An Investigation of

Hardware and Software Mindsets

by

Roy R. Cantu III

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degrees of

Bachelor of Science in Computer Science and Engineering

and Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

May 28, 1996

Author [illegible]
Department of Electrical Engineering and Computer Science
May 28, 1996

Certified by [illegible]
Wanda J. Orlikowski
Thesis Supervisor

Accepted by [illegible]
F.R. Morgenthaler
Chairman, Department Committee on Graduate Theses

An Investigation of
Hardware and Software Mindsets
by
Roy R. Cantu III

Submitted to the
Department of Electrical Engineering and Computer Science

May 28, 1996

In Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Given the growing importance of software versus hardware in modern computer systems, it is becoming increasingly necessary for companies traditionally rooted in hardware development to migrate to a predominantly software development environment. This migration requires an understanding of how individuals and organizations view and approach hardware and software. The concept of mindsets provides a means of characterizing the assumptions, expectations, and knowledge that individuals and organizations use to understand a technology. This paper introduces the notions of software and hardware mindsets and applies them in a case study of a computer company, Gamma Corporation, to assess their role in the migration from hardware to software development. The results revealed congruence in the ways in which individuals in Gamma approached hardware and software engineering problems and large differences in the ways in which Gamma's organizational practices and procedures were structured to manage hardware and software projects. Such structural incongruence appeared to be associated with some difficulties in the management of software projects at Gamma. Some implications of these results for software and hardware development are discussed.

Thesis Supervisor: Wanda J. Orlikowski
Title: Associate Professor of Information Technologies

DEDICATION

To my parents for waiting this long.

TABLE OF CONTENTS

INTRODUCTION

This thesis addresses the cognitive and cultural issues that arise as engineers and corporations attempt to adapt to the evolving role of developing and managing software and hardware. An investigation into this area yields answers as to how individuals approach hardware and software and organize themselves around it. Furthermore, it provides insight into the organizational changes that are necessary in order for technological evolution to properly proceed within the organization.

The literature supports the notion that given the phenomenal growth of hardware computing power, it is software that will continue to demand more and more resources in the development of modern computer systems. Curtis (1995, p. xvii) describes three epochs in computer systems development:

> *In the first epoch only the hardware mattered. Hardware was expensive. Software was cheap, and so were people who developed it. Only the elite used computers, and they could master the most arcane of interfaces. Programmers learned to write programs from wizards. Programs were called 'beautiful' because they used the hardware artfully, not because most people could use them.*
>
> *In the second epoch hardware became cheaper and software grew far larger and more expensive. Swarms of programmers began emerging from the universities. The growing uses for computers required an avalanche of software. Software development became a team activity. People were told to manage what had a few years before been considered an art. Programs were called 'beautiful' because other programmers could understand them, not because most people could use them.*
>
> *The third epoch started with Jerry Weinberg's* The Psychology of Programming *in 1971. Hardware eventually became a commodity and everyone started buying personal computers. Having bought computers, they had to figure out how to use them. Common folk had crashed the ball, but thanks to the complexity of the steps, they couldn't join the dance. The first mavens of usability design used social science terms like 'software psychology' and programmers paid little attention. Programs will now be called 'beautiful' because their functionality and interface are usable, and most people will use them.*

As a result of the increasing presence of software, computer development organizations traditionally rooted in hardware development have had to react quickly so as not to be left in software's wake. Specifically, the primary problem that has arisen may be stated as follows: Given the increasing importance of software in the development and maintenance of modern computer systems, how do organizations -- whose focus for years has been primarily hardware development -- migrate their culture, practices, and mindsets of their managers and engineers so that they can effectively produce both hardware and software? This thesis attempts to explore this problem via a case study of a computer organization. The organization chosen here, Gamma Corporation (a pseudonym), has been in the hardware development business for over 25 years and has only in the past five years recognized the increasing importance of software to its business.

Yin states that the purpose of a case study is to answer questions of "how" and "why" (Yin, 1989, p. 20). Examination of Gamma Corporation provides insight into how managers and engineers approached and viewed hardware and software development and the role of both within the organization. The case study also examines the experiences of this particular organization in its attempt to make the transition from an organization firmly rooted in hardware to one in which hardware and software coexist.

The ways in which engineers, managers, and organizations approach hardware and software is analyzed from the perspective of mindsets. Mindsets provide a way of characterizing the cognitive influences on how individuals and organizations act and approach issues, and is a concept which is discussed fully in the next chapter.

Having developed the concept of mindsets, the research setting and methodology will be described. In chapter four, the results are presented and discussed. The thesis concludes with suggestions for how computer organizations may effectively make the transitions necessary to support the changing nature of computer systems development.

LITERATURE REVIEW

This chapter provides a literature review of the concept of "mindsets" and how this concept has been and can be applied to understanding software and hardware development and management.

