Research proposal

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

Julia Kotlarsky
Ph.D. Candidate

Department of Decision & Information Sciences
Rotterdam School of Management, Erasmus University
P.O. Box 1738, 3000 DR Rotterdam, The Netherlands
Phone: +31-10-4081757, fax: +31-10-4089010
Email: jkotlarsky@fbk.eur.nl

Keywords: Automated IS development, CBCS, globalization of IS, IS architecture, IS project management methods and tools, research methodology

ISRL Categories: AI01, EE01, EF09, EL09, FA07, HA08
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 studies report that due to various gaps or distances between different sites, traditional coordination and control mechanisms usually become less effective in dispersed projects. They further suggest that traditional 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. Hence, the main question this research addresses 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.

The paper discusses motivation for conducting this research and elaborates on research questions. It presents the research model and methodology we intend to follow. Furthermore, it summarizes theoretical developments and initial review of tools currently available in this area, and presents initial results of first phase of the study as a model of GDSD Environment. This model defines categories of functional requirements for tools for support the managing of GDSD projects.
1. Introduction

Global distribution of software development projects has become widespread during the last years. This process of globalization has introduced new challenges in the management of projects. Thus, being an emerging area, management practice in managing GDSD has evolved primarily on an ad hoc basis. Presently there is a lack of coherent, theory-based methods and tools for managing GDSD. Our proposition is that despite a large number of tools currently available in the market (e.g. CASE tools, projects management tools, Groupware), even a combination of those tools provides only partial support for GDSD projects.

2. 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' (1987) 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 (The Standish Group 1995).

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 (Carmel 1999; Karolak 1999). 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 (Kumar and Willcocks 1996, 1999; Carmel 1999; Karolak 1999).

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 (Carmel 1999; Karolak 1999; Van Fenema 2001(forthcoming)) 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 intersite work dependencies and to coordinate and control the distributed work. Furthermore, traditional coordination and control mechanisms are less effective in GDSD projects. Carmel (1999) and Van Fenema (2001 (forthcoming) 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 are to elicit and describe the methods deployed in GDSD projects, to analyze the gap between existing ICT-based tools and these methods, and to develop and test an Internet-based integrated architecture of tools that would fill these gaps.

3. Research Approach

3.1 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 figure 1 shows, there is usually some overlap in the functionality provided by the different categories of tools (tools are explained in section 4.2.2):

**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.
3.2 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.

**Requirements**

**Phase 1: Theory- and Tool-Based requirements**
- **Literature**
  - Literature review
- **Theoretical requirements**
  - I Q1

**Tools**
- Survey of current tools
  - II Q2

**Phase 2: Field-Based requirements**
- **Market offering**
- **Gap analysis**
  - IV Q4.1, Q4.2
- **Field-Based requirements**
  - Q3.1, Q3.2
- **Projects**
  - Case study
  - Field survey

**Enhanced requirements for tools to support GDSD**

**Design**

**Phase 4: Developing integrated architecture**
- **Engineering**
  - Tools architecture to support GDSD (and method for use)
  - VI Q5.1, Q5.2
- **Experts**
  - Review/comparison

**Implementation**
- (Future research)
- Prototype

**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 conceptual 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). To explain each phase of the research and associated research questions and methods, we will use a taxonomy to classify Information Systems (IS) research methods, developed by Van Hillegersberg (1997)(appendix 1).

**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, methods and steps of phase 1.

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

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. Orlikowski and Robey 1991) 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 (Iivari 1996), while most of other CASE features are not used. Kemerer (1992) 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” (Srinarayan and Arun 2000). 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”(Orlikowski and Robey 1991). 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 analyzing the gaps between the three, and integrating them we will have a more complete set of requirements.

