Developing Internet-based integrated architecture for managing globally distributed software development projects

Julia Kotlarsky
Department of Decision & Information Sciences, Erasmus University
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
Email: jkotlarsky@fbk.eur.nl

Abstract

Given the increasing importance of globally distributed software development (GDSD) over the last decade, it is surprising that empirical research in this area is still in the very early stage. The few existing suggest that traditional *coordination* and *control mechanisms* can be effective for these projects only with support from appropriate information technology. However, at present, little is known about the success of current Information and Communication Technology (ICT) support in the context of GDSD projects. Therefore, the main question this research addresses is *what ICT-based support is appropriate for globally distributed software development projects?* The objectives of this research are to elicit and develop the functional requirements for ICT support for GDSD projects, to analyze the gap between existing tools and these requirements, and to develop an Internet-based integrated architecture of tools that would fill these gaps.

1. Motivation and Significance of the Topic

Historically the demand for software services has outpaced supply. As we enter the era of e-business, companies are increasingly adopting complex software systems to support their internal and external processes. Meanwhile, we are also witnessing an exponential increase in the use of embedded software systems. Appliances such as mobile phones, organizers, cars etc. are beginning to be equipped with sophisticated software that communicates over the web. The demand for software and consequently software developers is exploding in all parts of the world.

This imbalance between demand and supply is further exacerbated by the high levels of skill and training required for building software. Software engineering organizations have always had trouble meeting the growing demand for high quality software. Although numerous improvements have been introduced to software engineering practices, Brooks' [1] claim that "building software will always be hard" is now generally accepted. Brooks listed four unique characteristics of software that make software development more difficult than other system-engineering disciplines. Software systems are *complex*, *unvisualizable*, and are constantly subject to *change*. Furthermore, they are expected to *conform* to the needs of the continuously changing environment in which they operate. These "inherent" challenges make software engineering a complex discipline. Hence, skilled software engineers and experienced software project managers are scarce and relatively expensive in most regions of the world. Consequently, large software development projects are regularly delayed and often show huge budget overruns [2].

In order to build quality software faster and cheaper companies in industrialized countries are turning to GDSD projects. A number of economical and technical trends are likely to further accelerate the growth of distributed software development. Economical trends include globalization of the industry in general. Multi-national companies often require software systems to be developed for geographically dispersed locations. Moving parts of the development process to emerging countries (such as India and Israel that are known to have large pools of highly trained software engineers at relatively low-costs) can decrease development cost [3,4]. Another perceived advantage of global distribution is the reduction in project life-cycle times by using time zone differences to organize "follow-the-sun" (or "round-the-clock") development [3-6].

On the technological side, ongoing innovations in ICT, by eliminating the perception of distance, create new possibilities to cooperate in software development projects in a distributed mode. Moreover, software industry has recently started to adopt a

more modular component-based architecture that further facilitates distributed development of software products. Traditionally, component-based architectures have been successfully used in industries such as aircraft, automotive, electronics, computers, for setting-up globally distributed design and production. Within the software industry, component-based development by reducing interdependencies between components, holds the promise that software components may be developed largely independently at dispersed sites.

Given the increasing importance of GDSD, it is surprising that only a limited number of empirical studies on distributed software development currently exist. The few existing studies [3,4,7] report numerous problems in distributed projects. The time, governance, infrastructure, and culture gaps, associated with the geographical dispersion of work, make it more difficult to manage inter-site work dependencies and to coordinate and control the distributed work. Furthermore, traditional *coordination* and *control mechanisms* are less effective in GDSD projects. Carmel [3] and Van Fenema [7] suggest that these traditional mechanisms will be effective for dispersed projects only with appropriate technology support. However little is known of the success of current ICT support within GDSD projects. Hence, ***our main research question is what ICT-based support is appropriate for globally distributed software development projects?***

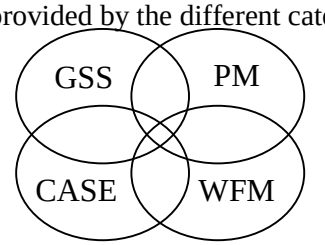
The objectives of this research therefore are to elicit and develop the functional requirements for ICT support for GDSD projects, to analyze the gap between existing tools and these requirements, and to develop and test an Internet-based integrated architecture of tools that would fill these gaps.

2. Research Approach

2.1 Research Questions

Our main research question is what ICT-based support is appropriate for globally distributed software development projects?

