to cope with higher degrees of social complexity. However, trusting mode of action is likely to survive only if the stages of familiarity and confidence are fulfilled and do not let the trusting parties down. [8,9, also 13]

In inter-organizational settings – as the distributed projects examined here – familiarity, confidence and trust can be seen as inseparable elements in any kind of persistent communication or interaction between parties. Trusting orientation is seen to be a central ingredient in, e.g., spontaneous offerings of help, creative collaborative work efforts and willingness to provide information [e.g., 6]. In inter-organizational situations trust may be best understood as a relationship between parties, not as a property of individuals or organizations. Trust prevails if each of the interacting parties acknowledges the right of the others to assess the competence and the intentions of their acts [14]. According to Mishra [11] the common conceptualisations of trust can be seen as including four distinctive components: *competence, reliability, openness,* and *concern*. Thus trust is established, if parties after an assessment are willing to be vulnerable to – or cooperate with – each other based on the belief that the other is competent, open, concerned and reliable [11]. These four components can also be interpreted as reflecting the central dimensions of confidence and trust described by Luhmann: confidence includes expectations of the competence and reliability of others, while trust implies that one expects others to be open and concerned.

This has direct relevance in networked software development environments, where various forms of communication are needed but the conditions for the
development of trust are not optimal. If sudden problems or uncertainties are encountered in a situation, where methods have been standardized to a large extent, the actor typically has to come up with creative solutions and cooperative efforts with the experts from other companies: this kind of situation would represent an example of increasing social complexity, which presupposes that the actor is able to trust the other party – otherwise social complexity may hinder the actor from finding solutions completely. If the other party then does not reciprocate trust or the actor is simply not familiar with anybody whom to contact, the actor probably decides to reduce intolerable social complexity by choosing to distrust and refrain from action completely: the other party might be accused of violating the cooperative relationship and the relationship might break up, while the problem might finally end up to some official instance. In this sense trust has concrete consequences for action in the face of social complexity in everyday project work.

METHODOLOGY AND EMPIRICAL DATA

The data was collected in an interview study that seeks to explore working practices and problems in networked, inter-organisational software development projects. The focus of the study was on networked projects that involve at least two companies: a customer and a subcontractor. Most networks still involved more than two organizational parties.

In each case the customer company was Finnish. Altogether eight customers, five subcontractors and nine distributed projects were studied. All customer companies except one were quite large and nationally well known. The data consists of transcribed thematic interviews of the project personnel and managers (N=46).

The examined networks can be classified according to the software development type in question:

- 4 distributed projects were developing software products
- 2 distributed projects were developing customer specific systems
- 3 distributed projects were developing embedded systems

The study follows the qualitative multiple case study approach [16]. According to Yin [16, pp. 1-3] case study approach is suitable when the aim is to understand complex social phenomena in their contemporary, real-life context and when the boundaries between the phenomenon and its context are not clearly evident. This corresponds to the aims of this study and to the characteristics of the phenomenon studied: First, social complexity is by definition a complex social and communicative phenomenon, which is hard to isolate from its context. Second, the compared software development networks represent examples of different kinds of communicative relationships between the social systems (e.g. customer) and their environment (e.g. subcontractor). Thus by comparison it is possible to distinguish the circumstances that generate social complexity without destroying the holistic relationship between the phenomenon and its context.

Besides the comparison between cases the qualitative analysis of the data follows a general grounded theory approach [15]. The aim of the this qualitative analysis is to construct a data driven interpretation of the role of social complexity in software development networks. However, the classification framework – or coding paradigm [15] – is derived from the central dimensions of the theories of social complexity and trust. In this sense, the concepts of social complexity and trust guide the classification of data, but the classification is still not determined by any existing theory. This is consonant with the general principles of grounded theory, even though strictly
formulated grounded theory emphasizes freedom from all theoretical preconceptions when approaching the data.

