INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI[®]

**DISTRIBUTED NEW PRODUCT DEVELOPMENT STRUCTURES AND
SCHEDULE OVERRUN**

by

Peiyao Wang

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering in Telecommunications Technology Management

Department of Systems and Computer Engineering

Carleton University

Ottawa, Canada, K1S 5B6

March 2005

© Copyright 2005 Peiyao Wang

Library and
Archives Canada

Bibliothèque et
Archives Canada

0-494-06842-6

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

The undersigned hereby recommend to
the Faculty of Graduate Studies and Research
acceptance of the thesis

**DISTRIBUTED NEW PRODUCT DEVELOPMENT STRUCTURES AND
SCHEDULE OVERRUN**

Submitted by

Peiyao Wang

in partial fulfillment of the requirements for the degree of
Master of Engineering in Telecommunications Technology Management

Rafik Goubran, Department Chair

A.J. Bailetti, Thesis Supervisor

Carleton University

March 2005

ABSTRACT

The objective of this research is to examine the relationship between attributes of new product distributed structures and schedule overrun. Data on 75 projects undertaken by Cisco Systems using distributed structures to develop new products were used to test four hypotheses. The hypotheses examined how (i) the proportion of the product design produced at project headquarters, (ii) the development load carried out at remote sites, (iii) the number of platform dependent components used in the product, and (iv) the number of components reused are related to schedule overrun. Results suggest that a high level of design carried out at project headquarters and development loads carried out at remote sites shorten schedule overrun.

Keywords: Distributed development structure, product development performance, product modularity, schedule overrun.

ACKNOWLEDGEMENTS

The author thanks very much his thesis supervisor, Dr. A. J. Bailetti, Director of the M. Eng. in Telecommunications Technology Management program at Carleton University, for his comments and suggestions on many drafts of this thesis and for his unwavering support.

The author also thanks Professor John Callahan for his review comments about the theoretical model and engineering attributes test methods.

The author sincerely thanks his wife Ying and daughter Emily who provided him with continuous support during the completion of the thesis.

TABLE OF CONTENT

ABSTRACT	iii
ACKNOWLEDGEMENTS	iv
TABLE OF CONTENT	v
LIST OF TABLES	viii
LIST OF FIGURES	ix
LIST OF APPENDICES	x
1. INTRODUCTION.....	1
2. LITERATURE REVIEW.....	7
2.1 Literature streams	7
2.1.1 Literature about development structure	8
2.1.2 Literature about development process	11
2.1.3 Literature about development strategy.....	12
2.1.4 Literature about schedule overrun	12
2.1.5 Literature about product complexity.....	14
2.1.6 Research gaps within the existing literature	14
2.1.7 Literature review summary	15
2.2 Lessons from the literature review	18
2.2.1 Distributed structure helps achieve global development efficiency	18
2.2.2 Modular design reduces component interdependency	19
2.2.3 Product modularity increases component reuse.....	20
2.2.4 Modular design enable distributed development	20
2.2.5 Component reuse reduces product development time	21
2.2.6 Engineering structures greatly affect development performance	22
2.2.7 Project rework increase schedule overrun	23
3. HYPOTHESES	24
3.1 Research model.....	24
3.2 Hypotheses development.....	26
3.3 List of hypothesis.....	29
4. RESEARCH METHODOLOGY	31
4.1 Unit of analysis	31
4.2 Study period.....	31
4.3 Sample selection	32

4.4 Variables	32
4.4.1 Number of components.....	32
4.4.2 Number of employees	32
4.4.3 Number of employees at headquarters.....	33
4.4.4 Number of locations.....	33
4.4.5 Number of components reused	33
4.4.6 Number of platform dependent components.....	34
4.4.7 Number of components designed at project headquarters	34
4.4.8 Design concentrated at project headquarters	34
4.4.9 Number of components developed by remote locations.....	35
4.4.10 Distributed development load.....	35
4.4.11 Start time.....	35
4.4.12 Scheduled duration.....	36
4.4.13 Actual duration.....	36
4.4.14 Schedule overrun	36
4.5 Data collection	37
4.5.1 Sample selection criteria.....	37
4.5.2 Sample collection procedure.....	38
4.5.3 Sample data.....	40
4.6 Data analysis.....	42
4.6.1 Descriptive statistics	43
4.6.2 Correlation tests	43
4.6.3 Hypotheses acceptance and rejection.....	44
5. RESULTS.....	45
5.1 Sample	45
5.2 Descriptive statistics.....	45
5.3 Reliability	49
5.4 Pearson correlations for explanatory variables	52
5.5 Correlations coefficients for schedule overrun	54
5.6 Regression analysis.....	56
6. DISCUSSION OF RESULTS.....	61
6.1 Summary of results.....	61
6.2 Discussion of results.....	63
6.2.1 Decisions when setting up a development project.....	63
6.2.2 Time related measures of project efficiency	64

6.2.3 Project size and duration	66
6.2.4 Engineering vs. questionnaire collected data.....	67
6.2.5 Attributes of distributed development structure	67
6.2.6 Component reuse	68
6.2.7 Concurrent process.....	68
6.2.8 Project stages and locations	68
7. CONCLUSIONS, LIMITATIONS AND SUGGESTIONS FOR FUTURE RESEARCH	70
7.1 Conclusions.....	70
7.2 Limitations	71
7.3 Suggestions for future research	72
References	73

LIST OF TABLES

Table 1. Factors identified in the academic literature.....	17
Table 2. Example of how data for the Gondor project was coded.....	42
Table 3. Descriptive statistics: raw data	46
Table 4. Descriptive statistics: variables.....	48
Table 5a. Factor analysis for explanatory variables.....	51
Table 5b. Component matrix	52
Table 6. Pearson correlation coefficients for explanatory variables.....	53
Table 7. Pearson correlation coefficients for schedule overrun.....	55
Table 8a. Results of stepwise regression	57
Table 8b. Results for the variables excluded from the stepwise regression	58
Table 9. Regression results for Model 1	60
Table 10. Pearson correlations for the variables in the research model and time related variables measuring project efficiency	65

LIST OF FIGURES

Figure 1. Positioning this research.....	4
Figure 2. Christensen theory on product development architecture	10
Figure 3. Theoretical models	16
Figure 4. Research model	25

LIST OF APPENDICES

Appendix A. An example of data available for a Cisco project	83
Appendix B. Reliability correlation matrix for explanatory variables	85
Appendix C. Ordinary least squares regressions for number of platform dependent and reused components.....	86

1. INTRODUCTION

The time reducing characteristics of (i) product strategy, (ii) development processes, and (iii) development structures have been studied in the product development literature.¹ Development structure has been found to affect development project performance (Datar, Jordan, Kekre, Rajiv, & Srinivasan, 1997; Fine, 1998; Sanchez, 1999). Two distinct product development structures have been identified: concentrated and distributed (Datar et al., 1997).² In a concentrated new product development structure, product designers and developers are located in one facility. In contrast, in a distributed structure the new product development effort is dispersed among multiple locations.

Companies use distributed development structures to achieve global efficiency (Snow, Snell, Davison, & Hambrick, 1996) and reduce the time from prototype to volume production (Datar et al., 1997). Distributed development is anchored around the notion

¹ Eisenhardt and Tabrizi, (1995), Callahan and Moretton (2001), Gopalan (1995), Griffin (1997) and Zirger and Hartley (1996) examined development process attributes such as: time spent in planning and testing, supplier, and sales and marketing involvement, supplier reduction, design freeze, number of designers using computer assisted design, degree of project overlap, number of design and load build iterations, time between milestones, project newness, project complexity and process formality; team structure attributes such as: multifunctional teams, dedicated teams, co-located teams, empowered teams, rewards for schedule attainment, power of the project manager, time as goal and senior management support; and product strategy attributes such as: product newness, incremental development, and part reduction.

² The relationships of other types of structures to project development performance have also been examined. Eisenhardt and Tabrizi (1995) examined compressed and experiential structures and Fine (1998) and Sanchez (1999) examined modular and integral structures.

of process concurrency (Sanchez, 1999). Process concurrency represents the degree to which different organizational functions simultaneously conduct project work (Tatikonda & Montoya-Weiss, 2001).

The objective of this research is to examine how key attributes of distributed structures used to develop complex software products relate to schedule overrun. A review of the literature indicates that software project schedule overrun is a major concern (Gopalan 1995; Jenkins, Naumann, & Wetherbe, 1984; Molokken & Jorgensen, 2003; Keil, Rai, Mann, & Zhang, 2003; Molokken, Jorgensen, Tanilkan, Gallis, Lien, & Hove, 2004; Reichelt & Lyneis, 1999; Standish Group, 1994; van Genuchten, 1991; Eden, Williams & Ackermann, 2005).

Modular product architectures enable distributed product development structures (Sanchez & Mahoney, 1996). Modularity allows product development tasks to be carried out concurrently in more than one location for the purpose of speeding up product introduction (Sanchez, 1999; Sosale, Hashemian & Gu, 1997). Modular components³ can be produced separately and used interchangeably in different product configurations without compromising the integrity of the system (Baldwin & Clark, 1997; Fine, 1998; Garud & Kumaraswamy, 1993; Mikkola & Gassmann, 2003). Modularity breaks up a

³ A modular component is defined as a physically distinct portion of the product that embodies a core design concept and performs a well-defined function (Henderson & Clark, 1990; Mikkola & Gassman, 2003).

complex system into discrete components. These components communicate with one another through standardized interfaces (Langlois, 1999).