Our proposition is that despite a large number of tools currently available at the software market, they provide only partial support for GDSD projects. Furthermore, those tools are not designed to work together. Our starting premise is that the current generation of ICT-based tools available for supporting GDSD can be categorized into four functional groups (figure 1). As the figure 1 shows, there is usually some overlap in the functionality provided by the different categories of tools (tools are explained in section 3.2.2):



Where:

GSS – Group Support Systems
PM – Project Management tools
CASE – Computer-Aided Software Engineering tools
WFM – Workflow Management tools

Figure 1: Functionality provided by tools to be included into GDSD architecture.

To address the main research question we have developed a number of sub-questions to assist in identifying the requirements for tools to support GDSD:

Q1 *Based upon current state of theory, what requirements for tools to support GDSD projects can be identified?*

Q2 *What functionality is offered by tools currently available in the market?*

Q3.1 *What features of these tools are used in practice in GDSD projects?*

Q3.2 *What are the requirements for an ICT-based architecture to support GDSD projects that can be identified from such projects in the field?*

Q4.1 *What is the gap between the field-based requirements (identified in Q3.2), theory-based requirements (Q1) and the functionality offered by existing tools (Q2)?*

Q4.2 *Based upon the analysis in Q4.1, what requirements for an ICT-based architecture to support GDSD projects can be identified?*

Q5.1 *What is an appropriate ICT-based architecture to support GDSD projects?*

Q5.2 *How can this architecture be effectively used in GDSD projects?*

To answer these questions the research has been divided into several phases. Results obtained at each phase will be used as inputs for the next phase to define a scope and further develop questions to be researched in a next step. There has been a strong tradition of using a phased approach in systems development work, therefore we also follow this approach.

2.2 Research Model and Research Methodology

Figure 2 describes the structure of the research and the phases and steps to be undertaken at each phase. Furthermore, the model explains the research method to be used at each step (in white rectangles). Sources of information for each phase are mentioned in the upper (grey) rectangles.