The analysis progressed through the phases of open coding, axial coding and selective coding [15], even though in practice the phases often overlapped with each other. In the open coding phase the data was classified according to a couple of themes that were central to the research problem and thoroughly discussed by all interviewees. These basic categories were very generally understood communication problems and communication practices in project work. In the axial coding phase the basic categories were differentiated to subcategories according to their common thematic dimensions. Further, applying a classification framework derived from the theoretical background helped to specify the relationships between these subcategories: the subcategories were further specified according to their corresponding trust–related components, and now these categories represented more concretely the sources and reduction mechanisms of social complexity in distributed projects. Finally, in the selective coding phase these categories were grouped to form common patterns of the sources of social complexity and of practices applied to cope with this complexity.

RESULTS

The comparative analysis of the distributed software development projects suggests that the projects can be divided into four types according to their characteristic levels of social complexity. These project types are named according to the types of subcontracting relationship, since the degree of social complexity in these distributed projects seemed to depend strongly on the characteristics of the subcontracting relationship in question. In this sense it can be stated that the nature of the subcontracting relationship influenced strongly the extent to which the companies had to accomplish mutual point-to-point adaptations – or in other words: to cope with social complexity in the distributed project.

![Diagram of project types](Image)

Figure 1. Classification of project types according to the degree of social complexity

**Body shopping**

Body shopping or personnel hiring represents a type of subcontracting where the social complexity does not actually increase as a result of the subcontracting. Here the
participants of the subcontractor are taken to the customer’s premises and assimilated to the social system of the customer. Consequently, the customer does not have to significantly develop its readiness to react to the environment. This was a major reason for the companies using body shopping to continue practising this type of subcontracting.

**Black box**

In the black box type the subcontractor is given a larger independent module with specified requirements. The specifications need to be very clear and should not include many critical dependencies to the development of other modules. Very few changes are expected. However, the supplier has to have good project management knowledge, which in turn decreases the customer’s needs to continuously monitor the progress of the work at the supplier. In this kind of a project the degree of cooperation and communication is expected to be low, which means that remarkable changes in social complexity are not expected in this type of subcontracting either.

**Independent subcontractor teams**

Independent subcontractor teams represent a type of subcontracting, where the subcontractor is involved in and given tasks during the whole project. Since the payment is typically arranged on hourly or piecework basis this type of relationship is not suitable for flexible change negotiations during the project. The subcontractor has allocated resources, which form quite permanent project teams, and the subcontractor’s own project manager typically does the allocation of tasks. However, the tasks given to the subcontractor tend to be quite small and clearly specified, not including critical interfaces to the modules being developed at other sites. However, in this type of subcontracting also the subcontractor is given some project management responsibility, because the deliveries and planning of new tasks are directly linked to the concurrent development work of the customer. This means that both customer and subcontractor must increase their readiness to keep in touch with and react to the changes in their environment.

**Transparent box**

In the transparent box type of subcontracting the subcontractor is typically involved in the project right from the beginning and given a larger responsibility. In contrast to the other types, there is a lot of uncertainty in the requirements and changes are expected. Therefore, a lot of cooperation and communication between the companies are needed during the project. Payment may be arranged by, e.g., various risk sharing solutions. The supplier has a project manager of its own and also internal project management responsibility. This subcontracting type is used when the requirements are uncertain, the customer might want to buy specific know-how and also give some of the project management responsibility away.

**Social complexity in the distributed projects: sources and ways to cope with it**

I introduce the identified sources of social complexity in distributed projects by presenting the results from two types of subcontracting relationships. This is done because the sources of social complexity in the black box type of subcontracting are very similar to those of independent subcontractor teams, but only more rare and less intensive. In ideal cases of the black box type any increases in the degree of social complexity are avoided completely. This is also the case with body shopping type, which did not illustrate clearly identifiable sources of social complexity in our data.
The sources of social complexity and the identified solutions to cope with it are presented in the following section according to the classification framework used in the analysis of the data. The classification framework is described in table 1.