Components can be divided into two categories depending on whether the company has applied these components in previous product architectures: new-to-the-firm and standard (Mikkola & Gassmann, 2003). New to the firm components refer to product-specific components that are introduced to the firm for the first time. It is useful to distinguish platform dependent components, a special type of new-to-the-firm component. A platform dependent component is a software component that supports new hardware operations such as hardware drivers. Platform dependent components are concurrently developed with the hardware platform they support. The development of platform dependent components requires that hardware and software designers be co-located. To port a platform dependent component to other platforms, development is required. A standard component refers to a component that has been used in previous or existing architectural designs by the company or qualified components that are available in the company's library (Mikkola & Gassmann, 2003).

Three considerations are critical in setting up a distributed product development structure: (i) the location of the project headquarters and the project responsibilities of the designers working in this location; (ii) the number of remote sites that will be part of the distributed product development structure and the responsibilities of the designers working in these locations; and (iii) the number of platform dependent components that will be developed and the number of standard components that will be reused.

Decisions on these considerations define four key attributes of a distributed structure: (i) the number of modules to be designed and developed by individuals co-located at project headquarters; (ii) the number of modules to be designed and developed by engineers in each remote site, (iii) the number of platform dependent modules to be developed, and (iv) the number of standard components to be reused.

Figure 1 positions this research relative to other research streams. The focus is to examine how the location where design and development activities are carried out (project headquarters and remote site) and module type (platform dependent and reused) are related to the schedule overruns of distributed development projects.

Figure 1. Positioning this research

Data from Cisco Systems' internal project code database was used for this research. There were two motivating factors for selecting Cisco Systems as the focal company for this research. First, the researcher is an employee of Cisco Systems, and has access to the required raw data and has the skills required to make sense of the results. Second, Cisco Systems dominates the data communication market.

This research is important to top management teams of companies and project managers that rely on distributed development structures to develop complex products. Companies such as IBM, Intel, and Microsoft develop complex products using distributed structures. According to a recent IDC report (IDC, 2001), U.S. companies will spend over \$17.6 billion outside the United States in 2005. This is more than three times the amount spent in 2001.

This research makes at least two contributions. First, this is the first empirical study on how the attributes of distributed projects affect schedule overrun. Second, the study extends the research on the management of development projects by focusing on the specific outcomes of engineering related decisions made by distributed development teams and their impact on schedule overrun.

The thesis is organized into seven chapters. Chapter 1 is the introduction. It describes the research objective, relevance of the research, and contributions. Chapter 2 provides the literature review. Four hypotheses are developed in Chapter 3. Chapter 4 provides the research method. Chapter 5 provides the results obtained from testing the hypotheses and

Chapter 6 discusses the results. Chapter 7 provides the conclusions, limitations and suggestions for further study.

2. LITERATURE REVIEW

This chapter reviews the literature. It is organized into two parts. The first part describes the various literature streams deemed relevant to this research. The second part identifies the lessons learned from reading the literature.

2.1 Literature streams

There are three major streams in the literature regarding product development performance. The first stream focuses on product development structure, e.g., whether to organize product development using a concentrated or distributed structure. The second stream focuses on product development processes, e.g., analysis of process sequence, time, cost, reliability, quality, and how well the output of each stage fits the following stage. The third stream focuses on key development factors, such as technology innovation, platform dependent development, component reuse, commitment, trust and so on.

Over the past decade, companies in the information technology industry have increasingly relied on outsourcing largely due to an overall industry downturn. Consequently, the use of distributed development structures has increased.

2.1.1 Literature about development structure

The overall task of product creation is being transformed into a process of modular architectures (Sanchez, 1999). Much of the current research seeks a good fit between product development structure and the task of new product creation. Various development structures are discussed in the literature: design concentrated at project headquarters and distributed structure (Datar et al., 1997); modular design and optimizing design (Sanchez, 1999); experimental design and compressed design (Eisenhardt & Tabrizi, 1995); and platform-based design and independent design (Krishnan & Gupta, 2001).

A concentrated development structure locates all product designers in one facility, while a distributed structure disperses product development effort among multiple manufacturing locations. Product prototyping is faster in a concentrated structure, though time to reach volume production is shorter in the distributed structure (Datar et al., 1997). Firms use distributed structures to achieve global efficiency, develop regional or worldwide cost/tax advantages, and standardize designs and operations (Snow, Snell, Davison, & Hambrick, 1996). Distributed development teams are now critical to the survival of many firms (Lurey & Raisinghani, 2001). The growing popularity of inter-organizational alliances, combined with a growing tendency to use flatter organizational structures and global networks has accelerated the need for large firms to coordinate activities that span geographical and organizational boundaries (Townsend, 1998).

Modular product design creates functional components with standardized interfaces, which allow the substitution of various components into product architecture without requiring changes in the design of other components. Typically, traditional optimized product design produces products with tightly coupled components. Design can be conceptualized as an iterative, recursive process that goes through many cycles of integration, testing, and fine-tuning to achieve the lowest possible cost or highest possible performance (Sanchez & Mahoney, 1996). Modular design is a function-oriented approach that can be integrated into different production systems, often with only minor modifications (Chang & Ward, 1995). Modularity separates products into subassemblies and components, reducing product complexity and increasing flexibility (Gershenson, Prasad, & Zhang, 2003; Baldwin & Clark, 1997). All component development processes can then be carried out concurrently, thereby speeding up product introductions (Sanchez, 1999).

The compression design strategy assumes that product development is predictable and can be planned out as a series of discrete steps. Such processes can then be compressed by shortening the time of each step. Experiential design strategy assumes that product development is a highly uncertain path through foggy and shifting markets and technologies. Fast development depends upon intuition and flexibility within that uncertain environment (Eisenhardt & Tabrizi, 1995).

The literature provides a rationale for the use of modularity in the increasingly distributed high-tech R&D development structure. Modularity in a distributed structure can achieve

global cost advantages and standardize design operations. This research is supported by Christensen product architecture theory (Christensen, 1997).

Figure 2. Christensen theory on product development architecture

Modular product architecture creates loosely coupled knowledge domains focused on each loosely coupled component (Sanchez, 1999). Organizations that create products tend to develop technical knowledge structures that reflect the functional decomposition of components (Henderson & Clark, 1990). Modularity in the design of products leads to modularity in the design of the organizations that produce such products (Sanchez & Mahoney, 1996). Few studies have examined modular and distributed development

structure combinations, looking for the combination that most positively affects product development performance.

2.1.2 Literature about development process

The product development process can be broken into six phases: planning; conceptual design; product design; testing; releasing process development and production start-up (Eisenhardt & Tabrizi, 1995). The process of product development can be grouped into four decision categories: concept development, supply-chain design, product design, and production and launch (Krishnan & Ulrich, 2001).

Much of the literature focuses on specific development phases in an effort to find the best development approach. Process models conducted for the purpose of analysis define a set of activities arranged in a specific order with clearly identified inputs and outputs. Within these models an output may be either a product or a service (Davenport, 1993). This type of research confirms that extensive supplier networks coupled with overlapping product development phases, proactive communication, and cross functional groups can streamline the process of product development (Clark & Fujimoto, 1991).

2.1.3 Literature about development strategy

There is a broad range of development strategies discussed within the literature. Here the focus is on the attributes of distributed development environments. The diverse backgrounds of cross-functional development team members can increase the amount and variety of information available to design products, thereby improves the efficiency and effectiveness of the design process (Brown & Eisenhardt, 1995). At the same time, in a context that includes different languages and cultures interaction within the production team becomes more complicated (Davison, 1994). The trust that is critical to a team's ability to make decisions and work well together can be built swiftly; however, this trust is often very fragile in global teams that never meet in person (Maznevski & Chudoba, 2000).

Virtual teams rapidly form, reorganize, and dissolve as the needs of a dynamic marketplace change (Jarvenpaa & Leidner, 1999). Problems associated with organization and team interactions mainly arise from the distributed organizational structure which greatly affects product development performance.

2.1.4 Literature about schedule overrun

Less than half of the projects met their time to market and budget objectives (Roberts, 1992). Studies show that 38% of the projects had an overrun of between 0 and 50%, and the average schedule overrun was 22% (Jenkins, Naumann, & Wetherbe, 1984). The

schedule overrun survey response stated that 80% of their projects were sometimes or usually late (Phan, Vogel, & Nunamaker, 1988). Schedule overrun is a widely existing product development problem, and these important to study.

The existing research about schedule overrun is mostly based on survey methodology. The author predefines possibilities or reasons, and collects data from the survey answer sheet.

The literature studies focus on the factors that affect project schedule overrun. The major factors can be categorized as planning issues, project rework issues, large and complex uncertainties issues. Time spent in planning should shorten product development time by rationalizing the process, reducing mistakes and shortening delays (Gupta & Wilemon, 1990). More time spent planning is associated with shorter software development time (Callahan & Moretton, 2001). Proper project planning, organization and tracking reduce schedule overrun (Thambain & Wilemon, 1986).

Product redesign increases schedule overrun (Griffin, 1997). Project cost and schedule overrun typically involve significant effort on rework (Reichelt & Lyneis, 1999). Renegotiating the plan and schedule increase schedule overrun (Van Genutchten, 1991).

The general uncertainties of conception generation and project evaluation increase schedule overrun (Griffin, 1997). Greater product complexity gives rise to coordination challenges during product development (Novak & Eppinger, 2001).

2.1.5 Literature about product complexity

Product complexity creates a variety of transaction costs, such as coordination cost to design and execute production (Novak & Eppinger, 2001). The number of component, component interactions and product novelty increase product complexity (Novak & Eppinger, 2001). Complex product development is defined as high cost, engineering-intensive task (Hobday, 1998). Large, complex development projects often experience substantial cost and schedule overruns (Reichelt & Lyneis, 1999).