The Concept of Mindsets

The literature on social cognition states that individuals act based on their own interpretations, or frames of reference, of the world (Berger and Luckman, 1967; Smircich and Stubbart, 1985; Weick, 1979; cited in Orlikowski and Gash, 1994, p. 176). Gioia describes an individual's frame of reference as "a built-in repertoire of tacit knowledge that is used to impose structure upon, and impart meaning to, otherwise ambiguous social and situational information to facilitate understanding" (Gioia, 1986, p. 56). Orlikowski and Gash (1994, p. 178) use the term "technological frame" to identify "that subset of members' organizational frames that concern the assumptions, expectations, and knowledge they use to understand technology in organizations." In particular, they write (1994, p. 175):

> *To interact with technology, people have to make sense of it; and in this sense-making process, they develop particular assumptions, expectations, and knowledge of the technology, which serve to shape subsequent actions toward it.*

An individual's perspective on a technology is inherently influenced by his/her technological frame. This frame is composed of the assumptions, preconceptions, and knowledge of a technology that will inevitably affect the way in which that individual approaches a particular technology.

Frames can be shared in the sense that a set of core attributes are seen to be in common among individuals. This notion is termed "congruence" by Orlikowski and

Gash and can be understood when compared to the concept of inheritance, in which each individual is distinct but certain base characteristics are shared with a family or parent. The concept of congruence/incongruence is important with respect to the notions discussed in this paper. Orlikowski and Gash (1994, p. 180) define the concept this way:

> *Congruence in technological frames would imply, for example, similar expectations around the role of technology in business processes, the nature of technological use, or the type and frequency of support and maintenance. Incongruence implies important differences in expectations, assumptions, or knowledge about some key aspects of the technology.*

The primary focus of this thesis is to define and understand frames of reference as they are applied to understanding the development of software and hardware. It is expected that some incongruence will arise, and that this will need to be identified and mechanisms proposed for organizations to deal with them effectively. Specifically, the purpose of this paper is not to propose a new (or variant) method of analyzing people's attitudes towards technology, but rather to use such already established concepts as stepping stones that will aid in understanding the specific problems involved with how individuals and organizations approach hardware and software. The terms that will be used henceforth are software mindsets and hardware mindsets, where mindset indicates a specific application of the heretofore described concept of technological frames. Therefore, I will use the following definitions:

- *Software mindset*: The underlying assumptions, expectations, and knowledge that individuals have about software systems and their development.
- *Hardware mindset*: The underlying assumptions, expectations, and knowledge that individuals have about hardware systems and their development.

Hardware and Software Mindsets

Humans are "nonmodular." The major problems of our work are not so much technological as sociological in nature. (DeMarco and Lister, 1987, p. 10)

Weinberg states, in his seminal work *The Psychology of Computer Programming*, "that great strides are possible in the design of our hardware and software too, if we can adopt the psychological viewpoint" (Weinberg, 1971, p. vii). The focus of this section is to capture the ways in which hardware and software engineering have been portrayed in the literature. From this literature review, a preliminary model of the technological frames of hardware and software engineers can be developed. This model can then be used to explore the results from the empirical case study.

Hardware Development

As described in the introduction, the historical progression of hardware and software is one in which the rate of hardware performance improvement has outstripped that of software by orders of magnitude. According to Brooks, "the anomaly is not that software has been so slow in its progress but rather that computer technology has exploded in a fashion unmatched in human history" (Brooks, 1995, p. 254). "The up-front design costs are so high that it's no longer worth it to develop special-purpose micros for every machine-tool and garage-door opener" (Hargrave, 1996, p. 109). As a result, the burden of development has been shifting to the software arena where the expectation is that software can take care of it. Today, hardware is so sophisticated and multi-purpose that the true challenge lies in developing software that can use it effectively.

Hennessy and Jouppi state that many of the dominant ideas of computer architecture and hardware today are old ideas that are resurfacing because of the new ways in which technology has allowed us to look at hardware (Hennessy and Jouppi, 1991, pp. 18-19):

> *The field of computer architecture has become quantitatively oriented, with comparisons driven by performance and cost. Thus, computer architecture is becoming more engineering and less art.*
>
> *This shift has not led to a dramatic increase in the number of revolutionary new ideas. Indeed, many of the dominant ideas in computer architecture in the 1980s were old ideas. For example, simplified load/store instruction sets go back to the early supercomputers at Control Data Corp. and Cray Research, while the emphasis on pipelining goes back to machines like Stretch.*
>
> *However, computer architecture remains challenging because the changing implementation technologies constantly alter the trade-offs, leading to reevaluation of older ideas or the adaptation of existing ideas to a new set of technology assumptions. Dramatic changes in technology, such as significant increases in integration level and decreasing memory cost, have been crucial to the development of many new architectures.*

As recently as February 17, 1996, *The Boston Globe* described MIT scientists lamenting over the fact that an IBM computer based on "brute force" processing was markedly better at chess than their machines based on elegant uses of artificial intelligence. The mechanism used by the IBM computer "is the same 'brute force' computing that has dominated the computer world for 50 years. In computing, brute force means simply amassing all the memory and speed possible -- which is considerable, given the computer industry's dazzling record of doubling memory and speed every 18 months -- and using it to crunch millions of options in the blink of an eye" (Yemma, 1996, p. 1).

Such statements indicate much of the current sentiment toward hardware development as seen in the literature. Hardware development has reached a level where it is no longer viewed as an art form but rather as a quantitatively oriented engineering discipline. Trade-offs in hardware design can be made on the basis of principles and rules rather than instinct or beauty.