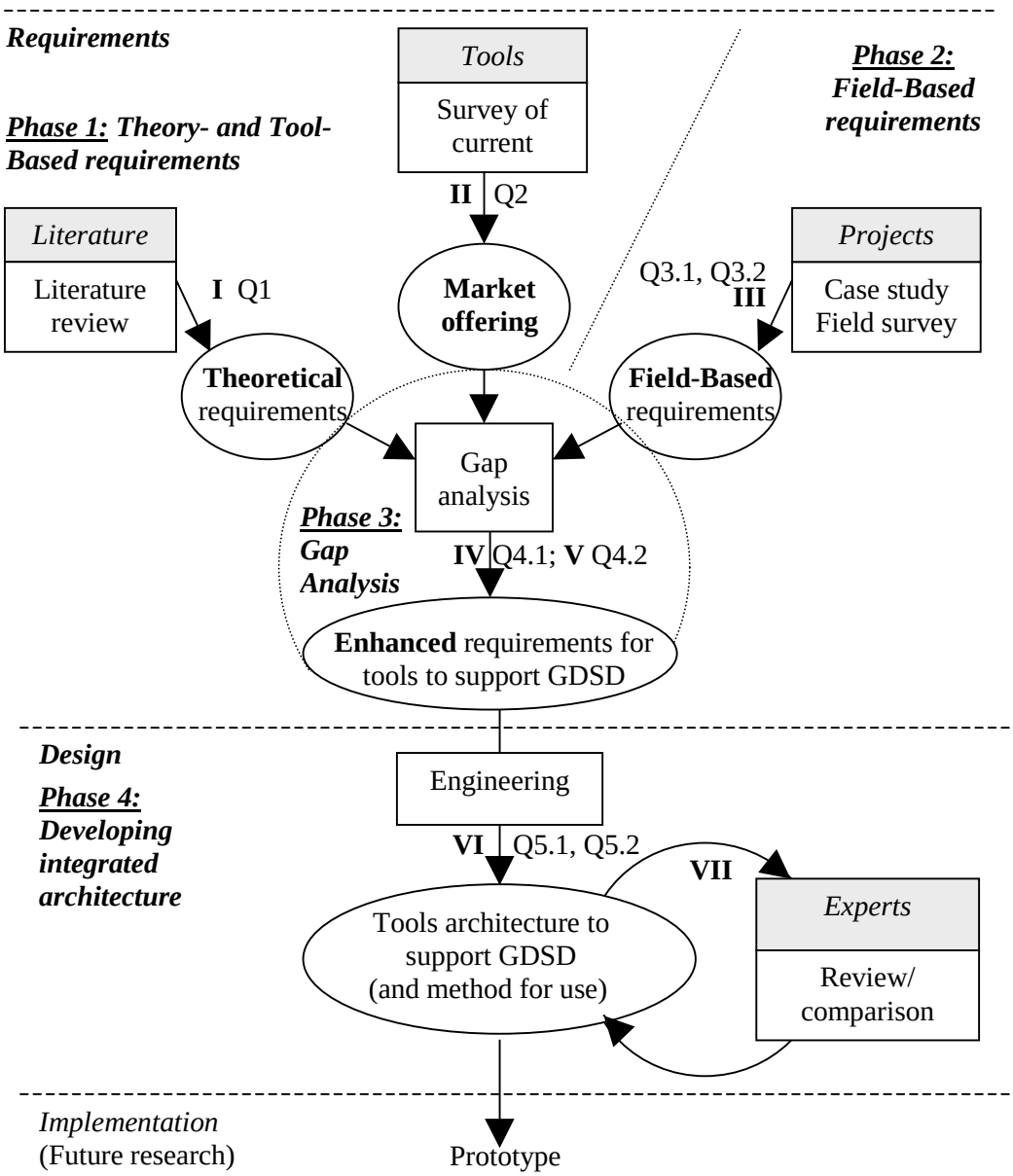


Figure 2: Research model

In *phase 1*, basing on existing literature, we will develop *theoretical functional requirements* for tools to support GDSD and functionality offered by existing tools (*market offering*). In *phase 2* we will look from a GDSD projects' perspective at the features of

tools currently used in real projects and what functionality project members would like to have in addition (*field-based requirements*). In *phase 3* we will conduct a *gap analysis* to see if there is a gap between theory-based requirement, field-based requirements, and market offering. Based on the gap analysis we will integrate the three sets of requirements into *enhanced functional requirements for tools support*. In *phase 4* we will develop a high-level design of the tools architecture and test the architecture. In future research, possibly, some prototype tools will be developed using the architecture.

As figure 2 shows, each phase addresses a number of research sub-questions (Q1-Q5.2), uses a variety of research methods and consists of number of steps (I-VII).

Phase 1

The objectives of this phase are to develop: (1) theoretical requirements for ICT-based tools to support GDSD and (2) functionality offered by existing tools.

The following table explains research questions, steps and methods of phase 1.

Research questions	**Methods**	**Steps to be undertaken**
Q1 What requirements for tools to support GDSD projects can be identified based upon current theory?	Literature study (academic, trade and professional)	**I** Development of *theory-based requirements* needed to support GDSD projects.
Q2 What functionality is offered by tools currently available in the market?	1) Review of tools' specifications and actual tools 2) Survey of tools' vendors and distributors	**II** Evaluation of existing tools (collecting and summarizing functionality provided by these tools into *market offer*). Evaluation framefork to be developed.

Table 1: Description of phase 1

In step II we will compare several tools (e.g. 3-4 market leaders and 1-2 new entrants) from each of our four target groups (see figure 1).

Phase 2

The purpose of this phase is to develop field-based requirements for the ICT-based architecture.

Existing studies (e.g. [8]) have shown that even if a tool can support a variety of requirements, not all the functionality of the tool is actually used in real projects. Usually only the major functionality (in which the tool is perceived to be strong) is used. For example, CASE tools have been reported as used mainly in the analysis and design phases [9], while most of other CASE features are not used. Kemerer [10] found that organizations were not using 70-90% of the CASE tools package purchased. Similarly, while CASE tools have features for project management and communications, they are reported as being minimally used for coordination tasks: "…organizations are leaning toward dedicated, user-friendly tools, such as Microsoft Project for resource management and Lotus Notes for communication" [11]. Furthermore, even if tools provide the required functionality, in reality "information technology will not always be used in ways envisioned by designers or intended by implementors" [8]. This leads us to think that *field-based requirements* for the architecture could be somewhat different than those based on the theory and existing tools. Therefore it is our hope that by analysing the gaps between the three, and integrating them we will have a more complete set of requirements.

The following table presents research questions, steps and methods of phase 2.

Research questions	Methods	Steps to be undertaken
Q3.1 What features of existing tools are used in practice in GDSD projects? **Q3.2** What requirements for an ICT-based architecture to support GDSD projects can be identified from such projects in the field?	Case studies (participant observation, interviews) Field survey using interviews and questionnaires	Case study of two-three GDSD projects and field survey of project managers and software developers in order to: **III.a** Study what functionality of tools available for project members is *used* in practice and what is *not used*(and why not used, e.g. because it is not required or some other tool is used to fulfill this functionality) **III.b** Study what functionality is identified by the project members as missing in tools they currently have (*desired functionality*). **III.c** Develop *field-based requirements* based upon the functionality that *is used* and *desired functionality*.