<table>
<thead>
<tr>
<th>Sources of social complexity</th>
<th>Ways to respond to social complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task dimension:</td>
<td>Evaluative dimension:</td>
</tr>
<tr>
<td>Lack of openness and</td>
<td>Lack of openness and concern</td>
</tr>
<tr>
<td>Competence</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. Classification framework used in the analysis of the data

**Independent subcontractor teams**

**Task dimension:** In this type of subcontracting the companies in the network faced the problem of how to exchange sufficient background knowledge concerning the specifications and nature of the development work early on in the project. Further, a difficult issue for the companies was to figure out, what kind of information the other party would actually need. From the point of view of social complexity, a crucial dimension demanding mutual point-to-point adaptation from companies focused in this sense around the clarification of the task. In this type of subcontracting standardization concerning the clarification of the task (clear architectural specifications and process descriptions) played a major role. In this way the aim was to establish mutual adaptations between companies right in the beginning of the project and to achieve a common and determined understanding of the goals and means (technologies, development solutions) before the subcontractor started to develop the modules.

Thus on the task dimension social complexity was generated first of all by *competence* demands regarding the proper level of definition and specification of the modules and coding standards.

Another strongly related major source of social complexity was generated by the need to achieve sufficient *openness* between companies regarding the relevant task-related background knowledge. Typical of this type of projects were lacking dialogical contacts at the time when tasks and specifications were communicated – clarification of task and specifications seemed to be a key area demanding good communication and reaction readiness at both companies. When the customer was unfamiliar with the personnel and culture of the subcontractor, it was difficult for the customer companies to assess, whether the subcontractor had internalized the specifications properly or whether they were just reluctant to make clarifying questions. On the other hand subcontractors were often left alone with the vague specifications and insufficient background knowledge, because there was no stable media for the collaborative coordination of the open questions in the specifications. Additionally, customers had not included proper plans for supportive activities to their processes.

Unexpected needs to renegotiate the goals and means during the project increased social complexity significantly and were difficult to solve once the development work had already established its course.
Executive dimension: Since the assumption in this type of a distributed project was that the tasks, methods and expected quality of the deliveries should be rather well communicated or at least trained in the beginning of the project, developers were not prepared to negotiate with and assist each other extensively once the project execution had started. This was a major source of social complexity in this type of subcontracting, because the companies had not typically introduced any such role to the project whose responsibilities would include the clarification of documents, e.g. test feedback, or technical assistance between the companies. The need for this kind of a mediating role was recognized only after major problems in the project work were encountered. Problems related to technical assistance and explication of the causes of errors in code represent here increasing social complexity on dimensions of openness and competence.

Interestingly the dimension of concern was also closely intertwined with the issues of technical assistance and expected quality of work. It turned out that remote teams were often left without any proper guidance when they jumped into uncertainties or open questions, e.g., regarding expected development solutions or quality of work. In three of these five cases studied a prolongation of this kind of an uncertain situation resulted in mutual prejudices: suppliers felt that customer was not really concerned about them nor interested to facilitate their working, whereas people at the customer felt that the supplier is purposely doing low quality work or tries to play games with them.

Delivery dimension: Since the practise of this mode of subcontracting meant frequent deliveries, exchange of documentation and test reports during the project, there was a strong need to identify those types of inter-organizational communication and exchange that could be standardized and automated, e.g., in the form of configuration and version management. However, since the subcontractors did not have access to many important databases or documents, there was a need to create other reliable means to keep subcontractors up-to-date. This typically required some kind of reliable and secure interpersonal contacts between companies. On the delivery dimension experiences of satisfaction or dissatisfaction seemed to depend strongly on the reliability of the deliveries between teams; inconsistencies easily awakened feelings of risk and uncontrollability and even questioned the grounds of the whole distributed project work. The challenge was, e.g., to establish reliability on the quality of the build and on optimal delivery cycle. Lack of reliability in quality or in the delivery cycle caused frustrating reactions, since developers’ own work suffered and there was nothing that they themselves could do to solve these problems. In this sense lack of reliability represented here an example of increasing social complexity, which destroyed a basis from the developers’ confident expectations and forced them to opt either trust or distrust towards the delivering party in order to reduce the complexity. When the other party in these circumstances did not clearly reciprocate trust, the result was to decrease social complexity by choosing to distrust (“since that team is not always reliable nor actively facilitating our goals, we do not take their interests or goals into consideration in our work either”).