Software Development

The process of developing software for computer systems has been likened to many things, most of them non-technical in nature:

> *The process of preparing programs for a digital computer is especially attractive, not only because it can be economically and scientifically rewarding, but also because it can be an aesthetic experience much like composing poetry or music.* (Knuth, 1968, p. v)
>
> *Computer programming is a human activity. One could hardly dispute this assertion, and yet, perhaps because of the emphasis placed on the machine aspects of programming, many people -- many programmers -- have never considered programming in this light. Among programmers, there is a certain mystique -- a certain waving of the hands which takes place wherever one tries to probe the manner in which programming is done. Programming is not done in a certain way, they say, it is just done. Either you can program or you cannot. Some have it; some don't.* (Weinberg, 1971, p. 3)
>
> *Programming, like music, blends esthetics and technology. The high-level plan, the middle-level concepts, and the low-level details must be correct and in harmony with each other. Discordant data structures or missed notations are jarring.* (Soloway, 1986, p. 1)

As can be seen, much literature describes software developers as viewing their profession as an art, to be practiced as they see fit. Accordingly, pure software development efforts are not meant to be structured or strictly managed if true innovation is to result. Structured policies are described as acceptable for traditional software products (e.g. the development of the world's millionth accounts payable program), but are not to be applied to more sophisticated development efforts.

Constantine describes the delicate nature in which managers must deal with "software cowboys" and "lone wolves" who program in sleepless fury without the constraints of management or data flow diagrams (Constantine, 1995, p. 47). DeMarco and Lister debunk discussions of a future in which software development will be automatable:

> *This is another variation of the high-tech illusion: the belief that software developers do easily automatable work. Their principal work is human communication to organize the users' expressions of needs into formal procedure. That work will be necessary no matter how we change the life cycle. And it's not likely to be automated.* (DeMarco & Lister, 1987, p. 27)

Overall, the literature portrays software development as much less disciplined and much more flexible and dynamic than hardware development.

Hardware and Software Management

Traditional project development and management leaves software design and completion as an "afterthought" to the hardware design process (Mittag, 1996, p. 37). As a result software engineers are prone to believe that hardware engineers are inflexible and tend to push their problems onto software to fix. Furthermore, the very nature of the traditional development schedule results in software being the only obviously late deliverable. It is expected that software can make up for defects in the hardware design, and that if corners are to be cut, they must be cut in the software development phase, simply because it is the last phase of the cycle.

Ideas of hardware and software codesign have been gaining favor as it has becoming increasingly clear that the final activity of traditional systems design schedules involves writing and debugging the software. The purpose of codesign is to make the activities of software and hardware development parallel so as to eliminate the "waiting for hardware" gap seen in the traditional project schedule. Such ideas, although espoused in the literature, are not often seen in practice.

Organizational Transition

This thesis addresses how organizations can learn to make effective transitions as shifts in focus between hardware and software become necessary. The delivery of computer systems that are both on budget and on schedule is a serious problem (Brooks, 1995, p. 5). Difficulties exist in accurately estimating development costs and schedules and measuring project progress. Furthermore, little thought is given to software maintenance during the development cycle, even though maintenance is reported to consume anywhere from 40% to 75% of the software effort (Vessey, 1983, p. 128). The trend in development costs is that software is accounting for an increasingly greater share of the costs involved in the development of computer systems (see Figure 1). With the problems of building computer systems identified and well-documented in the literature, attention at the organizational level is required in order to mitigate their negative impact on product development.

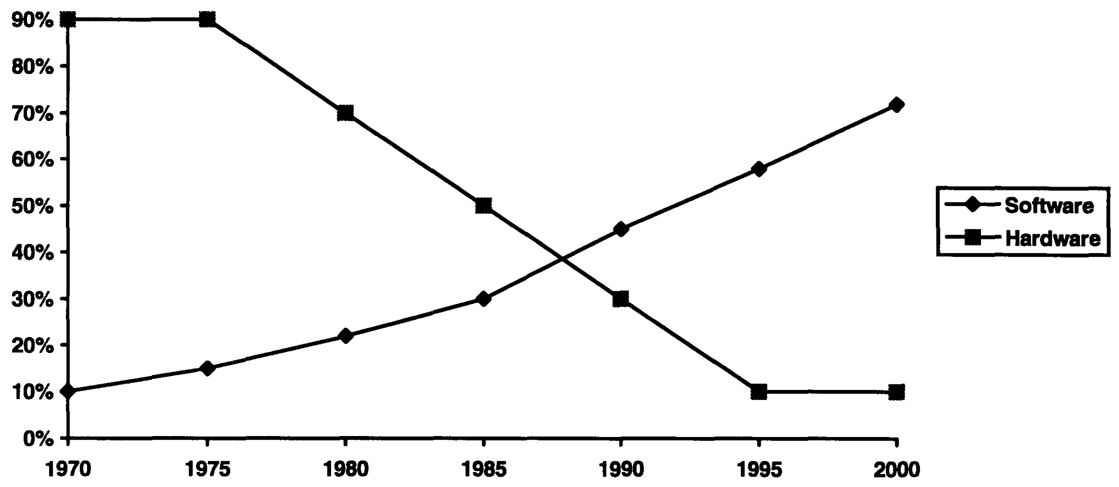


Figure 1: Hardware and Software Design Costs in Embedded Systems (from Hargrave and More, 1996, p. 110)

This thesis provides an attempt at identifying mechanisms for realizing many of the changes that are needed in order to maintain organizational control over computer systems development efforts. Specifically, the premise is that by identifying and influencing the frames through which hardware and software developers and managers approach their roles within the organization, change can be affected. Attempts at changing the fundamental ways in which organizations operate and develop products will often meet with resistance. Therefore, it is imperative that any attempts at change management be conducted with respect to both the economic and sociological impacts that they will have on the organization.