Table 2: Description of phase 2

Phase 3

The purpose of this phase is to conduct gap analysis of field-based requirements, functionality offered by existing tools and theoretical requirements.

The following table explains research question, step and method of phase 3.

Research questions	Methods	Steps to be undertaken
Q4.1 What is the gap between the field-based requirements, the functionality offered by existing tools, and the theoretical requirements?	Subjective/ Argumentative	**IV** Three-way gap analysis –of field-based requirements, the market offering and theory-based requirements.
Q4.2 What are requirements for an ICT-based architecture to support GDSD projects?		**V** Using a combination of results from previous steps, developing a set of *ehchanced integrated requirements* for tools to support GDSD.

Table 3: Description of phase 3

Phase 4

The objectives of this phase are to develop: (1) integrated architecture for ICT-based tools to support GDSD projects and (2 method for efficient use of the architecture.

The following table presents research question, step and methods of phase 4.

Research questions	Methods	Steps to be undertaken
Q5.1 What is an appropriate ICT-based architecture to support GDSD projects?	System engineering	**VI.a** Based upon the review and analysis of phases 1-3, define the objectives and scope of the proposed architecture. **VI.b** Define criteria to assess success of the proposed architecture. **VI.c** Using the principles of system engineering, develop a conceptual design of the integrated architecture.

	Subjective/ Argumentative (expert panel review)	**VII.a** Test the architecture: a group of experts will be asked to evaluate the architecture.
		VII.b Assess results of the evaluation of the architecture, revise objectives and scope if necessary, and develop proposals for re-design.
		VII.c Iterate to step VI.a, until satisfactory results are achieved.
Q5.2 How can this architecture be effectively used?	Engineering (of method)	**VI.d** Developing a method to provide guidelines for efficient use of the ICT-based tools integrated into the architecture.

Table 4: Description of phase 4

We are looking for definitions of system architectures proposed in literature. As for now, the one we have found the most suitable for an integrated architecture of ICT-based tools is a definition proposed by Van Der Linden and Muller [12]. According to their definition, the systems architecture we are developing will include:

- system structure, broken into hardware and software components
- visible attributes of these components, such as interfaces, resource usage, and other non-functional requirements
- constraints imposed on the components design
- system standards that all components must meet

The deliverable of the research project will be conceptual design of the architecture.

The recent tendency in software development industry as well as IS research is to integrate tools/systems with different specialization (usually best in their field or best of breed tools) and not develop a single tool/system that can do everything. Accordingly, in this project we are proposing an integrated architecture for these tools. Given the requirement that the tools operate in a globally distributed environment, they will need to work on globally distributed and globally accessible Internet/Intranet platforms. Consequently, the integrated architecture will be built using component-based modeling methods.

2.3 Selection of Research Methods

The research combine a *qualitative case study* (step III) and *system engineering* (steps VI and VII) methodologies.

The *qualitative case study methodology* (based on *positivist* philosophical assumption) has been chosen as a research strategy for steps III.a-III.c for a number of reasons. The two main reasons are: (1) The theoretical model of functional requirements for ICT-based tools, which will be employed in the empirical part of the research, comprises too many variables. Consequently it will not be possible to formulate and test quantitative hypotheses. (2) The research project is aimed at understanding of field-based requirements for ICT-based tools. In order to "understand people and the social and cultural contexts within which they live" [13], the research project employs a qualitative research methodology. Qualitative research offers an opportunity to gain in-depth insight in the actual problems management faces in global projects [14]. As such, the methodology is considered to match the *descriptive* character of this stage of the research. Qualitative research offers a range of methods, such as Action research, Case study research and Ethnographic research [13]. The choice for *positivist case study research* as an appropriate research strategy is based on (1) the characteristics of the research and (2) the contingencies of case study research as proposed by Yin [15]. (1) Typically case studies are proposed when previous research and theory directly supporting the research are limited. (2) "A case study method is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident" [15].

3. Earlier Research and Status of the Area

To address our main research question *what ICT-based support is appropriate for GDSD projects*, we have studied academic, trade and professional literature (both traditional and on Internet) in:

- traditional and global *project management* (generic and software)
- *software development* (traditional and global)
- distributed *coordination* and *control*
- *concurrent engineering* (co-located and distributed)

In addition we have reviewed the literature on *virtual projects*, *organizations* and *teams*, *remote*. Furthermore, we will conduct a market analysis to get a picture of the current state of ICT-based tools. The literature review is presented under two sub-topics:

⇒ Theoretical developments in the area

⇒ Practical developments – prototype and existing ICT-based tools (commercial and academic).