Evaluative dimension: The need for evaluation of and feedback on one’s work was a general and quite strong source of social complexity in this type of distributed projects. Especially remote teams and subcontractors felt a need for certain transparency and consistent checks upon the development work and progress of the whole project. However, transparency on the level of the whole project seemed to be very difficult to achieve. Consequently some remote teams felt that after all they were not aware, whether their quality of work had really been adequate for the project or
how their contributions finally affected the progress of the project. Problems of evaluation reflected lack of both openness and concern between companies. In some cases this kind of lacking overview resulted in decreasing motivation and difficulties to assess the relation between one’s own development tasks and the other interconnected tasks. Another important area of evaluation concerned the continuity of the supplier’s future work: it was important for both parties to be able to rely on the continuity of their relationship in the sense that customer has enough attractive projects for the supplier to keep to its resources steadily employed. In this type of subcontracting both parties aimed clearly to long-term relationship with each other, and open evaluation was critical when they were seeking optimal working practices to develop their relationship.

*Ways to respond to social complexity:* The practices required to cope with the increase in social complexity can be divided into three different response types: familiarity, confidence and trust, as presented in the second chapter. When evaluated from the perspective of the argument between standardization and active communication [e.g., 2], the results indicate that in this type of subcontracting the active communication should concentrate more on the beginning, and lay foundations for more standardized mode of development work later on in the project. Since social complexity was generated by insufficient openness on quite broad scale, on task-, executive and evaluative dimensions, the required practices had to somehow make up this deficit in openness. This was done in these projects by various practices that promoted familiarity. Such practices must naturally concentrate on the starting phase of the project, since subsequent needs to broaden the base of familiarity can only be performed on the basis of some initial familiarity. By arranging common kick-off meetings, e.g., associated with sub-projects and interdependent modules, those individuals who would be changing information or cooperating with each other, got to know each other already at the beginning of the project. Other successful practices were collocated reviews, training occasions and joint planning meetings, where great amounts of background information and tacit knowledge could be exchanged. Also the establishment and updating of an organizational chart for all the members to see, e.g., on the project web pages proved to be an important ingredient of familiarity, even though it remained absent in many cases.

Generally after the starting phase the development work should take quite a standardized mode, since excessive communication tends to be a sign of inadequate specifications and vague processes or process descriptions. However, a more standardized way of working seems to be possible only after familiarity and certain trust are already established in the earlier phases. In this sense it seems, that the development work in the independent subcontractor teams –type is ideally based on confidence and confident expectations – trusting orientation may not be necessary in a broader scale, since many developers are not expected to actively call into question the nature of various development solutions nor actively plan or reflect interconnected interface solutions with remote teams. Instead, trusting orientation seems to be necessary between the central contact persons who are expected largely to control the information exchange within and between teams. Here the active roles of project managers and certain technical contact persons were accentuated: development work fared well in cases where the project manager of the customer and project manager of the subcontractor had developed confidential and active relationships and were eagerly negotiating about the project status with each other. These managers could in turn direct the information exchange to specific areas and find the right participants for those occasions. Similarly the role of a technical contact person seemed to be
critical for these projects, because one expert could coordinate the open questions and problems effectively and release other members from information and communication overloads. In this sense the strong roles of technical contact persons and project managers functioned to decrease excessive social complexity by establishing confidence. Trust was still an important ingredient in interpersonal relationships, especially between the most active communicators, and that might have been the reason why fluent development work seemed to flourish best in more mature customer-subcontractor–relationships.