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

| 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? |

### Methods

| 1) Case studies (participant observation, interviews) |
| 2) Field survey using interviews and questionnaires |

### Steps to be undertaken

| 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*. |

| Q4.1 What is the gap between the field-based requirements, theory-based requirements and the functionality offered by existing tools? |
| Q4.2 What are requirements for an ICT-based architecture to support GDSD projects? |

| Subjective/Argumentative |

| IV Three-way gap analysis –of field-based requirements, market offering and theoretical requirements. |
| V Using a combination of results from previous steps, developing a set of *enhanced integrated requirements* for tools to support GDSD. |

---

**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 questions, method and steps of phase 3.

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

**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.

<table>
<thead>
<tr>
<th>Research questions</th>
<th>Methods</th>
<th>Steps to be undertaken</th>
</tr>
</thead>
</table>
| 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. |
| 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 nonfunctional requirements
- constraints imposed on the components design
Thus, system standards that all components must meet

Furthermore, architecture should be “future-proof” and minimize complexity. “Future-proof” architecture is one that covers a family of products enabling reuse, extends to accommodate the integration of additional services, and allows the exchange of hardware components and their controlling software as the hardware evolves.

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.

3.2 Selection of Research Methods

In his seminal book “Sciences of the Artificial” Nobel laureate Herbert Simon distinguishes between research methodologies for the “sciences of the natural” and “sciences of the artificial.” While the objective of the former is to find explanations for the phenomenon occurring in nature, the objective for the latter is to build and improve artifacts that improve the human condition. Simon argues that Management Science, being a pragmatic discipline, should be concerned with developing better ways for analyzing, planning, implementing, and controlling management, i.e., building and improving artifacts for management. Inspired by Simon, the proposed research project will follow the methodology of the Sciences of the Artificial.

The research combines a qualitative case study (step II) 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 II.a-II.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].

4. 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. 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 on the above mentioned topics
⇒ Practical developments – prototype and existing ICT-based tools (commercial and academic).

4.1 Theoretical Developments in the Area

Evaristo and van Fenema (1999) have presented a typology of project management that is based on two categories: number of projects and number of sites, both as single or multiple (figure 3).

<table>
<thead>
<tr>
<th>Type of Cooperation (number of locations involved)</th>
<th>Project environment (number of projects discussed)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local (one location)</td>
<td>Single project</td>
</tr>
<tr>
<td></td>
<td>I. Single co-located project</td>
</tr>
<tr>
<td></td>
<td>Site of organization (ORG 1)</td>
</tr>
<tr>
<td>Globally distributed (several sites belonging to one or more organizations)</td>
<td>Multi-project</td>
</tr>
<tr>
<td></td>
<td>III. Multiple co-located projects</td>
</tr>
<tr>
<td></td>
<td>Site of organization (ORG 2)</td>
</tr>
<tr>
<td></td>
<td>II. Single project distributed over multiple sites</td>
</tr>
<tr>
<td></td>
<td>IV. Multi-site multi-project environment</td>
</tr>
</tbody>
</table>

Figure 3: A typology of project management (based on Evaristo and Fenema, 1999)

We will conduct a survey of resources (literature, methods etc.) currently available for those four categories, and based upon the survey will decide on which of the categories (II or IV) we will focus in our research.

When studying the literature, we were interested in finding out if general project management concepts were applicable to the management of software development projects of each of the above four types. For example we were interested in understanding if the concepts for managing co-located projects were equally relevant for distributed projects. We
present a brief summary of our literature study according to the four categories and expand on those studies that were found to be relevant and contributed to our research.

I. Single co-located project

This literature could be characterized as traditional management literature. A lot of studies exist on the topic of project management and management of co-located work. The majority of publications include academic textbooks, practitioner books and articles (e.g. Turner 1993; Lock 1996). The academic literature distinguishes between software project management and project management of physical system development. Projects discussed under the topic “Software project management” are large-scale projects, that is usually the case of GDSD. In addition to the educational literature, some papers present case studies addressing specific issues in project management (e.g. requirement analysis (Crowston and Kammerer 1998)).

II. Single project distributed over multiple sites (belonging to one or more organizations)

This literature can be classified as literature on global system development (II.1) and literature on virtual teams and remote collaboration (II.2).