3.1 Theoretical Developments in the Area

This literature can be classified as literature on global system development (I) and literature on virtual teams and remote collaboration (II).

(I) Global system development

Essential part of this literature comes from experience in managing projects. Most reported research focuses on different issues of onsite and offshore outsourcing (especially in automotive and software industry). It primarily presents case studies conducted within real projects, and/or surveys and interviews of project managers, executives and product developers. However, only a limited number of empirical studies currently exist that discuss management issues in GDSD projects with a *project* unit of analysis. The few studies that exist [3,4,7] report numerous problems with distributed projects. Global distribution has brought new challenges to the already problematic management of software projects. First, the lack of face to face communication in a geographically distributed project reduces the "observability" of the software products thereby compounding the problems of "visualization" identified by Brooks [1]. Furthermore, the time, governance, infrastructure, and culture gaps associated with the geographical dispersion of work make it more difficult to manage inter-site work dependencies and to coordinate and control the distributed work. Distance also leads to a loss of "teamness" and causes further problems in inter-site communications. From a management perspective, the reduced "observability" introduces difficulties related to integration of organizational structures, management and development practices, tools and configuration management systems [4]. Thus moving to global distribution can be a painful process for organizations. Carmel [3] paraphrases one software manager: "no one in their right mind would do this".

In his book based on ten years of research, Carmel, while arguing that there are *five centrifugal forces* influencing global software teams (figure 3), proposes *six centripetal forces* for successful global software teams (figure 4):

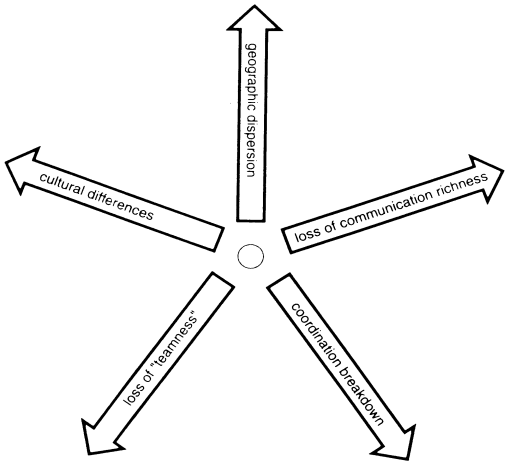


Figure 3: Five centrifugal forces

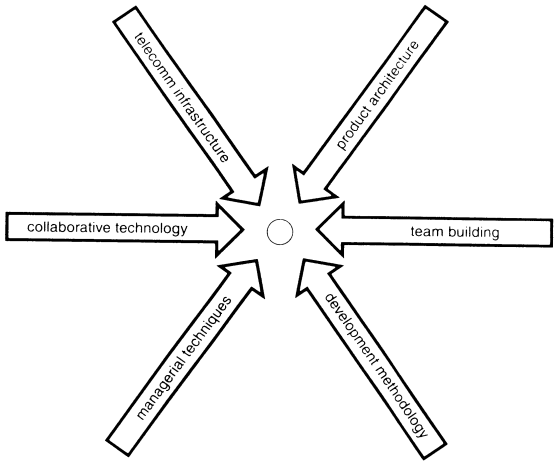


Figure 4: Six centripetal forces

The *Collaborative Technology* (one of the *centripetal forces*), Carmel defines as:

1. *Generic Technology* that includes well-known tools listed in the time-place matrix (figure 5):

Place \ Time	same	different
different	video-conference audio-conference e-chat e-whiteboard	e-mail voice-mail video-mail groupware platform calendar/scheduling discussion list
same	meetingware	

Figure 5: The time-place matrix for *Generic Technology* (adopted from [3]).

2. *Technology to Support Software Engineering* that provides the following functions:
   1) Software Configuration Management
   2) Project status
   3) Notification services
   4) Projects schedule and tasking
   5) CASE and process management
   6) Programming tools
   7) Bug and change tracking
   8) Team memory and knowledge center

Summarizing, the available literature on GDSD shows that project distribution has greatly affected the *communication, coordination,* and *control mechanisms* used in such projects. Due to the "gaps" inherent in a globally distributed development situation, mechanisms traditionally used in a co-located mode have been found to have reduced effectiveness in a distributed environment. However, at present it seems there is no comprehensive and cohesive theoretical basis for managing GDSD/IS projects. Carmel's work presents important conclusions that could be seen as background for theory of managing such projects, but the whole work does not have a shape of a theory. Initial theoretical framework is presented by Van Fenema [7]. This theory is based on an integration of existing theories from different fields and their contribution for (1) Coordination and Control Theory and (2) Distributed Work Coordination and Control. The theory first presents an integrated view on *coordination* and *control mechanisms* (see

appendix 1) then explains how they are affected by global distribution. It also discusses how they are adapting for a distributed mode. However this theory is very recent and it has not yet been tested for managing real projects.