**Transparent box**

*Task dimension:* In this transparent box –type of subcontracting the relationship between the customer and the subcontractor was closer and communication between them was also more intensive than in the previous types. However, extensive communication and adaptation between companies might not be needed as early as the starting phase of the project, since the subcontractor is also typically given own planning-responsibility in the beginning. The most critical needs for point-to-point–adaptations were encountered only in later phases. The expected high level of subcontractor involvement and innovative input means that the subcontractor’s personnel should be willing to take initiative in this kind of project work. In our cases, this was not so much of a problem, because the subcontractor was typically actively involved in the project right from the specification phase and had thus acquired a deep understanding of the system to be developed before starting its work. On the task dimension one of the major goals seemed indeed to be that the subcontractor acquires a deep understanding of the system to be developed and on the basis of this understanding starts to plan specifications for the modules. In other words, now the required *openness* and *competence* did not target to the exchange of already existing specifications but to the internalisation of the principles of a larger technical system and creation of the specifications. In contrast to the independent subcontractor teams type, here the subcontractor was a mere receiver of information but also an active creator and sender, a fact that seemed to make the communication between companies somewhat more balanced and easier.

*Executive dimension:* Vague specifications and frequent changes – even sudden needs to cancel the development of certain features – implied that the teams in the network could not adhere to rigid norms and rules. Instead the development teams had to show readiness to continuously evaluate the future directions of development work and collaboratively negotiate about the emerging dependencies to other modules. Characteristic for the transparent box type was that also the development teams of the subcontractor had quite much power to decide on the specific development solutions in their modules; the subcontractors had more innovative roles and more autonomy regarding their development tasks than in other types of projects. It seemed that in this type of project the subcontractor was more committed to the goals of the whole project and was more eager to take broader responsibility of the success of the project. Of course, in this type of project, it was also possible to arrange the payment based on risk sharing, which affected the subcontractor’s role in the project. Since the specifications of the modules were sharpening during the project and the changes in other modules had to be taken into consideration also in one’s own development work, developers had to be aware of the progress in the whole project. This was the most crucial aspect of the *openness* dimension and a reason why open communication between developers was strongly supported. If openness was lacking, doubts about the *competence* of the other party started to rise. There simply had to be good readiness to
solve sudden interface-problems with remote teams: e-mails, mailing lists, phone, chat or collocated visits were used depending on the nature of the problem in question. Here the importance of concern was also accentuated: teams could not simply afford to ignore each others’ needs or requests, because all work in the network was so closely interconnected. Because of the high level of collaborative work, the aim was also to integrate or standardize the development tools to a large extent.

*Delivery dimension:* In the transparent box project type, synchronization of the deliveries (e.g. software builds, test reports) seemed to be highly critical for the functioning of the whole network. Since the project work included much uncertainty and innovative but risky technological solutions, the project seemed to require a frequent and reliable delivery rhythm. This was a way to achieve better reaction capabilities across whole network; loose or unreliable deliveries tended to cause immediate problems in the development work and frustrate the working and visibility of many developers. A *reliable* delivery cycle functioned as the pulse of the whole project and once it jumped into obstacles or delays, the whole development work tended to lose predictability and controllability. In this sense the delivery dimension was clearly an area, where the companies accomplished rapid mutual adaptations on the basis of each other’s communicative messages (the deliveries). Another characteristic point was that since the subcontractor was given modules, which included critical interfaces to other modules and were important in the functioning of the whole system, the integration phase was typically more complex and demanding than in the other types of projects. Especially in the integration phase both customer and subcontractor needed to keep their resources on alert, since emerging problems might have been very difficult to solve without help from the few experts on that particular area.