RESEARCH METHODS

This chapter provides an overview of data collection and analysis methods used in this study as well as a description of the site at which the study was conducted.

Description of the Site

The study was conducted at a small computer hardware development firm, henceforth known as Gamma Corporation (a pseudonym). Gamma Corporation is a large independent supplier of hardware products for large-scale computer systems, and it generates annual revenues of approximately $50 million. The firm employs 60 employees at its main development headquarters with 20 salespeople distributed globally. The story of Gamma is one of a company traditionally oriented towards developing hardware confronted with a large software development effort on its hands. The result was software that was not maintainable or reliable, and a company in which the social relationships among and within project teams had significantly deteriorated.

For 26 years, Gamma had developed hardware products for large-scale computer systems. These products were purely hardware endeavors completed by small teams of hardware engineers in a number of months. In 1992, Gamma decided to enter a new multi-billion dollar market with a new product, ProdX (a pseudonym). ProdX represented the first large-scale product that required a significant software development effort.

Gamma Corporation uses a matrix management strategy for creating project teams. The engineering organization consists of the following departments: Advanced Development (three individuals), Hardware Engineering (eight individuals), Software Engineering (eight individuals), Engineering Support and Validation (five individuals), and Mechanical Engineering (three individuals). Project teams are created with a team leader and personnel from Advanced Development, Hardware Engineering, and Software Engineering. Teams on average consist of five individuals, and these individuals may switch between teams on an as needed basis. Mechanical Engineering and Engineering Support/Validation provides support to all teams.

Outside of engineering, the other departments at Gamma Corporation are: Design Assurance, Manufacturing/Production, Field Service, Sales/Marketing, and Test Engineering. These departments provide support for all released products from engineering.

Data Collection

Interviews were conducted with 15 employees of Gamma Corporation, where the background and positions included senior managers, managers, engineers, technicians, and marketers. The interviews were done in two rounds, with the initial round being more open-ended than the second round. The intent was to take experiences, perceptions, and other reactions from the initial round, analyze them, and then use them to derive more structured interview protocols for the second round.

The following guidelines were provided to the interviewee before the interview:

- The name of the company will remain anonymous in this study.
- The name of the interviewee will remain anonymous in this study.
- The interview is voluntary. The interviewee may refuse to do the interview.
- The interviewee may refuse to answer any questions in the interview.
- The management of the company will have no access to the data collected.

The specific interview protocols are presented in the Appendix.

Project data were also collected and analyzed to illuminate the ways in which hardware and software projects are perceived at the managerial level. Although project data were not the primary data used in this study, they provided valuable information to supplement the data generated through interviews. Specifically, development schedules, where available, proved very useful in discussions involving assumptions about how hardware and software engineers work and are expected to work.

Data Analysis

The data were analyzed using a combination of inductive and deductive techniques. Strauss and Corbin (1990) discuss the techniques of open coding and axial coding. Open coding involves a coding of the data that is derived from the data itself, with no pre-existing categories defined beforehand. Axial coding provides pre-existing data categories that the investigator can use before the data are analyzed. The analysis of the data from this case study of Gamma Corporation involved both techniques.

The literature already suggests concepts of technological frames and mindsets which are clearly applicable in this particular research. Given this, categories already exist for analyzing and grouping data according to some of these themes (deductive). In this case, axial coding was utilized to analyze the data.

However, much of the data that were collected concerned the specific relationships, congruencies, and incongruencies between hardware and software frames within Gamma, and these themes were developed using grounded theory (inductive). Here, techniques of open coding proved very helpful.

First round interview data was analyzed using an iterative approach. The data were read repeatedly to identify common themes that were suggested by prior work as well as emerged from the data. New thematic categories were not created unless they were corroborated by multiple individuals in the study.

Once a set of thematic categories was obtained, a more specific second round interview protocol was developed with the intent of strengthening and expanding upon the themes revealed in the first round. Second round interview data were read with the first round categorizations in mind. Categories that were not supported by new data were eliminated and where applicable, new categories created that more accurately supported the data. Certain participants in the study were asked for additional input on various issues in order to refine the categorizations and interpretation of results. These results are discussed in the following chapter.

RESULTS

The results chapter discusses the themes that characterize the data collected at Gamma Corporation.

Gamma Corporation Background

The setting of this research is Gamma Corporation, the largest independent supplier of hardware products for large-scale computer systems. This section provides background regarding the nature of work that occurs at Gamma, and offers a view of the attitudes (and accompanying problems) relating to hardware and software development at Gamma.