(II) Virtual teams and remote collaboration

This literature mainly deals with communication problems between teams and team members working on relatively small task, where a task is not considered to be a part of the project/process. Although in those studies a typical unit of analysis is a *team* (and not *project* as in our research), still some of the studies, especially those focusing on use of tools for remote communications, are relevant for management of GDSD projects.

3.2 Surveying Existing ICT-Based Tools

We are now conducting a survey of commercial tools currently available on the market. As earlier mentioned, our target tools are Project Management, CASE, Groupware and Workflow tools. Furthermore, we are examining prototypes of academic tools that fit in the above-mentioned categories developed to support either co-located or distributed development of physical products and/or software.

3.2.1 Prototypes of Academic Development Support Tools

When studying the literature about tools, we have found several projects that have developed models and prototype tools to support concurrent engineering. These tools are primarily used in the context of physical product development, and mainly in a co-located mode. However, we believe that some ideas proposed in that literature may also be applied to the support of GDSD. This belief may be justified as transmitting digital components over distance is less challenging than physical components and distribution of software development does not necessary require duplication of equipment. Consequently, as far as the product visualization and sharing is concerned, distributed software development is not likely to result in as difficult logistics challenges as visualizing physical products and components at a distance.

We will briefly present two related projects, whose ideas will be employed in our research:

1) *Project Coordination Board (PCB)* conducted by the Concurrent Engineering Research Center of West Virginia University. Londono et al. [16] propose a computer-based system called the PCB to facilitate coordination of group works by an electronically-networked (virtual) team of product developers. The main idea of the system is that there is a *common workspace* that "is equivalent to a meeting table around which product developers gather to discuss and to reach consensus in traditional engineering environments" [16]. A prototype of the PCB was developed. However this prototype did not implement all the ideas proposed by the authors. We did not find any further publications of the research group that was related to the PCB system. It seems that even though the model underlying the PCB systems was developed some time ago, there is still no computer-based tool that implements all their ideas.

2) *Global Engineering COordination Support (GECOS) project* conducted by TAI Research Center of Helsinki University of Technology. The goal of the project was to develop a system, methods, and recommendations to address the problem of managing virtual development teams in traditional manufacturing environments (i.e., virtual prototyping of physical products). The project was concluded in February 2000. However, it did not reach all of its goals. From private communication we have been told that most of the software produced was not finalized for real use due to the lack of time and resources. The project however did result in tool prototypes. It produced some technological studies and two master theses, which describe the software prototypes. The software itself is owned by the "consumer" industries and is not publicly available.

The GECOS project has been conducted with strong participation of industry. Furthermore we have found more projects developing tools to support distributed/ concurrent system engineering. The commonalties between the projects are:

1. Establishment of a research group, which can be characterized as a "brain" of the projects.
2. Support of and basis in "consumer" industries that funded the projects and were used for case studies and testing of pilot tools.
3. Participation of technical partners - industries to which technical (software) development were outsourced (based on theoretical findings of the research group).

Existence of projects that develop tools, and the fact that the majority of them were contracted and funded by industry, clearly shows the need of industry for the support for product development. However, two of the projects that are most relevant to a distributed development context have, as of today, not been completed and were not in the realm of distributed software development. This further underlines the challenge of developing ICT support for distributed work projects.

3.2.2 Commercial Tools

According to our proposition presented in section 3.1, we will also conduct an analysis of four groups of tools available commercially in the market

1. *Project Management tools* support traditional project management activities, such as planning and scheduling, some support resource and budget management. Advanced tools include enterprise-wide systems that are able to anticipate problems, errors and bottlenecks by appropriately allocating resources during forward and backward passes (simulation) through the list of activities of the project [17].
2. *CASE tools* are used to support software development process, mainly analysis and design (modeling).
3. *Groupware tools* (also referred as GSS, collaborative technology and Computer-Supportive Cooperative Work) are used to support (remote) collaboration. They combine telecommunication with integrated functionality for messaging, documentation and time management, and support real-time and asynchronous communications.
4. *Workflow tools* are (1) generic tools used to model and then manage workflow and (2) off-the-shelf workflow models that sometimes require customization and used for managing workflow in organization.