*Evaluative dimension: Openness* on the evaluative dimension was a problematic issue in a different manner compared to the independent subcontractor teams type, since due to the highly collaborative nature of the development work teams were to some extent aware how their contributions had succeeded and affected the progress and attainment of project goals. However, openness in evaluation was problematic in another manner: in cases where the network was fragmented and consisted of several parties, it was difficult to evaluate what kind of information had already been exchanged between given parties, unless everything was documented – and that was often not the case. It was difficult to find out who had already been informed on a given matter and who actually might need the information. The amount of exchanged information was so great in this kind of a project that it was difficult to evaluate who might need the piece of information. Still the parties depended heavily in their working on the concern and willingness of others to inform them in critical issues. Unless the exchange of information was automated to a large extent, information blocks were easily created and teams started to doubt each other’s willingness to cooperate. In this sense the responses to social complexity tended to slide into distrust, which in turn made the whole project work much more difficult.

*Ways to respond to social complexity:* The results indicate that in this type of subcontracting the focus is strongly on active communication and standardization of development work should merely support these active communication efforts. Since uncertain specifications and frequent changes are characteristic of this project type, companies need to adapt and react to each other’s development work accurately during the whole project. Further, these mutual reactions must be accomplished among a very large group of people, including project managers, system architects, developers and testers, and consequently these people should get familiar with each
other early in the project. The presupposed base of familiarity has to be here both larger and deeper than in the independent subcontractor teams –type. Communication suffers badly, unless the members do not have quite detailed knowledge of each other’s backgrounds and roles when they face needs to plan common technical interface solutions or solve sudden interface problems collaboratively. Here a helpful practice was to arrange collocated planning or problem solving occasions between those members from different sites whose tasks contained many interdependencies. This was an effective way to establish familiarity and spread the tacit knowledge that would otherwise remain unrecognized.

A trusting orientation allows most flexibility in response to social complexity, as was described in the second chapter. Development work in the transparent box –type generates such high degrees of social complexity that without a general and stable trusting orientation the development work runs into difficult problems, most notable being the decreasing readiness to communicate and cooperate. A stable trusting orientation was not likely to survive alone in these projects; it presupposed also sufficient familiarity among the members of the network and enough confidence, in the sense of standard development process routines. In the transparent box type maybe the single most decisive practice was the frequent and reliable delivery cycle, which decreased the members’ difficult load in the midst of overwhelming uncertainty and unpredictability. In this sense it was precisely the frequent and reliable delivery process that provided needed confidence in the development work and functioned in turn as an incentive to bestow trust in order to overcome other unpredictable events. In occasions where even delivery cycle generated unpredictable consequences, the degree of social complexity began to be intolerable for the developers to handle in their work.

CONCLUSION

On the basis of the empirical findings it seems that different subcontracting types create different degrees of social complexity. In other words, companies in different distributed project types face different needs to adapt and react to changes happening in other companies of the network. Different degrees of social complexity may in turn demand different complexity-reducing responses. In cases of moderate social complexity the role of confidence is accentuated, whereas in cases of high social complexity the role of trust in accentuated. In the independent subcontractor teams –type the promising practices in responding to social complexity seem to focus on the establishment of confidence, especially by means of frequent and open project manager contacts and assistance provided by a salient technical contact person. These practices tend to strengthen the confident orientation among project members and help them to concentrate on their essential tasks. In the transparent box –type the successful practices in responding to social complexity seem to focus on the establishment of general trusting orientation among project members. Here the required flexible communication and openness are not likely to survive unless all three modes of familiarity, confidence and trust are properly established and supported in the network.

By concentrating properly on the practices that support the development of antecedent conditions of trust – familiarity and confidence – in the beginning of projects, we believe that companies can develop the ability to better respond to different uncertainties in their project work. However, this paper provides only initial outlines of some successful practices that may be useful in responding to social complexity. In the future our aim is to analyze the nature of social complexity in
distributed software development projects more thoroughly, e.g., by studying the
typical sources of problems in different kinds of software development project types.
We believe that when the nature of social complexity in different project types is
better understood, it also becomes possible to formulate adequate supporting practices
for different kinds of networks and projects.

REFERENCES


81-90.


Vol. 34, No. 4, 1992, pp. 53-72.


42, No. 2, 1999, pp. 80-86.


[14] van der Smagt, T., “Enhancing virtual teams: social relations vs. communication technology”, Industrial