2.1.6 Research gaps within the existing literature

The existing research identifies several development structures and many social interaction factors affecting new product development performance. Many studies examine one development structure or focus on a specific factor in the development process, but existing research lacks an engineering methodology perspective on how modular product architecture and design structure affect the duration of new product development.

Some important questions are not adequately addressed within the literature. For example, what are the major issues that surround the deployment of a standard engineering process in a distributed development environment? Are there particular

engineering models that best fit for distributed product development? How can distributed development cost savings be balanced with project schedule overrun? There is a need for good answers to these questions.

This thesis will examine the high-tech industry distributed structure phenomenon from an engineering methodology perspective, and offer an engineering model that fits well with distributed product development. Most studies have researched multiple companies' data at a certain period of time. This research is the first empirical study to examine distributed development performance based on one company's internal project code base containing thousands of projects over the past 10-year period.

2.1.7 Literature review summary

There are many factors that can affect development performance. Management needs to identify the key factors that can greatly improve new product development performance. For example how to overcome communication barriers, cultural conflicts, and time zone logistics in a distributed environment. Moreover, modular product architecture, project leadership and process management are also critical to product development. Management teams need an integrated model that streamlines the development process. Here are several models of how development structures and modular architecture can affect the duration of development cycle.

Figure 3. Theoretical models

This review is summarized in Table 1, which lists issues and key factors to be focused on or overcome in a distributed development environment. Table 1 identifies the studies on development structures and engineering dimension key factors affecting a company's development performance. The $\rightarrow$ is used to identify a relationship. $X \rightarrow Y$ means "X affects Y."

Table 1. Factors identified in the academic literature

Study	Proposition	Rationale
Datar et al.(1997)	Concentrated structure → shorten product prototype	Cross-product learning by locating all product designers in one facility.
Datar et al.(1997)	Distributed structure → shorten volume production	Allow close interaction with local customers and engineer, results in shorter time to volume production.
Sanchez & Mahoney (1996) Kirk (1990)	Modularity → product code complexity	Modularity reduces connectivity between program parts, which in turn reduces code complexity.
Langlois (1999) Kusiak (2002)	Modularity → product design flexibility	Modular products result in flexible designs
Sanchez & Mahoney (1996) Sosale et al. (1997)	Modularity → product design re-use	The re-use of an existing design with minor changes is possible with a modular architecture.
Gershenson et al. (1999).	Modularity → product life-cycle costs	Modularity used to reduce life-cycle costs by reducing the number of processes and reducing repetitive processes.
Eisenhardt & Tabrizi (1995)	Compress design strategy → shorten develop time	Based on the assumption that product development is predictable, process can then be compressed by shortening the time of each design step.
Krishnan & Gupta (2001)	Platform-based design → leverage investment	Product platforms shared across the product family enable better leverage investments in product development.
Datar et al.(1997)	Knowledge sharing → development efficiency	Cross-product learning among designers fosters rapid product introduction.

Study	Proposition	Rationale
Pauleen (2003)	Leadership → task motivation	Leadership fosters a workable balance of cohesion, unity and task motivation.
Kayworth & Leidner (2001/2002)	Leadership → decision making	Influence the effectiveness of small-group decision making by leadership and structuring of the group process.
Maznevski & Chudoba (2000)	Trust → decision processes efficiency	Trust is critical to team's ability to manage decision processes.
Snow et.al. (1996)	Commitment → task on schedule	Committed to the team's mission, can be counted on to perform their respective tasks.
Snow et al.(1996)	Standard process → group process efficiency	A group process for communicating information, making decisions, handling conflicts, and so on. Teams can then rely on their process to produce desired business results.

2.2 Lessons from the literature review

This paper provides seven lessons learned from the literature reviews.

2.2.1 Distributed structure helps achieve global development efficiency

Firms use distributed development teams primarily to help achieve regional or worldwide cost/tax advantages, and to standardize designs and operations (Snow et al., 1996).

Distributed development unavoidably creates development process concurrency. Process concurrency represents the degree to which different organizational functions simultaneously conduct project work (Tatikonda & Montoya-Weiss, 2001). Project teams rapidly form, reorganize and dissolve when the needs of a dynamic marketplace change (Jarvenpaa & Leidner, 1999). Distributed teams are playing an increasingly important role in organizational life and offer organizations the flexibility to remain competitive (Pauleen, 2003). The growing popularity of inter-organizational alliances, combined with a growing tendency to use flatter organizational structures and global networks has accelerated the need for large firms to coordinate activities that span geographical, as well as organizational boundaries (Townsend, 1998). All component development processes can then be carried out concurrently thus speeding up product introductions (Sanchez, 1999). Distributed design increases global development efficiency in cost and standards (Snow et al., 1996).

2.2.2 Modular design reduces component interdependency

Modularity is a design structure that reduces the degree of interdependency and thus the frequency and cost of communication (Langlois, 1999). Modular product architecture not only creates flexible product designs, it also enables the design of loosely coupled, flexible, “modular” organizational structures (Sanchez & Mahoney, 1996). A modular product is made up of modules, building blocks. The more products fit into modular design, the more common modules are available (Gershenson et al., 2003). Organizations often produce products and systems that mirror their internal communication process

(Conway, 1968). Modularity in the design of products leads to modularity in the design of the organizations that produce such products (Sanchez & Mahoney, 1996).

2.2.3 Product modularity increases component reuse

Ease of assembly and disassembly of modules is essential for the success of module re-use (Scheidt & Zong, 1994). The re-use of an existing design with minor changes is possible with a modular architecture (Sosale, Hashemian & Gu, 1997). Modular products result in flexible designs, which enable a company to respond to changing markets and technologies by rapidly and inexpensively creating product variants derived from different combinations of the existing or new modules (Kusiak, 2002). Software modularity reduces connectivity between program parts which in turn reduces code complexity (Kirk, 1990). Firms able to borrow new technology from one product group and apply it to another have become more efficient at R&D (Ettlie, 1998). Product modularity increases opportunities for component re-use and introduces assembling flexibility at the product component level.

2.2.4 Modular design enable distributed development

Modular component development processes can be carried out concurrently and can speed up product introductions (Sanchez, 1999). The success of the computer industry is attributable to modularity, the 'building of complex product or process from smaller

subsystems that can be designed independently yet function together as a whole (Baldwin & Clark, 1997). Modular design allows for the dividing of design tasks and for parallel development processes. This reduces time and effort needed to upgrade an existing product.

Modularity allows a designer to control the degree to which changes in processes or requirements affect the product, and promoting modularity gives designers more flexibility to meet these changing processes. Modularity is able to reduce life-cycle costs by reducing the number of processes and reducing repetitive processes (Gershenson, Prasad, & Allamneni, 1999).

2.2.5 Component reuse reduces product development time

The re-use of similar modules can reduce product development time (Allen & Skalak, 1998). Module reuse reduces time required to develop and validate code (Sosale et al., 1997). Effective reuse of standardised “common components” may significantly reduce development time (Sanchez & Collins, 2001). Reuse of component design increases the efficiency of product creation and speed to market (Sanchez, 1999).

The extent of component reuse helps designers to reduce the time to develop and test new components and also reduces the time to integrate all components into a system level product. Existing technology reuse helps to set up a component repository which results in less time needed for project management teams to locate the right component to reuse.

2.2.6 Engineering structures greatly affect development performance

Each engineering structure has natural advantages of resource allocation, which in turn greatly affects the time between product prototype and volume production. By locating all product designers in one facility under concentrated development structures, increase interaction among cross-product designers, and this cross-product learning advantage results in shorter time to develop prototypes. A distributed structure disperses product development efforts among multiple manufacturing locations. Such structure allows close interaction with local customers and production engineers with the advantage that time to reach volume production is shorter (Datar et al., 1997).

The underlying assumption of the compression strategy is that product development is a predictable process and that such a process can then be compressed by shortening the time of each step. From the experiential perspective, one way to speed up product development is through frequent iteration (Eisenhardt & Tabrizi, 1995).

Modular product architecture makes possible the rapid introduction of improved product variations as soon as improved components become available (Sanchez, 1999). Platform-based development approaches become more profitable than the independent development of products as the integration benefits from platform optimization increase and overall costs decrease (Krishnan & Gupta, 2001). Concurrent modular development speeds up product introduction (Sanchez, 1999).

2.2.7 Project rework increase schedule overrun

There have 50% of survey responses stating that frequent changes in design and implementation were usually a reason for cost overrun (Phan et al., 1988). Project cost and schedule overrun typically involve significant effort on rework (Reichelt & Lyneis, 1999), and product redesign increase schedule overrun (Griffin, 1997).

Good planning can help reduce project rework. Proper project planning, organization and tracking reduce schedule overrun (Thambain & Wilemon, 1986). Time spent in planning should shorten product development time by rationalizing the process, reducing mistakes and shortening delays (Gupta & Wilemon, 1990). Renegotiating the plan and schedule increases schedule overrun (Van Genutchten, 1991).

Moreover if sales, marketing and customers are involved, planning can avoid unnecessary project rework. Sales and marketing involvement early in the development cycle leads to shorter development time (Callahan & Moretton, 2001).

3. HYPOTHESES

This chapter is organized into three sections. Section 3.1 introduces the research model used to anchor the hypotheses developed in this research. Section 3.2 develops the hypotheses. Section 3.3 provides a list of the hypotheses tested in this research.

3.1 Research model

Figure 4 provides the research model used in this research. It illustrates how four attributes of distributed development structures relate to schedule overrun. The first two attributes highlight where software components are designed and developed. The first attribute focuses on the components designed at project headquarters while the second one focuses on the type of components used in the product: platform dependent components and standard reused components.