Management Style

The Gamma management style of project development teams emphasizes results versus planning. Code is generally developed without design reviews or walkthroughs, and module design and size decisions are left to the discretion of the individual developer. Milestones are established every few weeks by polling the developers as to what they believe they can accomplish, and adjusting that by the targets expected by engineering upper management. A milestone is declared complete when the basic functionality expected is demonstrated. Little thought or attention is given to the maintainability of the existing code.

Few upper managers have software experience, and therefore tend to view projects through the eyes of hardware development. My perception is that buy-in from upper management of process changes will be more difficult to achieve than buy-in from the developers or project managers. Gamma upper management is reluctant to place strong engineers as full-time managers of technical projects, because strong engineers are in short supply and new engineers require time to train. As a result, most managers at Gamma have little "hands-on" software or hardware development experience.

Projects

Only four software projects have ever been undertaken at Gamma. Each of these used C as its high level language with certain modules written in assembly language. One project (ProdW) has been delivered, although a team is still working on upgrades, enhancements, and maintenance. Another project (ProdX) is in beta testing with limited shipments to customers. Some development is still underway on ProdX, and a follow-on project to ProdX, ProdX+ entered development in October 1995 and has just recently been put on indefinite hold for lack of resources. A third project (ProdY) was canceled after two years, recently revived under a new name (ProdZ) and then subsequently canceled again. The cancellations occurred because the product never demonstrated acceptable functionality.

ProdX and ProdX+ both used automated scheduling packages to generate schedules and set development milestones. Copies of these schedules were obtained and analyzed (see below). Even without analysis, it is well-known within Gamma that ProdX consistently missed milestones and underestimated the size of the software task at hand.

Types of Hardware and Software Being Developed

As background, it is important to make clear the types of software and hardware development that the individuals at Gamma Corporation are involved in. The nature of hardware and software is such that there is a wide spectrum along which a given project can fall. The employees of Gamma Corporation are experienced in developing hardware and software at both extremes of the spectrum, but have little experience in developing hardware and software in the middle areas of the spectrum.

The hardware systems traditionally developed at Gamma are board-level memory boards. These boards are plug-compatible with existing memory boards made by other vendors. Each board design typically involves one or two hardware engineers. Such hardware can be tested with the philosophy that "if it's broke, it's broke." For example, typical tests involve continuously writing and then reading back memory locations to ensure data integrity. Other tests require passing pre-defined verification tests and diagnostics that have been standardized and developed on the host computer system independent of Gamma's development. Passing these verification and diagnostic tests provides a very good indicator of whether the hardware system has met its specifications. This mode of hardware development represents one extreme of the spectrum, because other types of hardware development do not exhibit the single function rigidity displayed by Gamma hardware projects. These other classes of hardware are more flexible in what is required as input and expected as output and can be re-programmed quickly to modify functionality within a given range of constraints.

The software developed at Gamma Corporation is software for real-time embedded systems. This software is tailored to proprietary hardware developed at Gamma Corporation, and its purpose is to emulate and be compatible with proprietary hardware and software developed by other vendors. Because the other vendors' hardware and software are proprietary, Gamma Corporation has been forced to reverse engineer the designs of other vendors in order to provide compatibility. These reverse engineering efforts are difficult to complete with full confidence given the complexity of other products. As a result, assumptions made in the reverse engineering specifications and built into the software must be modified as new issues are uncovered during software development and testing. Such conditions lead to functional requirements that change frequently as the software is developed.

This mode of software development can be seen to be at one end of the hardware/software development spectrum. Although changing requirements are experienced in any software project, the process of reverse engineering propriety interfaces and the issues that arise as a result, make the challenges faced by Gamma software developers particularly difficult and unpredictable.

Hardware and Software Teams

Software development teams at Gamma Corporation can best be characterized as "collaborative problem-solving teams" (Constantine, 1995, p. 73), that is:

> *The aim in such a group is to hang loose and talk things through so that competing goals can be integrated and alternative approaches can be synthesized. They are, in a sense, continually reinventing themselves, changing the way they work to fit the needs of the moment and the group's long-term goals. Who is "in charge" and how they are in charge depends on what the group is doing.*

Software teams generally consist of approximately five individuals, all of whom are very technically involved with the project. Of these individuals, one is designated Project Leader, and is therefore the one "officially" designated to coordinate with other areas of the company. However, this role is not strictly implemented, and other team members (whether encouraged or not) also take on such a coordination role as the project progresses and evolves. Meetings and discussions are held frequently and are used to exchange ideas about new design issues and required modifications.

Hardware development teams at Gamma Corporation are best described as "breakthrough teams" (Constantine, 1995, p. 66), that is:

Breakthrough teams actually depend on individual initiative to coordinate their activities. Decisions are not centralized but are made independently, close to the action, by whomever encounters the problems and has the knowledge to resolve them. What keeps such a group on course is a kind of friendly competition; what keeps them from running off in every direction at once is their common interest in and love of the game and their mutual respect for each other as players.

The hardware teams at Gamma are usually smaller than the software teams, with no more than three individuals working on a given hardware product. A high level manager presides over these teams and does not interact as closely or as technically with the team as the Project Leader does with the software team. Work proceeds in a more methodical and orderly manner, with much less time spent in meetings and discussions and much more time spent in the lab areas doing development and debugging.