(II.1) 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). The majority of this literature is based on experience gained by participating in distributed projects (i.e., being a project member), or by being an observer while researching such projects. It primarily presents case studies conducted within real projects, and/or surveys and interviews of project managers, executives and product developers. This literature is based on the accumulated experience and knowledge and therefore is of relevance to the development of architecture for supporting the management of GDSD projects.
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 (e.g., Carmel 1999; Karolak 1999; Van Fenema 2001(forthcoming)) 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. 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 (Karolak 1999). Thus moving to global distribution can be a painful process for organizations. Carmel (1999) 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 4), proposes *six centripetal forces* for successful global software teams (figure 5):

![Figure 4: Five centrifugal forces](image1)

![Figure 5: Six centripetal forces](image2)
Carmel defines the Collaborative Technology (one of the six centripetal forces) as:

1. **Generic Technology** that includes well-known tools listed in the time-place matrix (figure 6):

<table>
<thead>
<tr>
<th>Time</th>
<th>Place</th>
</tr>
</thead>
<tbody>
<tr>
<td>same</td>
<td>same</td>
</tr>
<tr>
<td>video-conference</td>
<td>meetingware</td>
</tr>
<tr>
<td>audio-conference</td>
<td></td>
</tr>
<tr>
<td>e-chat</td>
<td></td>
</tr>
<tr>
<td>e-whiteboard</td>
<td></td>
</tr>
<tr>
<td>different</td>
<td>different</td>
</tr>
<tr>
<td>e-mail</td>
<td>e-mail</td>
</tr>
<tr>
<td>voice-mail</td>
<td></td>
</tr>
<tr>
<td>video-mail</td>
<td></td>
</tr>
<tr>
<td>groupware platform</td>
<td></td>
</tr>
<tr>
<td>calendar/scheduling</td>
<td></td>
</tr>
<tr>
<td>discussion list</td>
<td></td>
</tr>
</tbody>
</table>

   Figure 6: The time-place matrix for Generic Technology (adopted from Carmel, 1999).

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 2001(forthcoming). 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* (figures 7 and 8) then explains how they are affected by global distribution. It also discusses how they are adapting for a distributed mode (see model for building the theory in appendix 2). However this theory is very recent and it has not yet been tested for managing real projects.

![Figure 7: Coordination mechanisms: An integrative view (adopted from Van Fenema 2001(forthcoming)).](image_url)

<table>
<thead>
<tr>
<th>Controllers:</th>
<th>Objects of control:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Input</td>
</tr>
<tr>
<td><strong>Hierarchical supervision</strong></td>
<td>Hierarchical selection of inputs</td>
</tr>
<tr>
<td><strong>Co-workers (clan)</strong></td>
<td>Ballotage</td>
</tr>
<tr>
<td><strong>Technology</strong></td>
<td>Technical monitoring of inputs</td>
</tr>
<tr>
<td><strong>Self</strong></td>
<td>Self selection</td>
</tr>
<tr>
<td><strong>Contractual party</strong></td>
<td>Selection of contractual party</td>
</tr>
<tr>
<td><strong>Third party</strong></td>
<td>Third party input control</td>
</tr>
</tbody>
</table>

![Figure 8: Control mechanisms: An integrative view (adopted from Van Fenema 2001(forthcoming)).](image_url)
(II.2) Virtual teams and remote collaboration

This literature mainly deals with communication problems between teams and team members working on relatively small tasks, 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.

III. Multiple co-located projects

This literature mainly discusses programme (program) management: “[t]he management of a portfolio of small- to medium-sized projects” (Turner 1993) developing physical products (e.g. manufacturing, construction), while the major characteristic of software development is large-scale size. If one considers distributed projects as a collection of small projects running in dispersed sites, it seems that methods for programme management could be relevant to managing GDSD projects. However, methods proposed in the literature for multi-project management - prioritization and master project scheduling (Turner 1993), and Theory of Constraints (Goldratt 1997) - imply the existence of accurate project planning (estimation of task duration). Given the high level of uncertainty and complexity, and various contingencies, accurate software project planning for global projects at this time is very challenging (Chatzoglou and Macaulay 1996). Therefore, while concepts of programme management will be useful in understanding distributed projects, it is likely that GDSD projects will require a somewhat different approach to deal with multi-site project management.

IV. Multi-site multi-project environment

At the time of this report we were not successful in finding any literature on this topic. We can conclude that there is no literature on managing multiple software projects neither co-
located nor distributed. This lack seems to be a severe disadvantage since in real software engineering organizations the most likely situation is that there are several projects running concurrently, some of which may also be globally or geographically distributed. Therefore, currently we are looking at both single and multiple distributed projects (II$^{nd}$ and IV$^{th}$ categories) as part of our scope. Given the magnitude of the task, we still need to decide if we will focus on both or one of these categories of projects.