In a distributed structure, development effort is dispersed across development locations. One of these locations is the project headquarters; the others are referred to as remote locations. Thus, each distributed structure has $N + 1$ locations, where N is greater than zero and refers to the number of remote locations that are part of the distributed structure.

Project headquarters refers to the location where the project manager, product manager, product architect and product marketing personnel are co-located with designers. Remote sites only include personnel that undertake design and development tasks. In a remote

site, a group of experienced engineers focus on developing a certain part (e.g., component, module) that is needed for the new product.

Figure 4. Research model

A platform dependent component is a new-to-the-firm component. Platform dependent components are software components concurrently developed with the hardware platform they are designed to support. The development of platform dependent components requires that hardware and software designers be co-located. A standard component refers to a component that has been used in previous or existing architectural designs by the company or qualified components that are available in the company's library (Mikkola & Gassmann, 2003).

Keil et al. (2003), Heemstra and Kusters (1989), and the Standish Group (1994) found that overruns increase with project size. However, Jenkins et al. (1984) and van Genuchten (1991) found that the occurrence of overruns was equally likely for small, medium and large projects. Based on the empirical results reported by Keil et al. (2003), project size was used as a control variable.

3.2 Hypotheses development

The hypotheses in this section were developed in the context of distributed product structures set up to develop complex software products. The objective of this research excludes comparisons between product development structures. These differences have been examined elsewhere.⁴

The most important role of project headquarters is to enable, facilitate, and coordinate corporate-knowledge stocks and flows (Lagerstrom & Andersson, 2003). Proximity increases communication frequency and information richness (Daft & Lengel, 1986; Zirger & Hartley, 1996). Concentrating design production in project headquarters increases communication frequency and communication richness between designers and the project leader, product manager, sales and marketing personnel, and other functions. Co-location shortens the time required for designers to learn about

⁴ The time reducing characteristics of concentrated and distributed structures have been examined by Datar et al., (2001); characteristics of compressed and experiential structures have been examined by Eisenhardt and Tabrizi (1995); and characteristics of modular and integral structures have been examined by Fine (1998) and Sanchez (1999).

product requirements and other company products (Datar et al., 1997) and the time required for the multifunctional project team to solve problems, make decisions, understand constraints, identify potential problems, and learn about project uncertainties (Zirger & Hartley, 1996). Designers working at project headquarters can learn about customer requirements, potential problems and other projects faster than designers working at remote locations. This shortens the time that designers located at project headquarters will take to produce a design. Co-location also may increase the likelihood that downstream problems will be observed early in the development process, when they are easier to correct (Imai, Nonaka, & Takeuchi, 1985). For software development projects, the benefits of co-location at project headquarters may reduce downstream problems that cause rework. Therefore,

Hypothesis 1: High levels of design concentrated at project headquarters shorten schedule overrun.

Modularity is an approach used to organize complex products and processes efficiently (Baldwin & Clark, 1997). The modular approach decomposes complex tasks into simpler activities so they can be managed independently. Modularity allows software components to be produced separately without compromising system integrity (Mikkola & Gassman, 2003). Modularity also allows the modules of a software product to be produced concurrently in more than one location for the purpose of speeding up product introduction (Sanchez, 1999; Sosale, Hashemian & Gu, 1997). Structures with high process concurrency, such as distributed development structures, strongly influence time

to market (Tatikonda & Montoya-Weiss, 2001). Development delays in one location will not affect development in other locations. Coordination costs are higher for high concurrency processes. The costs involved coordinating many remote sites each of which develops a small number of modules may be too high. The existence of coordination costs suggests that the average number of modules undertaken by remote sites is the attribute that affects operational outcomes of distributed development structures. A distributed project structure's ability to develop software components may be largely contingent on the number of remote sites with the right development capacity and how the development load is balanced across the remote sites and the project headquarters. Therefore,

Hypothesis 2: High levels of distributed development loads shorten schedule overrun.

A product is comprised of components that are platform dependent. Platform dependent components support new hardware operations such as hardware drivers. To port a platform dependent component to other platforms, development is required. Platform independent components are easily portable to other platforms. Van Genutchten (1991) reports that software components that are developed in parallel with hardware components cause overruns. The use of dependent components increases the schedule risk of the project because designers need to spend more time developing software, which is new to the firm, and verifying hardware-software integration. The use of platform development components prevents designers in remote sites from developing software components concurrently. Therefore,

Hypothesis 3: High levels of platform dependent development increases schedule overrun.

Module reuse reduces the time required to develop and validate code (Sosale et al., 1997) and integrate components into a system level product. The greater the number of product component reused reduces the time engineers spend developing and testing new components. A great majority of product development projects reuse existing components (Karagozoglu & Brown, 1993). Component repositories reduce the time engineers spend searching for the right components to reuse. Effective reuse may significantly reduce development time (Sanchez & Collins, 2001). Therefore,

Hypothesis 4: High levels of component reuse shorten schedule overrun.

3.3 List of hypothesis

The hypotheses to be tested in this study are:

Hypothesis 1: High levels of design concentrated at project headquarters shorten schedule overrun.

Hypothesis 2: High levels of distributed development loads shorten schedule overrun.

Hypothesis 3: High levels of platform dependent development increases schedule overrun.

Hypothesis 4: High levels of component reuse shorten schedule overrun.

4. RESEARCH METHODOLOGY

This chapter describes the unit of analysis, the study period, sample selection, how the variables were measured, how the data was collected, and how the data was analyzed.

4.1 Unit of analysis

The unit of analysis was a software development project undertaken by Cisco Systems using a distributed development structure.

4.2 Study period

Cisco Systems undertook the software development projects examined in this research from January 1995 to December 2004. This research was undertaken using Cisco Systems' project code database. This database was accessed during November 2004 to February 2005.

4.3 Sample selection

The sample was drawn from the Cisco Systems software project code database. This database includes all development projects undertaken using a distributed structure from 1995 to 2004.

4.4 Variables

4.4.1 Number of components

Number of components refers to the number of components used to develop a new product.

4.4.2 Number of employees

Number of employees refers to the number of employees allocated to the project after the commitment to execute the project is made. This includes development engineers, designers, project managers and product marketing personnel.

4.4.3 Number of employees at headquarters

This refers to the number of employees allocated to the project working at the project's headquarters. Normally designers, project managers and product marketing personnel are all located at headquarter.

4.4.4 Number of locations

Number of locations refers to the number of development locations involved in the development of the product. The number of location is equal to the number of remote locations involved in the development of the product plus one. The one accounts for the project's headquarters.

4.4.5 Number of components reused

Number of components reused refers to the number of standard components that have been used in previous or existing architectural designs by the company or qualified components that are available in the company's library (Mikkola & Gassmann, 2003).

4.4.6 Number of platform dependent components

Number of platform dependent components refers to the number of components that rely on a product specific hardware or platform and need to be developed concurrently with the hardware.

4.4.7 Number of components designed at project headquarters

The number of components designed at headquarters refers to the number of components that were designed centrally at the location where the project manager, product architect designer and product marketing personnel were located.

4.4.8 Design concentrated at project headquarters

Design concentrated at project headquarters was calculated as the ratio between the number of components designed at project headquarters and the total number of components used to build the new product.

4.4.9 Number of components developed by remote locations

The number of components developed by remote locations refers to the total number of components developed at all remote locations.

4.4.10 Distributed development load

Distributed development load was calculated as the ratio between the number of components developed by remote locations and the number of remote locations involved in the building of the new product.

4.4.11 Start time

Start time refers to the date of the project execution commit.

A project execution commit is a top management team approval of product requirements. The date of project execution commit marks the start of resource allocations and development funding.

4.4.12 Scheduled duration

Scheduled duration refers to the difference in weeks between the date the product was to go be ready for shipment and the date of the project execution commit.

4.4.13 Actual duration

Actual duration refers to the time between the actual date the product was ready for shipment and the date of the project execution commit.

4.4.14 Schedule overrun

Schedule overrun is the number of weeks by which the actual duration of the development project exceeded the estimated duration (Gopalan, 1995). Project overrun refers to the difference between the first customer shipment date and the originally scheduled time to market date.

4.5 Data collection

The researcher collected the data required from Cisco Systems' project code database.

The data was collected using a four-step process. First, projects were selected randomly using the information on project creation date. Project dates fell within the study period, January 1995 to December 2004.

Second, the researcher checked whether the project charged with the design of the product was set up as a distributed structure project and used a modular architecture. This entailed examining the project code database to ensure that the project involved at least two development locations and that the product architecture was modular.

Third, the researcher made sure that the product was shipped to customers.

Fourth, the researcher ensured that all the information listed in section 4.4 was available for the project in the database.

4.5.1 Sample selection criteria

Project data was collected randomly from Cisco Systems' project code database using the date in which the project was added to the database.

This database contains information on approximately 3,000 approved developed projects and shipped products.

All projects in the sample met the following criteria: (i) project was developed using a distributed structure with at least two development locations and (ii) the product was designed using a modular structure.

4.5.2 Sample collection procedure

Cisco Systems project code database contains information on approximately 3,000 development projects. Product project name, business unit, and the date that the record was added to the database are the three fields that can be used to sort the database.

The most important design document is the executive plan. It includes data on the date the project started, expected duration, and number of employees.

The most important development document is the program plan. It includes data on number of components, component types, number of components reused, and locations used to design and develop the project.

Obtaining the data required for this research was a time consuming efforts. The collection of the data for 23 projects took three weeks of full time effort.