Hardware and Software Mindsets at Gamma Corporation

Both hardware and software mindsets were defined earlier as the underlying assumptions, expectations, and knowledge that individuals have about hardware and software, respectively. At the outset of this study it was expected that the frames or mindsets that individuals had about software and hardware, would be quite different depending on whether they were involved in software or hardware development. However, this study found that the mindsets that Gamma members had about engineering approaches to software and hardware were quite similar, although they distinguished between hardware and software project development, and saw these as needing to be quite different.

Approaching Hardware and Software Problems

Overwhelmingly, it was acknowledged at Gamma Corporation, that although stereotypes of hardware and software engineers do exist, the fundamental qualities necessary to be good at either or both are in actuality the same.

> *My overall belief is that good engineers are good engineers because of the talents they have and their approach to problem solving, and in general that applies to most engineering disciplines.* (Manager)
>
> *My opinion is that there are definitely stereotypes and some people fall into that category, but the true engineer I think reasons things out the same way.* (HW Engineer)
>
> *When you're able to look at both of these as one entity instead of two, then you truly have an engineer.* (SW Engineer)
>
> *People that are good at it are people who have learned there's a lot of technique. I guess I'd say that's involved with people who are good at either software or hardware designs.* (Manager)
>
> *You gotta be able to abstract [to excel at hardware or software development]. If you can't abstract, like I said, just get out of here.* (SW Engineer)

The findings at Gamma Corporation indicate that there is a common belief in an "engineering discipline" that individuals share regardless of whether they are developing hardware or software. These results run contrary to stereotypical notions of the individuals required to develop hardware and software. Such stereotypes paint software engineers as renegade artists and cowboys while portraying hardware engineers as clean-cut robots.

> *Software engineers are expected to be in at midnight wearing jeans and hardware engineers show up during the day in white shirts and suits.* (Manager)

The results of this study do not support these stereotypes -- at least at Gamma Corporation -- from the purely engineering approach. However, such stereotypes may be evident in the ways in which software and hardware projects are managed, and the kind of practices that are reinforced. This topic is covered in the next section.

Characteristics of Hardware and Software Projects

Although there seemed agreement that, from an engineering perspective, the approaches to hardware and software development problems at Gamma Corporation are quite similar, it was also evident that from an organizational perspective the management of hardware and software projects at Gamma is very different. Based on the interview data, hardware and software projects were characterized as differing along five dimensions: predictability, functionality, testability, complexity, and changeability.

Table 1 presents interview data from individuals at Gamma Corporation highlighting the differences in hardware and software projects discussed below.

Predictability

Members of Gamma described the development of hardware as considerably more predictable and controllable than software. Experience in the hardware development business had enabled engineers to predict hardware complexity based on simple metrics (e.g., board size). No such metrics had been generated for the software side. Software estimation methods -- e.g., COCOMO (Boehm, 1981) -- provide little insight into the time required to develop the complex, embedded software products that Gamma Corporation produces. Software estimation methods are more appropriate for estimating projects that can be comprehensively defined before implementation. Such thorough specification was impractical for the types of software systems being developed at Gamma Corporation.

Functionality

The intended functionality of hardware developed at Gamma was generally well-defined. Software, however, was perceived to be very functionally adaptive. Multiple generations and variations of software products at Gamma Corporation had been based on the same hardware platform, with the assumption that software can be quickly adapted to many different variations and functions, while hardware cannot. Any features not present in the hardware were assumed to be capable of being rendered in software. Hardware was often described as "special-purpose," whereas software is more "general-purpose."

Gamma Corporation's ProdY (later renamed as ProdZ) went through three years of software iterations on the same hardware platform before it was recognized that the lack of functionality provided by the hardware could not be compensated for in software. Instead of modifying the hardware, the software project was canceled.

Testability

Members of Gamma saw software as considerably more difficult to test robustly than hardware. Hardware was described as having more of a pass/fail quality than software. For example, the hardware boards at Gamma Corporation were placed in in-house testers overnight. After a night of running, a board would be declared to be either pass or fail. Similarly, in the design stage of hardware, simulators were available for testing the hardware design before it was implemented. No such simulators were available for testing software before its implementation.

With products having a large software component, tests at Gamma Corporation typically ran for weeks, with engineers attempting to simulate customer environments and situations. No purely procedural tester had been developed, and neither was such a tester possible. It is typically impossible mathematically to test all possible code paths in a given piece of software. Even conditions which on the surface appear to be software "bugs," may not be so clear -- bugs may be later re-labeled as "features" or attributed to "user error."

Complexity

The tooling provided for software and hardware projects at Gamma were at quite different levels of sophistication. In general, hardware development was supported by increasingly sophisticated tools that guide and support the engineer through all phases of the development life cycle. These tools significantly reduce the complexity faced by the hardware developer. For example, powerful tools for up-front design and simulation are available that can significantly reduce the likelihood of error in the final hardware product. Such tools have tended to structure the entire hardware development process in ways not yet available for software. Hardware engineers were generally not as familiar with the less sophisticated software processes and tools that do exist. Traditional software development tools include CASE (computer aided software engineering) tools, source code control, configuration management, and documentation tools. These tools, although helpful, have not yet reached the level of sophistication provided by hardware tools and familiar to Gamma engineers.