4.2 Surveying Existing ICT-based Tools

We will conduct 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 will examine 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.

4.2.1 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 distributed software development. 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.
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. We will present three related projects, whose ideas will be employed in our research. 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.

The three projects are:

1) *Project Coordination Board (PCB)*

Londono et al. (1992) propose a computer-based system called the PCB (figure 9) to facilitate coordination of group works by an electronically-networked (virtual) team of product developers. The project was conducted by the Concurrent Engineering Research Center of West Virginia University.

![Diagram of the Project Coordination Board](image)

**Figure 9:** The Project Coordination Board (adopted from Londono et al., 1992).

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. The common workspace serves as a public place on
the computer network for the team to view the current state of the evolving product”
(Londono et al. 1992). The common workspace includes the information about product,
activities and organization (virtual team) structures that are interconnected.

Londono et al. have developed a prototype of the PCB. However this prototype did
not implement all the ideas proposed by the authors. It provided preliminary functionality to
support common visibility, workflow management, and tracking progress. According to the
authors, at the time the paper was written (in 1992) the prototype of the system was
transitioned to various customer sites for evaluation in order to guide further development.
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.

The other two projects have been conducted with strong participation of industry.
The commonalities between both 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).

2) Global Engineering COordination Support (GECOS) project
This two-year research project was 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).
Requirement for the systems were:
- Easy project set-up and management.
- Affordable, common technological infrastructure and tools. The infrastructure was to be Internet-based and was to allow remote project and configuration management, remote collaboration, and remote software maintenance.
- Development of processes: definition and support.
- Human issues: teamwork, effective management, and reward mechanisms.

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.

3) **CODESCO project: A Practical Communication and Decision Support Environment for Managing Concurrent Product Development.**

The project was conducted by research groups of University of Nottingham (UK) and University of Bremen (Germany). The goal was to develop an efficient and integrable tool for information and knowledge retrieval. The project was finalized in September 2000. It developed a Web-enabled environment, consisting of a Decision Support Module, a Communication Module and an on-line methodology for communication and decision support. These modules were tested in two “customer” companies. We have acquired a free trial release of the Communication Model from the CODESCO consortium and will test it as part of our research. According to the description of the Communication Module, it seems to be quite generic and possibly could be suitable for GDSD projects.

To conclude, several research projects have been undertaken by research groups and industries for developing tools to support concurrent product development. Mostly the
projects have concluded with prototype tools, which implement the proposed concepts only partially. Some of the proposed concepts and their related functionality are very relevant for managing GDSD projects. Especially the model of Londono et al. as it proposes the idea of the common workspace that allows remote access to the information, that is very important in the case of GDSD. Furthermore, we are going to check if we can use the ideas underlying some of these prototype tools for developing our GDSD architecture.

4.2.2 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 (Gould 1998).

2. **A collection of CASE tools** (or **CASE environment** according to the definition of Koskinen 2000) provide automated support for one or more parts of the system development life-cycle, 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.
5. A Model of Globally Distributed Software Engineering Environment

Based upon current research in the area we have developed a model of GDSD environment (figure 10). It employs and integrates some of the ideas proposed in previous research. The model combines:

- software project management principles
- *six centripetal forces* for successful global software teams of Carmel (1999)
- Theory of Distributed Work Coordination and Control of Van Fenema (2001 (forthcoming)).

It implements the idea of the *common workspace* proposed by Londono et al. (1992). However, since their model considers the development of physical products, we have adapted the concept to apply it to software development.

The model defines *categories of functional requirements* for methods and tools to support the managing of GDSD projects. The categories of functional requirements defined in the model do not make a distinction between theory-based methods/techniques (to be implemented manually) and those supported by tools. The goal of the model is to identify major functions that compose a GDSD environment so it can be used as a basis for studying field-based requirements and evaluation of existing methods and tools. Currently we are further elaborating on each category defined in the model by developing a features list. Then we will modify/adapt it to the requirements will be identified in the field and use as a framework for evaluation of ICT-based tools available in the market.