Data was collected in two waves. During the first wave, 300 projects were selected. Of the 300, 130 were projects set up as distributed development structures with a modular architecture. Of the 130, the data required for this research was available for 23 projects.

The second wave of data collection benefited from what was learned during the first wave. From the first wave it was known that selecting 300 projects would yield approximately 23 projects with a distributed structure, modular architecture and all the data required. The 23 represent 8 % of the 300 projects selected initially.

To reach a sample size of 75 projects, 52 more projects were needed. It was decided that another 650 projects had to be selected randomly ($.08 \times 650 = 52$).

The random sampling plan consisted in partitioning the database into 26 subsets sorting by project name. From each subset, the records selected during the first wave were blocked and then 25 records were selected. This provided the 650 records from where the 52 projects could be selected.

4.5.3 Sample data

For each project, 11 fields were collected. To illustrate the data collected for each development project, the data for the Gondor project is provided below. This project entailed the development of a high density Gigabit Ethernet module for a middle range router.

Project execution commit date (EC): the date that management team approved the product requirement design and started to allocate man power and development funding. This date was December 1, 2001.

Target time to market (TTM): from execution commit (December 1, 2001) until project to market date (March 1, 2003) was a total of 60 weeks.

Product first customer shipment (FCS): from execution commit, until the first customer shipment (FCS) (June 3, 2003), a total of 72 weeks.

Engineer head count (ENG_TT): hardware engineer 8, software engineer 18, dev-test engineer 10. Total of 36 engineers were involved.

Number of employees at headquarters (ENG_HQ): hardware engineers, 8, software engineers, 14.

Number of locations (SITE): 3

Components number in total (COMP_TT): total 29 components in this project.

Components reusable (COMP_RU): leverage Share portable adapter (SPA) designs. 7 software components reused from SPA.

Components with platform dependent design (COMP_PD): 2 hardware platform dependent components.

Components central designed (COMP_CD): 19 components were designed at project's headquarters.

Components remote developed (COMP_RV): 7 components were developed at remote locations.

Project name: Gondor.

Table 2 provides an example of how the data for the Gondor project was coded.

Table 2. Example of how data for the Gondor project was coded

IDX	TTM	FCS	ENG_TT	ENG_HQ	SITE	COMP_TT	COMP_RU	COMP_CD	COMP_RV	COMP_PD
1	60	72	36	22	3	29	7	19	7	2

4.6 Data analysis

The first step of the data analysis was to provide descriptive statistics for the variables in this research and examine whether or not they were normally distributed. For each variable, Skewness and Kurtosis statistics were calculated and then used to assess whether or not the variable was normally distributed.

The second step was to test the null hypothesis that pairs of variables were associated. If the variables were normally distributed, Pearson's correlation coefficients analysis would be performed. Otherwise, Spearman correlation analysis will be performed. The findings were used to test the stepwise regression between factor and result variables.

The third step used regression analysis to explore the relationships between multiple component attributes and schedule overrun.

4.6.1 Descriptive statistics

Descriptive statistics for all the variables were calculated using SPSS statistical tool and the results used to determine whether the variables were normally distributed. To decide whether or not a variable was normally distributed, its Kurtosis and Skewness statistics were examined. A variable was considered to be normally distributed when its Skewness statistics was less than twice the standard error and its kurtosis was within the +2 and -2 range.

4.6.2 Correlation tests

Correlation analysis and ordinary least square regressions were used to test the hypotheses. If the variables were normally distributed, Pearson correlation tests would be used, and if the variables were not normally distributed, Spearman bi-variate correlation tests would be used.

A stepwise regression with forward selection using all the explanatory variables will be performed. If two or more correlated variables are entered, separate linear regressions will be used to examine the regression coefficients of the explanatory variables.

4.6.3 Hypotheses acceptance and rejection

Pearson or Spearman correlation coefficients will be used to test the four hypotheses. A hypothesis is supported when the correlation coefficient has the sign (+ or -) that was predicted and it was significant at $p < .10$.

Regressions will also be used to test the hypotheses. The number of independent variables depends on the number of variables that are not correlated. A hypothesis is supported when the regression coefficient for the independent variable in the regression has the sign that was predicted and it is significant at $p < .05$.

5. RESULTS

This chapter is organized into six sections. The first section describes the sample and the second provides descriptive statistics for the raw data and the variables used in this research. The third, fourth and fifth sections provide the results of the Pearson correlation analysis. The sixth section provides the regression results.

5.1 Sample

The sample is comprised of 75 development projects undertaken by Cisco Systems from 1995 to 2004. Each project in the sample: (i) used two or more locations to develop new products, (ii) had a modular architecture for the product, and (iii) shipped product.

5.2 Descriptive statistics

Table 3 provides the descriptive statistics for the raw data used to calculate the variables in the research model illustrated in Figure 4.

Table 3. Descriptive statistics: raw data**(Sample size = 75)**

	Min.	Max.	Mean	Standard deviation	Skewness Statistics (Std error = .277)	Kurtosis Statistics (Std error = .548)
Number of components	11.00	83.00	40.51	16.155	.733	.128
Number of employees	10.00	84.00	35.95	15.885	.782	.278
Number of development locations	2.00	5.00	3.29	0.997	.216	-.998
Number of employees at headquarters	7.00	53.00	21.20	9.520	.872	.592
Number of components reused	.00	17.00	6.49	3.998	.807	.147
Number of components designed at headquarters	11.00	68.00	31.77	13.947	.772	.209
Number of components developed at headquarters	9.00	61.00	26.03	10.848	.737	.731
Number of platform dependent components	1.00	18.00	6.80	4.070	.627	-.020

Scheduled duration	9.00	81.00	33.63	17.496	.719	-.116
Actual duration	12.00	92.00	40.81	20.672	.710	-.247
Schedule overrun (in weeks)	.00	35.00	7.18	5.819	2.055	6.459

Table 3 indicates that schedule overrun is not normally distributed. Thus, the square root of schedule overrun was used as the dependent variable.

Table 4 provides the descriptive statistics for the eight variables used in this research. The first four are the explanatory variables illustrated in Figure 4: (i) design concentrated at project headquarters, (ii) distributed development load, (iii) number of platform dependent components, and (iv) number of components reused. The next three variables are proxies for the control variable, size: (v) number of employees, (vi) number of components, and (vii) scheduled duration. The last variable shown in Table 4 is the dependent variable: (viii) square root of schedule overrun.

Table 4. Descriptive statistics: variables**(Sample size = 75)**

	Min.	Max.	Mean	Standard deviation	Skewness Statistics (Std error = .277)	Kurtosis Statistics (Std error = .548)
Design concentrated at project headquarters	47.83	100	77.42	12.319	-.457	-.373
Distributed development load	1.00	18.00	6.86	3.819	.868	.250
Number of platform dependent components	1.00	18.00	6.80	4.070	.627	-.020
Number of components reused	.00	17.00	6.49	3.998	.807	.147
Number of components	11.00	83.00	40.51	16.155	.733	.128
Number of employees	10.00	84.00	35.95	15.885	.782	.278
Scheduled duration	9.00	81.00	33.63	17.496	.719	-.116
Square root of schedule overrun	.00	5.92	2.48	1.019	.489	1.262

For the eight variables in Table 4, the Kurtosis statistic is within the +2 and -2 range. For two of the eight variables in Table 4, design concentrated at project headquarters and schedule overrun, the Skewness statistic was less than twice the standard error. The value of the Skewness statistics for (i) distributed development load, (ii) number of platform

dependent components, (iii) number of components reused, (iv) number of components, (v) number of employees, and (vi) scheduled duration are slightly greater than twice the standard error. This indicated a slight deviation from normality for the distributions of these six variables.

5.3 Reliability

Factor analysis was used to detect the latent structure of the first seven variables shown in Table 4. Tables 5a and 5b show the results of the factor analysis. Table 5a shows that the first three principal components have eigenvalues greater than one. Together these three factors explain almost 74 percent of the total variance.

The following five variables had the highest loadings in the first component: number of components, number of employees, number of reused components, number of platform dependent components, and scheduled duration. Distributed development load had the highest loading in the second component and, design concentrated at project headquarters had the highest loading in the third component.

The Cronbach alpha for the five variables with high loadings in the first component was 0.4564. Given that the Cronbach alpha was much smaller than 0.80, it was concluded that these five variables do not measure a single one-dimensional latent construct.

The Cronbach alpha for the three control variables, number of components, number of employees and scheduled duration was .6474. This result suggests that the three control variables were not measuring a one-dimensional latent construct (i.e., size). The Cronbach alpha for number of components and number of employees was 0.7834. This was close enough to 0.80. Thus, it was concluded that these two variables were measuring the same construct.

Based on the results of the Cronbach alpha tests, it was decided to use two control variables: number of employees and scheduled duration.

Table 5a. Factor analysis for explanatory variables**(Sample size = 75)**

Principal Component	Initial Eigenvalues			Extraction Sums of Squared Loadings		
	Total	% of Variance	Cumulative %	Total	% of Variance	Cumulative %
1	2.649	37.846	37.846	2.649	37.846	37.846
2	1.431	20.445	58.291	1.431	20.445	58.291
3	1.097	15.669	73.690	1.097	15.669	73.960
4	.785	11.221	85.182			
5	.614	8.772	93.954			
6	.282	4.024	97.978			
7	.142	2.022	100.00			

Table 5b. Component matrix

	Component 1	Component 2	Component 3
Number of employees	.730	-.495	.107
Number of components	.930	6.120E-02	.140
Scheduled duration	.473	-5.37E-02	-.540
Number of components reused	.785	.462	2.964E-02
Numbers of platform dependent components	.527	-.560	.214
Design concentrated at project headquarters	-1.68E-02	.331	.828
Distributed development load	.364	.736	-.207

5.4 Pearson correlations for explanatory variables

Table 6 provides the Pearson correlation coefficients (2 tailed tests) for the explanatory variables shown in the research model in Figure 4.