Changeability

Once a hardware board had been "laid out" or sent to "fab" (fabrication), future modifications at Gamma were met with considerable resistance. Such resistance was the result of a perceived difficulty in re-doing or correcting hardware designs. The validity of such perceptions, given the quick turnaround and flexibility seen in modern hardware development efforts, is questionable, yet the perception continued to exist within Gamma. Given this resistance to change on the hardware side of development, considerable effort was expended to ensure that the original hardware design was sound. As noted earlier, sophisticated tools for simulation and verification were used by the engineers before any physical hardware was ever touched or assembled.

Hardware functionality was only changed when it was broken or when it had been designed in such a way that the very life of the product was threatened. Gamma Corporation used a complex ECO (engineering change order) process for applying changes to hardware functionality. Kooshian (1995, p. 1) describes such as process as follows:

> ***Hardware is designed with a rigorous, well-known, religiously implemented process supported by high-quality, robust, and expensive tools. The software, on the other hand, is often coded up late in the game by a bunch of smart people, using whatever tools they can get their hands on.***

The hardware project management process at Gamma involved many phases including release documentation, schematics, and bills of materials. There was no equivalent project management process defined and in place for software. Because software was intangible and less visible, little attention at Gamma had been paid to formalizing its release and control throughout the company. With formal release and control procedures in place for hardware and not software, it was much easier at Gamma to change software rather than hardware in all aspects of the production schedule.

It was clear from the interviews conducted at Gamma Corporation that the individuals recognized and understood the key differences in hardware and software projects. However, it was also clear that as an organization, Gamma Corporation had failed to recognize these differences, and had not acted on them as it made the transition from a hardware to software company.

	Hardware	Software
Predictability	Things are a little more controlled in an exclusive hardware environment...processes are a little more disciplined and rigid. (HW Engineer) We didn't always hit the mark [on hardware projects]. But, we weren't that far off, and when we didn't hit the mark, we had an idea of what would get us to the end. (Manager) Mechanical stuff is predictable when you have to deal with it so much. (SW Engineer) Hardware development has well-defined goals. (SW Engineer)	The environment in which hardware development takes place is much more structured and tends to proceed in a much more controlled fashion. (SW Engineer) Software tends to be more "loosey goosey." You only know just how tough things are when you get into it. You see things and think you need two or three weeks to do it and in three or four days it is done. Other things you say two weeks and two months later you are still grinding away at it. (SW Engineer)
Functionality	Hardware has a dedicated function, what it does and how it does it. (HW Engineer) I think our hardware is fairly rigid; it has to do ABC to work. (Marketer)	The software, you have so many different variations and take offs. (HW Engineer) Software can be extremely adaptive. (SW Engineer) [Software must cover] all the areas that the customer might invoke, all the different scenarios and how the customers might use the equipment. (Marketer)
Testability	In hardware if the piece is broken it's broken. (HW Engineer) There were probably a lot fewer variables that went into either developing the product, testing it, or debugging it. (Manager) There are some very good tools for both validating your [hardware] design before you commit to actually generating physical hardware. (SW Engineer)	Things aren't tested because the pass/fail isn't as it is on the hardware side. (HW Engineer) With the many combinations that are possible, it becomes impractical to test in an effective manner all of the paths of your software portion of the product. (SW Engineer) There are no real ways to validate [software], prior to writing code. (SW Engineer)
Complexity	Ask anyone who does hardware what they want to build, complexity wise you know a board 11 by 8. If you look at that with a rough idea of complexity I can almost tell you how long it will take to design. (SW Engineer) We kind of find that there aren't very many complex hardware designs happening anymore. (SW Engineer) In general, hardware is simpler than software. (SW Engineer)	I think writing software is probably a lot harder. (Marketer) Hardware development has been in a sense less complex...its easier to decompose and document a pure hardware program than it is a system or software program. (Manager) However you would like to measure complexity, the software in a large system is a lot more complicated than the hardware. (SW Engineer)
Changeability	It's a lot harder to change hardware. (Manager) Hardware to date has not been absolutely flexible so it has a tendency to impose restraints on the software. (SW Engineer)	So you have to make your [software] designs very flexible. (Marketer) You get the advantage on the software side that software can be extremely adaptive. The presentation that is given to the customer [is that it] is hypothetically possible for it to be extremely adaptive or changeable. (SW Engineer) Software problems are only defined as the development proceeds. (SW Engineer)

Table 1: Differences in Hardware and Software Projects

Management Practices at Gamma Corporation

At Gamma Corporation, there was no clear organizational understanding of the organizational changes required in order to adapt to the different nature of the products that were being developed. The management structure and philosophy at Gamma Corporation remained constant throughout the entire transition to software development, although very different products were being developed than had been developed in the past. The data shown in the previous sections highlight that for individuals at Gamma, the approach and skills required for software development were seen to be the same as those required for hardware development. However, as the scope is enlarged to consider project management, it is apparent that certain process changes were required. A lack of understanding of the new processes that were required to transition a primarily hardware company to a more software oriented one have created difficulties at Gamma Corporation.