4. A Model of Globally Distributed Software Engineering Environment

Based upon a survey of current research in the area we have developed model of a GDSD environment. It employs and integrates software project management principles with some of the ideas proposed in previous research (i.e. [3,7,16]). It implements the idea of the *common workspace* proposed by Londono et al. [16]. However, since their model considers the development of physical products, we have adapted the concept to apply it to software development.

The model of a GDSD environment (figure 2) defines *categories of functional requirements* for tools to support the managing of GDSD projects. The idea is to use this model as a basis for evaluation of existing tools (those described in section 3.2.2.). We are further elaborating on detailed functional requirements (based on the categories defined in the model) in order to use it as a framework for evaluation of ICT-based tools used to support GDSD projects. This section presents and briefly explains the model at its current stage.

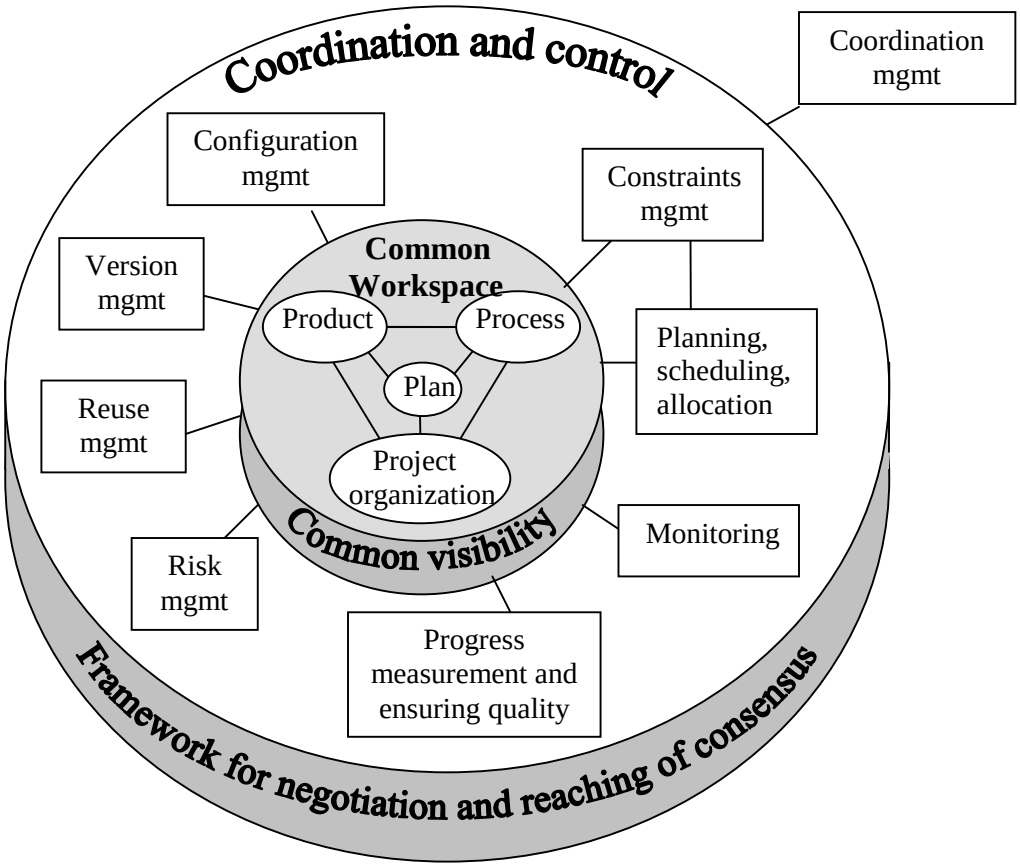


Figure 6: A model of Globally Distributed Software Engineering Environment

The inside area of the model consists of a ***common workspace*** and contains the *product*, *process*, *project organization* and *plans* that are interconnected. They are used to describe and to maintain data information required by (remote) project members during the product development life cycle.

The ***coordination and control framework***, surrounding the *common workspace*, includes the following activities:

- *Planning, scheduling, allocation*
- *Constraint management*
- *Monitoring*
- *Progress measurement and ensuring quality*
- *Risk management*
- *Configuration management*
- *Version management*
- *Reuse management*

All these activities are taking place on two organizational levels – (1)project level (i.e., inter-site/global level) and (2)intra-site level (within local teams). They are applied to all relevant components of the *common workspace*.