Table 6. Pearson correlation coefficients for explanatory variables**(Sample size = 75)**

	Design concentrated at project headquarters	Distributed development load	Number of platform dependent components	Number of components reused	Number of employees
Distributed development load	-.005 (.968)	1			
Number of platform dependent components	-.041 (.726)	-.053 (.651)	1		
Number of components reused	.127 (.279)	.464*** (.000)	.075 (.522)	1	
Number of employees	-.107 (.363)	-.121 (.300)	.491*** (.000)	.347*** (.002)	1
Scheduled duration	-.163 (.163)	.096 (.413)	.134 (.251)	.302*** (.008)	.245** (.034)

* p < 0.10 (two tailed)

** p < 0.05 (two tailed)

*** p < 0.01 (two tailed)

Two tailed tests of the alternative hypothesis at $p < 0.10$ show that there were linear associations for the following pairs of variables:

- distributed development load and number of components reused
- number of platform dependent components and number of employees
- number of components reused and number of employees
- number of components reused and scheduled duration
- number of employees and scheduled duration

Examination of the results in Table 6 indicates that four variables are not correlated among themselves: design concentrated at project headquarters, distributed development load, number of platform dependent components, and scheduled duration

5.5 Correlations coefficients for schedule overrun

Table 7 provides the results of the Pearson correlation analysis (one tailed tests) for the explanatory variables and schedule overrun.

Table 7. Pearson correlation coefficients for schedule overrun
(Sample size = 75)

	Design concentrated at project headquarters	Distributed development load	Number of platform dependent components	Number of components reused	Number of employees	Scheduled duration
Square root of schedule overrun	-.356*** (.001)	-.333*** (.002)	.225** (.026)	-.150 (.101)	.171* (.071)	.435*** (.000)

* p < 0.10 (one tailed)

** p < 0.05 (one tailed)

*** p < 0.01 (one tailed)

One tailed tests of the alternative hypothesis at $p < 0.10$ show that:

- design concentrated at project headquarters and distributed development are negatively related to the square root of schedule overrun
- number of platform dependent components, number of employees and scheduled duration are positively related to the square root of schedule overrun

5.6 Regression analysis

Stepwise regression

Tables 8a and 8b show the results of using the stepwise regression with forward procedure. The adjusted R square for the stepwise regression of the most significant model is 0.388. The F statistic is significant at $p < 0.01$.

Three variables were entered into the regression in the following order: (1) scheduled duration; (2) distributed development load, and (3) design concentrated at project headquarters.

Table 8a shows that the regression coefficients for design concentrated at project headquarters (-0.289) and distributed development (-.376) are both negative and significant at $p < 0.01$. The correlation coefficient for scheduled duration (0.424) is positive and significant at $p < 0.01$.

Table 8a. Results of stepwise regression

Explanatory Variables included	Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Adjusted R square	Std. error of the estimate
	B	Std. Error	Beta				
(Constant)	4.189	.667		6.280	.000	.388	.797
Distributed development load	-.100	.024	-.376***	-4.110	.000		
Design concentrated at project headquarters	-2.39E-02	.008	-.289***	-3.135	.002	F	Sig.
Scheduled duration	2.469E-02	.005	.424***	4.578	.000	16.628** *	.000

* p < 0.10

** p < 0.05

*** p < 0.01

For each variable excluded from the stepwise regression model, Table 8b provides the beta in estimate, t, significance, partial correlation and collinearity statistics.

Table 8b. Results for the variables excluded from the stepwise regression

	Beta In	t	Sig.	Partial Correlation	Collinearity Statistics
					Tolerance
Numbers of platform dependent components	.139	1.526	.132	.179	.977
Number of components reused	-.098	-.891	.376	-.106	.687
Number of employees	-.010	-.106	.916	-.013	.915
Number of components	-.047	-.464	.644	-.055	.821

Ordinary least squares regressions

To examine the relationship between number of employees and schedule overrun, the following regression model was specified.

Model 1

$$\text{Schedule overrun} = b_0 + b_1 * \text{design concentrated at project headquarters} + \\ b_2 * \text{distributed development} + \\ b_3 * \text{number of employees} +$$

Table 9 shows the results for Model 1 obtained using ordinary least squares regression.

The adjusted R square for Model 1 is 0.216. The F statistic is significant at $p < 0.10$. The regression coefficients for design concentrated at project headquarters is -0.348 and distributed development load is -0.324. Both are negative and significant at $p < 0.01$.

The regression coefficient for number of employees is not significant at $p < 0.10$.

Table 9. Regression results for Model 1

Explanatory Variables included	Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Adjusted R square	Std. error of the estimate
	B	Std. Error	Beta				
(Constant)	5.083	.771		6.593	.000	.216	.902
Design concentrated at project headquarters	-2.88E-02	.009	-.348***	-3.359	.001		
Distributed development load	-8.64E-02	.028	-.324***	-3.122	.003		
Number of employees	6.063E-03	.007	.095	.906	.368	F	Sig.
						7.805 ***	.000

* p < 0.10

** p < 0.05

*** p < 0.01

6. DISCUSSION OF RESULTS

This chapter is organized into two parts. The first part provides a summary of the results. The second part provides a discussion of the results and links these results to those published in the literature.

6.1 Summary of results

The objective of this research was to examine how project attributes that result from decisions made while setting up a distributed product development structure relate to schedule overrun. Four hypotheses were tested using data on development projects undertaken by Cisco Systems Inc. from 1995 to 2004.

Hypothesis 1 predicted that high levels of design concentrated at project headquarters shorten schedule overrun. This hypothesis was supported. Indeed, as indicated in Table 7, the Pearson correlation coefficient for design concentrated at project headquarters and the square root of schedule overrun is negative and significant (-0.356 , $p < 0.10$). Moreover, the regression coefficients for design concentrated at project headquarters shown in Tables 8a and 9 are also negative and significant, -0.289 and -0.348 respectively, $p < 0.10$.

Hypothesis 2 predicted that high levels of distributed development loads shorten schedule overrun. This hypothesis was supported. In Table 7, the Pearson correlation coefficient

for distributed development load and the square root of schedule overrun is negative and significant (-0.333 , $p < 0.10$). The regression coefficients for distributed development load in Tables 8a and 9 are also negative and significant, -0.376 and -0.324 , $p < 0.10$.

Hypothesis 3 predicted that the number of platform dependent components used is positively associated schedule overrun. This hypothesis was partially supported. The Pearson correlation coefficient for number of platform dependent components and the square root of schedule overrun shown in Table 7 is positive and significant (0.225 , $p < 0.10$). However, as indicated in Table 8b, the number of platform dependent components was not entered into the stepwise regression. The “beta in” for number of platform dependent components is positive with $p < 0.132$, close to $p < 0.10$.

Hypothesis 4 predicted that higher levels of component reuse shorten schedule overrun. This hypothesis was not supported. In Table 7, the Pearson correlation coefficient for the number of components reused and the square root of schedule overrun is not significant at $p < 0.10$. Moreover, the results of the stepwise regression procedure in Table 8b show that the number of components reused was not a variable entered into the regression equation.

6.2 Discussion of results

This research focuses on how decisions made to set up distributed development projects anchored around modular designs affect schedule overruns. The focus of this research excludes development decisions within a project and concentrated structures.

6.2.1 Decisions when setting up a development project

Research in product development can be organized based on the fundamental decisions that are made by intent or default into: (i) decisions in setting up a development project and (ii) development decisions within a project (Krishnan & Ulrich, 2001). Decisions in setting up a development project can be further organized into: (i) decisions on product strategy and planning; (ii) decisions on the organization that will carry out the development of the product; and (iii) decisions on project management (Krishnan & Ulrich, 2001).

To reduce schedule overrun, the results of this research suggest that:

- decisions on product strategy and planning should focus on reducing the number of platform dependent modules developed for the new product
- decisions on the team location should focus on concentrating the production of the product design in project headquarters and increasing the average number of modules developed in remote sites
- decisions on project management should focus on schedule overrun control

The results of the stepwise regression analysis suggest that concentrating design at project headquarter and distributing development load in remote sites are more effective at reducing schedule overrun than reducing the number of platform dependent components and increasing the number of components reused.

6.2.2 Time related measures of project efficiency

Project performance is multifaceted, difficult to measure and can be measured in a variety of ways (Griffin & Page, 1996). Profits, revenues, market share and customer satisfaction can be used to measure project effectiveness while product quality, project cost, unit cost, and time related measures can be used to measure project efficiency (Brown & Eisenhardt, 1995; Tatikonda & Montoya-Weiss, 2001). Time related measures of project efficiency include development time (Callahan & Moretton, 2001; Eisenhardt & Tabrizi, 1995; Griffin, 1997; Zirger & Hartley, 1996), schedule overrun (Gopalan, 1995; Reichelt & Lyneis, 1999; van Genutchen, 1991); and time-to-market (Datar et al., 1997; Griffin & Page, 1996; Tatikonda & Montoya-Weiss, 2001).

This research focuses on schedule overruns (difference between actual duration and scheduled duration). Table 10 provides the Pearson correlations (two tailed) for the four attributes examined in this research and three time-related variables associated with

project efficiency: schedule duration, actual duration, and the square root of schedule overrun.