This section presents and briefly explains the model at its current stage:
Figure 10: The model of Globally Distributed Software Engineering Environment

The inside area of the model illustrates the common workspace and contains the product, process, project organization structures that are interconnected and together could be interpreted as plans. 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
- Outsourcing management
- Risk management
- Configuration management
- Version management
- Reuse management
All these activities applied to all relevant components of the *common workspace*. Consequently, most of the coordination and control activities could be considered as product-, process- and organization-oriented.

The third dimension of the model (dark grey) displays the requirements of the *common workspace* and *coordination and control framework* necessary to make available working in a distributed setting and support remote communications.

⇒ **Common accessibility** required in a distributed environment to make *product*, *process*, *project organization structures* and *plans* transparent (visible) through remote sites and to allow remote access to the required data/information.

⇒ **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 *common workspace* as well. On the technical level the above mentioned functionality will be provided by collaborative technology.

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

Crowston (1997) 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, in distributed projects (1) traditional coordination and control mechanisms are less effective and other - adapted mechanisms (Van Fenema 2001 (forthcoming)) – are required to coordinate and control project activities and (2) more sophisticated technical support is needed to support these adapted mechanisms and allow parallel (concurrent) software development from remote locations. Distribution of a project introduces need for technology that was not necessary in a co-located project. The minimal technology needed is telecommunication to allow for communications between remote team members and networks-infrastructure to make possible fast data/information exchange between sites. Often replication of a project repository is needed. Furthermore, distributed working requires establishment of common rules and procedures (e.g. how to use configuration management systems).

Since the focus of this study is on globally distributed software development, when developing further each of the categories defined in the model, we will concentrate on issues that are unique for distributed projects (compared to co-located projects).

6. Assumptions and Limitations

1. Taking into account the difference between customer-oriented software product vs. packaged software (Keil and Carmel 1995), different approaches are needed to manage those two types of projects (e.g. when developing packaged software, time-to-market very important, while custom software should be developed to conform to customer requirements). Therefore we are going to concentrate on customer-oriented projects.

2. Object-Oriented and Component–Based Development (CBD) has become widespread and has prevailed over the traditional (structured) approach for software development.
CBD means that a product is assembled from pre-existing components using Assemble-to-Order approach, while traditional development implies Design/Make-to-Order approach. For each of these two cases the development is different, and therefore differences might exist in project management techniques. However, at present the research will include both Assemble-to-Order CBD and Design/Make-to-Order software. Later we might focus on the projects using CBD.

7. Contribution of the Research: Scientific and Managerial Relevance

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, it will provide guidelines and tools for increasing the efficiency of managing GDSD projects, that, according to Simon (1996) (section 3.2), could be also considered as a scientific contribution). The project outcome will be also useful for companies developing software tools, since it will simplify on their practice by providing the companies with analysis of user requirements - requirements for tools of the growing number of companies that deploy GDSD projects.

8. References


### Appendix 1

<table>
<thead>
<tr>
<th><strong>Empirical Research Methods</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Case Study</strong></td>
<td>Examination of a single or small number of developed systems with no experimental design or controls. It includes action research and participant observation.</td>
</tr>
<tr>
<td><strong>Field Test</strong></td>
<td>Examination of several developed systems with an experimental design and controls.</td>
</tr>
<tr>
<td><strong>Survey</strong></td>
<td>Examination of several or more organisations or systems with an experimental design but no controls.</td>
</tr>
<tr>
<td><strong>Lab Experiment</strong></td>
<td>Laboratory study with an experimental design and high degree of controls.</td>
</tr>
<tr>
<td><strong>Archival Research</strong></td>
<td>The examination of historical documents or recorded data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Non-Empirical Research Methods</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subjective/ Argumentative</strong></td>
<td>Based more on opinion and speculation than observation</td>
</tr>
<tr>
<td><strong>Engineering</strong></td>
<td>Development of a new methodology, language or system</td>
</tr>
<tr>
<td><strong>Theorem Proof</strong></td>
<td>Includes mathematical logic, axiomatic proofs etc.</td>
</tr>
<tr>
<td><strong>Modelling/ Simulation</strong></td>
<td>Mathematical modelling and simulations, game/role playing</td>
</tr>
</tbody>
</table>

Classification of research approaches (adopted from Van Hillegersberg 1997).
Appendix 2

The research model in developing Theory of Distributed Work Coordination and Control (adopted from Van Fenema 2001(forthcoming))