The third dimension of the model (dark grey) displays technical characteristics of the *common workspace* and *coordination and control framework*:

⇒ ***Common visibility*** required in a distributed environment to make *product, process, project organization* and *plans* transparent through remote sites.

⇒ ***Framework for negotiation and reaching of consensus*** provides functionality needed to support remote communications. It applies to all elements of the *coordination and control framework* and *management object* as well. On the technical level the above mentioned functionality will be provided by collaborative technology.

Together, *common visibility* and *framework for negotiation and reaching of consensus* are supposed to eliminate (or at least reduce) the perception of distance and unite remote project members by creating a joint project environment.

Crowston [18] defines coordination as the task of integrating interdependent activities and resources to achieve organizational goals. Thus, the ***coordination management*** function is supposed to integrate all resources and interdependent activities of the *coordination and control framework* to accomplish a collective set of tasks. This function is dealing with all

available coordination mechanisms, techniques and supporting technologies. It is responsible for enacting relevant coordination mechanisms and appropriate technology: to do it at the right time, in appropriate situations (under proper conditions) and within the right interface (to integrate the right parts of work or to make available communication of right people).

At the current stage the model could be applied for any software development project – co-located and distributed. However, to support distributed projects more sophisticated technical support is required. Running parts of a project concurrently at remote sites means that the project exists in several dimensions (local and inter-site). This, in turn, require strong technical support to coordinate and control project activities within each site and between sites. For example, from technical point of view, configuration and version management of remote project seems to be much more complicated that in co-located projects. Therefore when developing further each of the categories defined in the model, it will be more clear that the model support GDSD.

5. Implications of the Research

The implications of our research are relevant for both management research and management practice. From a scientific perspective, this research would provide an opportunity to test the applicability and effectiveness of existing theories and to improve them in action. From a practical point of view, this research will provide guidelines and tools for increasing the efficiency of managing GDSD projects. The project outcome will be also useful for companies developing software tools by providing them with an analysis of user requirements - requirements for tools of the growing number of companies that deploy GDSD projects.

References

1. Brooks, F. P. (1987). "No Silver Bullet." *Developer productivity* (November): 39-48.
2. The Standish Group (1995). Chaos, (http://standishgroup.com).
3. Carmel, E. (1999). *Global Software Teams: Collaborating Across Borders and Time Zones*. Upper Saddle River, NJ, Prentice-Hall P T R.
4. Karolak, D. W. (1999). *Global Software Development: Managing Virtual Teams and Environments.* Los Alamitos, California, IEEE Computer Society
5. Kumar, K. and L. P. Willcocks (1996*). Offshore Outsourcing: A Country Too Far?* European Conference on Information Systems, Lissabon, Portugal.
6. Kumar, K. and L. P. Willcocks (1999). Holiday Inn's *'*A Passage To India*'. Global Software Teams:Collaborating Across Borders and Time Zones.* Upper Saddle River,NJ, Prentice-Hall PTR.
7. Van Fenema, P. C. (2001 (forthcoming). Coordination and Control of Geographically Distributed Work: The Case of Global Information Systems Projects. *Ph.D. thesis, Erasmus University.* The Netherlands.
8. Orlikowski, W. J. and D. Robey (1991). "Information Technology and the Structuring of Organizations." *Information Systems Research* **2**(2): 143-169.
9. Iivari, J. (1996). "Why Are CASE Tools Not Used?" *Communications of the ACM* **39**(10): 94-103.
10. Kemerer, C. F. (1992). "How the Learning Curve Affects CASE Tool Adoption." *IEEE Software* (May): 23-28.
11. Srinarayan, S. and R. Arun (2000). "CASE Deployment in IS Organizations." *Communications of the ACM* **43**(1): 80-88.
12. Van Der Linden, F. J. and J. K. Muller (1995). "Creating Architectures with Building Blocks." *IEEE Software* **12**(6): 51-60.
13. ISWorld (1997). Qualitative Research Internet Site, http://www.auckland.ac.nz/msis/isworld/index.html.
14. Marshall, C. and G. B. Rossman (1995). *Designing Qualitative Research.* Thousand Oaks, CA, Sage.
15. Yin, R. K. (1994). *Case Study Research: Design and Methods*. Newbury Park, CA, Sage.
16. Londono, F., K. J. Cleetus, et al. (1992). Coordinating a Virtual Team, Concurrent Engineering Research Center, West Virginia University.
17. Gould, L. S. (1998). "Project Management Hits the Desktop.*" Automotive Manufacturing & Production* (January): 66-70.
18. Crowston, K. (1997). "A coordination theory approach to organizational process design." *Organization Science* **8**(2): 157-175.