Table 10. Pearson correlations for the variables in the research model and time related variables measuring project efficiency
(Sample size = 75)

	Scheduled duration	Actual duration	Schedule overrun (square root)
Design concentrated at project headquarters	-.163 (.163)	-.239** (.039)	-.356*** (.002)
Distributed development load	.096 (.413)	.009 (.937)	-.333*** (.003)
Number of platform dependent components	.134 (.251)	.176 (.132)	.225* (.053)
Number of components reused	.302*** (.008)	.221* (.057)	-.150 (.199)

*** $p < 0.01$

** $p < 0.05$

* $p < 0.10$

Results shown in Table 10 suggest that the four variables examined in this research are related to scheduled duration, actual duration and schedule overrun in different ways. Design concentrated at project headquarters is not related to scheduled duration, but it is negatively correlated to actual duration and scheduled overrun. Distributed development load

and number of platform dependent components, are not related to scheduled duration or actual duration, but are related to schedule overrun. The number of components reused is associated to the scheduled duration and actual duration but not to schedule overrun.

6.2.3 Project size and duration

The Cronbach alpha for number of components and number of employees in the project is 0.78. We can conclude that both variables are measuring the same dimension and that this dimension can be referred to as project size. Do scheduled duration and actual duration measure project size in the same way that number of components and number of employees do?

To answer this question, two Cronbach alphas were calculated. The Cronbach alpha for number of components, number of employees in the project, and scheduled duration was 0.64 and the one for number of components, number of employees in the project, and actual duration was 0.60.

The results of the reliability analysis suggest that scheduled duration and actual duration do not measure the same dimension that number of total components and number of employees in the project do. It is surmised that scheduled duration and actual duration may measure project complexity as well as project size.

6.2.4 Engineering vs. questionnaire collected data

Previous studies on scheduled overrun and development time used questionnaires to collect data. This study uses actual engineering data. To the writer's knowledge this may be the first time that actual engineering data has been used to examine how decisions made when setting up projects affect schedule overruns.

6.2.5 Attributes of distributed development structure

Development structure, development process, and product development strategy affect project efficiency (Zirger & Hartley, 1996). Datar et al. (1997), Snow et al. (1996), and Scott (2001) examine differences between development structures. Datar et al. (1997) found that when a development project is set up as a concentrated structure, the time required to produce a prototype is shorten and that when the project is set up as a distributed structure the time required for volume production is shorten. However, the research reported by Datar et al. did not identify the specific factors that affect project performance. The research carried out by the writer identifies four attributes of distributed development structures and the results of testing the hypotheses suggest that concentrating design production at headquarters and distributing development load at remote sites decreases schedule overrun. This adds to what was published by Datar et al. (1997).

6.2.6 Component reuse

Sosale et al. (1997), Gershenson et al. (2003), and Kusiak (2002) report that modular designs increase components reuse. Sosale et al. (1997) and Sanchez and Collins (2001) reported that component reuse decrease development time. The results of this study suggest that the number of components reused is positively related to duration and is not related to schedule overrun. Thus, the findings from Sosale et al. (1997) and Sanchez and Collins were not found to hold for distributed development structures.

6.2.7 Concurrent process

Sanchez (1999) and Tatikonda and Montoya-Weiss (2001) argue that the ability to carry out development concurrently reduces development time. The average development load on each remote location reflects the concurrent engineering capability of the firm. The results of this research show that the greater the distributed development load, the shorter the schedule overrun. Thus, the results of this research are consistent with the results published by Sanchez (1999) and Tatikonda and Montoya-Weiss (2001).

6.2.8 Project stages and locations

Research that examines how activities undertaken at different stages of the development cycle affect development time break down the development process into various phases:

planning; conceptual design; product design; testing; releasing process development and production start-up (Eisenhardt & Tabrizi, 1995; Callahan & Moretton, 2001).

The results of this research suggest that the location where design and development activities are undertaken affects schedule overrun. To reduce schedule overrun, early stage activities should be carried out centrally at project headquarters while downstream development activities can be carried out in remote locations.

7. CONCLUSIONS, LIMITATIONS AND SUGGESTIONS FOR FUTURE RESEARCH

7.1 Conclusions

This research examined the relationship between dimensions of distributed project structures and schedule overrun. Four hypotheses were developed and tested using data on 75 distributed projects carried out by Cisco Systems.

The key findings from this research include:

- Concentrating the production of the design at project headquarters where sales and marketing personnel are located shortens schedule overrun
- Distributing the development load in remote locations shortens schedule overrun
- Number of platform dependent components used may increase schedule overruns
- Number of components reused does not affect schedule overrun
- Scheduled and actual duration do not measure project size in the same way that number of components and number of employees assigned to the project do

7.2 Limitations

This study has at least three important limitations. The first limitation is that the sample was drawn from only one company. It is difficult to generalize these results.

The second limitation is the lack of a variable measuring product complexity in the research model shown in Figure 4. Complexity may affect schedule overrun (Hobday, 1998; Novak & Eppinger, 2001). The number of product components, component interactions, and product novelty increase product complexity (Novak & Eppinger, 2001). Complex product development is a high cost, engineering-intensive task (Hobday, 1998). Large and complex development projects often experience substantial cost and schedule overruns (Reichelt & Lyneis, 1999). Greater product complexity gives rise to coordination challenges during product development (Novak & Eppinger, 2001). Product complexity directly affects design and develops effort and cost.

The third limitation is that the research did not consider the executive priority of the development projects. Executives assign a greater priority to certain projects. This was not taken into account in the research model provided in Figure 4.

7.3 Suggestions for future research

The first suggestion is to test the model shown in Figure 4 using a sample comprised of distributed projects drawn from companies in various industries.

The second suggestion for future research is to include measures of complexity and project priority in the model shown in Figure 4.

The third suggestion for future research is to examine how the number of standard interfaces amongst components affects the numbers of components reused and schedule overrun. The greater the number of standard interfaces deployed, the larger the number of common reusable components that will be available. Future research can test the relationship between the number of standard components defined and schedule overrun.

References

Allen, K. R., & Skalak, C. S. 1998. Defining product architecture during conceptual design. *Proceedings of the 1998 ASME Design Engineering Technical Conference*, Atlanta, Georgia, DETC98/DTM-5650.

Baldwin, C. Y., & Clark, K. B. 1997. Managing in an age of modularity. *Harvard Business Review*, **75(5)**: 84–93.

Brown, S. L., & Eisenhardt K. M. 1995. Product development: Past research, present findings, and future directions. *Academy of management review*, **20**: 343-378.

Callahan, J. & Moretton, B. 2001. Reducing software product development time. *International Journal of Project Management*, **19(1)**: 59-70.

Clark, K. B., & Fujimoto, T. 1991. Product development performance: Strategy, organization, and management in the world auto industry. Boston, MA: Harvard Business School Press.

Chang, T.S., & Ward, A. C. 1995. Design-in-modularity with conceptual robustness. *21st International Conference on Advances in Design Automation*, ASME, **82(1)**: 493-500.

Christensen, C. M. 1997. The innovator's dilemma: When new technologies cause great firms to fail. Boston, MA: Harvard Business School Press.

Conway, M. E. 1968. How do committees invent? *Datamation*, **14(10)**: 28-31.

Daft, R. L., & Lengel, R. H. (1986). Organizational information requirements, media richness and structural design. *Management Science*, **32**: 554-571.

Datar, S., Jordan, C., Kekre, S., Rajiv, S., & Srinivasan, K. 1997. New product development structures and time-to-market. *Management Science*, **43(4)**: 452-464.

Davison, S. C. 1994. Creating a high performance international team. *Journal of Management Development*, **13(2)**: 81-90.

Davenport, T. H. 1993. Process innovation, reengineering work through information technology. Boston, MA: Harvard Business School Press.

Eden, C., Williams T., & Ackermann F. 2005. Analysing project cost overruns: comparing the “measured mile” analysis and system dynamics modeling. *International Journal of project management*, **23**: 135-139

Ettlie, J. E. 1998. R&D and global manufacturing performance. *Management Science*, **44(1)**: 1-11.

Eisenhardt, K. M., & Tabrizi, B. N. 1995. Accelerating adaptive processes — product innovation in the global computer industry. *Administrative Science Quarterly*, 40(1): 84–110.

Fine, C. H. 1998. Clockspeed: Winning industry control in the age of temporary advantage. Boston, MA: Perseus Books Publishing.

Garud, R., & Kumaraswamy, A. 1993. Changing competitive dynamics in network industries: An exploration of Sun Microsystem's open system strategy. *Strategic Management Journal*, 14: 351-369.

Gopalan, S. 1995. **Reasons for schedule overrun in software development projects.** Master Thesis, Carleton University, Ottawa

Griffin, A. & Page, A.L. (1996) PDMA success measurement project: Recommended measures for project success and failure. *Journal of Product Innovation Management*. Vol. 13: 478-496.

Griffin, A. 1997. The effect of project and process characteristics on product development cycle time. *Journal of Marketing Research*, 34: 24-35.

Gupta, A. K., & Wilemon, D. L. 1990. Accelerating the development of technology based new products. *California Management Review*, 32(2): 24-44.

Gershenson, J. K., Prasad, G. J., & Zhang, Y. 2003. Product modularity: definitions and benefits. *Journal of Engineering Design*, 14(3): 295–313

Gershenson, J. K., Prasad, G. J., & Allamneni, S. 1999. Modular product design: a life-cycle view. *Journal of Integrated Design and Process Science*, 3: 3–26.

Heemstra, F.J., & Kusters, R.J. 1989. Controlling software development costs: a field study. *Proceedings conferentie organization and information Systems*, 13: 13-15.

Henderson, R.M., & Clark, K. B. 1990. Architectural innovation: The reconfiguration of existing product technologies and the failure of established firms. *Administrative Science Quarterly*, 35: 9-30.

Hobday M. 1998. Product complexity, innovation and industrial organisation. *Research Policy*, 26: 689-710.

IDC report, 2001. Offshore outsourcing increasing from U.S. *InfoWorld*, <http://www.infoworld.com/articles/hn/xml/01/02/28/010228hnoutsource.html> (Accessed March 15, 2005)

Imai, K., Nonaka, I., & Takeuchi, H. 1985. Managing the new product development process: How Japanese companies learn and unlearn. *Harvard Business School Press*, The Uneasy Alliance: 337-381.

Jarvenpaa, S. L., & Leidner, D.E. 1999. Communication and trust in global virtual teams. *Organization Science*, **10(6)**: 791-815.

Jenkins, A. M., Naumann, J. D., & Wetherbe, J. C. 1984. Empirical investigation of systems development practices and results. *Inform Manage*, **7**: 73-82

Karagozoglu, N., & Brown, W. B. 1993. Time-based management of the new product development process. *Journal of product innovation management*, **10**: 204-215.

Kayworth, T. R., & Leidner, D. E. 2001/2002. Leadership effectiveness in global virtual teams. *Journal of Management Information Systems*, Winter, **18(3)**: 7–40.

Keil, M., Rai, A., Mann, J. E. C., & Zhang, P. 2003. Why software projects escalate: The importance of project management constructs. *IEEE Transactions on Engineering Management*, **50(3)**: 251-261.

Kirk, B., 1990. Understanding the art of modular engineering. *Microprocessors and Microsystems*, **14**: 181–185.

Krishnan, V., & Ulrich K. 2001. Product development decisions: A review of the literature. *Management Science*, **47(1)**: 1-21.

Krishnan, V., & Gupta, S. 2001. Appropriateness and impact of platform-based product development. *Management Science*, **47**: 22-36.

Kusiak, A., 2002. Integrated product and process design: a modularity perspective. *Journal of Engineering Design*, **13(3)**: 223-231.

Langlois, R. N. 1999. Modularity in technology, organization and society. *Conference on “The roots and Branches of Organizational Economics” September 26-27*. Stanford University.

Lurey, J. S., & Raisinghani, M. S. 2001. An empirical study of best practices in virtual teams. *Information & Management*, **38(8)**: 523-544.

Lagerstrom, K., & Andersson, M. 2003. Creating and sharing knowledge within a transnational team: The development of a global business system. *Journal of World Business*, **38(2)**: 84-95.

Maznevski, M. L., & Chudoba, K. M. 2000. Bridging space over time: global virtual team dynamics and effectiveness. *Organization Science*, **11(5)**: 473–492.

Mikkola, J. H. & Gassmann, O. 2003. Managing modularity of product architectures: Towards an integrated theory. *IEEE Transactions on Engineering Management*, 50(2): 204-218.

Molokken, K., & Jorgensen, M. 2003. A review of surveys on software estimation. *IEEE International Symposium of Empirical Software Engineering*. September: 223-230

Molokken, K., Jorgensen, M., Tanilkan, S. S., Gallis, H., Lien, A. C. & Hove, S. E. 2004. Project estimation in the Norwegian software industry: A summary. Lysaker, Norway: *Simula Research Laboratory*.

Novak S., & Eppinger S. D. 2001. Sourcing by design: Product complexity and supply chain. *Management Science*, 47(1): 189–204.

Pauleen, D. J. 2003. Leadership in a global virtual team: an action learning approach. *Leadership and Organization Development Journal*, 24(3): 153-162.

Phan, D., Vogel, D., & Nunamaker, J. 1988. The search for prefect project management. *Computerworld*, September: 95-100

Reichelt, K., & Lyneis, J. 1999. The dynamics of project performance: Benchmarking the drivers of cost and schedule overrun. *European Management Journal*, 17(2): 135-150.

Roberts, E. B. 1992. Strategic management of technology: global benchmarking. MA and PA Consulting Group, London.

Sanchez, R. 1999. Modular architecture in the marketing process. *Journal of Marketing*, **63**: 92-11.

Sanchez, R., & Mahoney, J. 1996. Modularity, flexibility, and knowledge management in product and organization design. *Strategic Management Journal*, **17**: 63-76.

Sanchez, R., & Collins, R. 2001. Competing and learning in modular markets. *Long Range Planning*, **34**: 645-667

Scheidt, L. G., & Zong, S., 1991. An approach to achieve reusability of electronic modules. *Transactions of the IEEE*, June: 331–336.

Scott, G., 2001. Strategic planning for technology products. *R&D Management*, **31**(1): 15-26

Standish Group. 1994. *The chaos report*. The Standish Group.
http://www.standishgroup.com/sample_research/chaos_1994_1.php (Accessed March 15, 2005).

Snow, C. C., Snell, S.A., Davison, S. C., & Hambrick, D.C. 1996. Use transnational teams to globalize your company. *Organizational Dynamics*, **24(4)**: 50-67.

Sosale, S., Hashemian, M., & Gu, P. 1997. Product modularization for reuse and recycling. *Concurrent Product Design and Environmentally Conscious Manufacturing*, **94**: 195–206.

Tatikonda, M. V., & Montoya-Weiss, M. M. 2001. Integrating operations and marketing perspectives of product innovation: The influence of organizational process factors and capabilities on development performance. *Management Science*, **47(1)**: 151-172.

Thambain, H. J., & Wilemon D. L. 1986. Criteria for controlling projects according to plan. *Project Management Journal*, June: 75-81

Townsend, A. M., DeMarie, S. M., & Hendrickson A. R. 1998. Virtual teams: Technology and the workplace of the future. *Academy of Management Executive*, **12(3)**: 17-29.

Van Genuchten, M. 1991. Why is software late? An empirical study of reasons for delay in software development. *IEEE Transactions on Software Engineering*, **17(6)**: 582-590.

Zirger, B.J., & Hartley, J., 1996. The effect of acceleration techniques on product development time. *IEEE Transactions on Engineering Management*, 43(2): 143-152.
Identical article in: *The Journal of Product Innovation Management*, 13(6): 555-556.

Appendix A. An example of data available for a Cisco project

One of the example data record is coming from project “Gondor”, the project is try to develop high density Gigabit Ethernet module on middle range router.

Record (Project) index (IDX): 1

1. Target time to market (TTM): from concept commit (Dec01) till project scheduled time to market date (March03), total 60 weeks
2. Product FCS (FCS): from concept commit (Dec01) till FCS (Jun03), total 72 weeks
3. Engineer head count (ENG_TT): hardware engineer 8, software engineer 18, dev-test engineer 10. in total 36 employee involved
4. Components number in total (COMP_TT): total 29 components in this project
5. Number of employee in HQ (ENG_HQ): hardware engineer 8, software engineer 14
6. Number of locations(SITE): 3
7. Component reusable (COMP_RU): leverage Leto&GE SPA designs. 7 SW components reusable from Leto
8. Components with platform dependent design (COMP_PD): 2 hardware platform dependent components.
9. Components central designed (COMP_CD): 19 components were designed at project’s headquarters.
10. Components remote developed (COMP_RV): 7 components were developed at remote locations.

11. Project name: Gondor.

Table 2 provides an example of how the data for the Gondor project was coded.

Table 2. Example of how data for the Gondor project was coded

IDX	TTM	FCS	ENG_TT	ENG_HQ	SITE	COMP_TT	COMP_RU	COMP_CD	COMP_RV	COMP_PD
1	60	72	36	22	3	29	7	19	7	2

Appendix B. Reliability correlation matrix for explanatory variables

Table B.1. Reliability correlation matrix for explanatory variables**(Sample size = 75, exclude one-dimensional variables)**

Cronbach Alpha=.2645 Std item Alpha=.4487	Concentrated design at project headquarters	Distributed development load	Number of platform dependent components	Number of component reuse	Number of Employees
Concentrated design at project headquarter	1.0000				
Distributed development load	-.0048	1.0000			
Number of platform dependent components	-.0411	-.0530	1.0000		
Number of component reuse	.1265	.4641	.0751	1.0000	
Number of Employees	-.1066	-.1212	.4914	.3470	1.0000
Scheduled development time	-.1629	.0959	.1342	.3025	.2448

Appendix C. Ordinary least squares regressions for number of platform dependent and reused components

Table C.1. Regression results for Model 2

Explanatory Variables included	Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Adjusted R square	Std. error of the estimate
	B	Std. Error	Beta				
(Constant)	3.956	.678		5.832	.000	.399	.790
Design Concentrated at project headquarters	-2.37E-02	.008	-.286***	-3.135	.003		
Distributed development load	-9.77E-02	.024	-.366***	-4.037	.000		
Number of platform dependent components	3.482E-02	.023	.139	1.526	.132		
Scheduled development time	2.358E-02	.005	.405***	4.372	.000	F	Sig.
						13.268 ***	.000

* p < 0.10

** p < 0.05

*** p < 0.01

Table C.2. Regression results for Model 3

Explanatory Variables included	Unstandardized Coefficients		Standardized Coefficients	t	Sig.	Adjusted R square	Std. error of the estimate
	B	Std. Error	Beta				
(Constant)	4.435	.729		6.082	.000	.152	.938
Design concentrated at project headquarters	-2.74E-02	.009	-.331***	-3.068	.003		
Number of platform dependent components	5.516E-02	.027	.220	2.049	.044		
Number of components reuse	-3.17E-02	.028	-.125	-1.150	.254	F	Sig.
						5.417 ***	.002

* p < 0.10

** p < 0.05

*** p < 0.01