> *In that sense Gamma was making the transition or we are attempting to make the transition to a new type of company. But, maybe in a sense we are even worse off than a start-up in that we didn't start off with the key people who really understood what it was that we were trying to get into. So that probably as much as anything -- if you look back -- was probably the cause for some of the problems that we have been encountering.* (Manager)

As Kooshian (1995, p. 3) notes: "It is unlikely that the manager of the software group in a hardware-oriented company can solve any of these problems without significant support from senior management." When Gamma members were asked whether the current culture at Gamma Corporation was suitable to the type of products it was trying to deliver (i.e., primarily complex software products), their responses were quite consistent:

> *No, not at all. The company isn't managed properly at all.* (HW Engineer)
>
> *No. I think the corporate culture on one hand was not geared toward developing this type of program.* (Manager)

No. The culture that exists at Gamma is very appropriate for the company that it used to be. (SW Engineer)

In general, affecting the proper organizational changes required to make a shift from a hardware orientation to a software one is difficult. However, the situation at Gamma Corporation was particularly difficult as both the hardware and software development efforts were characterized as extreme, and little to no attempt was made organizationally to change the development processes and practices as the transition was undertaken. This finding is further reinforced by the data derived from examining Gamma's project schedules.

Project Schedules

I've never done a schedule that was worth anything. (Manager)

In addition to the data generated through individual interviews and observation, project schedules were obtained from Gamma Corporation and analyzed for insights into the management practices at the company.

Analysis of such project data illuminates the ways in which hardware and software projects were perceived by Gamma management, and they reveal some of the preconceived notions dominant at Gamma as to how hardware and software engineers should work on hardware and software projects.

Figure 2 depicts a generalized view of a typical embedded systems project schedule involving both hardware and software development.

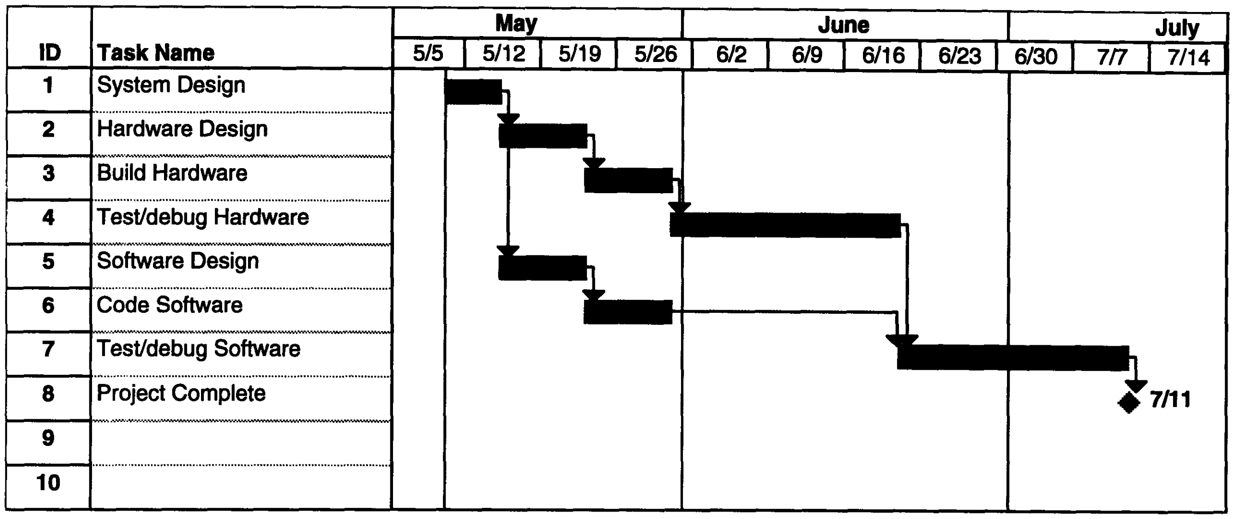


Figure 2: Typical Software / Hardware schedule (from Mittag, 1996, p. 37)

As Mittag (1996, p. 37) observes:

> *The most telling point about this schedule is the order of operations. There is a fairly significant amount of overlap at the beginning of the project, but in general there isn't much going on in the software department between the time the initial code testing is complete and when software testing begins. The slack time is more than made up at the end of the project, however, where all the work is being done by the software group. In other words, it's the software group's fault if the project is late because they were the last to work on it.*

This typical schedule reinforces many of the stereotypes that exist concerning hardware and software development. Most prominent is the belief that software development can compensate for faults present in the hardware and is therefore inherently in the critical path to product release.

An examination of project schedules at Gamma Corporation project schedules indicates that Gamma had fallen into this and other, more pernicious traps. As one engineer at Gamma stated:

> *On any project, software engineers will be the last ones working on it. It's the nature of the animal that you are going to be doing software last.* (SW Engineer)

Analyzing these traps provides insight into problematic management practices that have led to persistent difficulties in Gamma's product development and delivery. The first project examined -- ProdX+ -- shows the evolution of a schedule from October 1995 to January 1996 (see Figures 3 and 4). (Note that the terms used in the schedules have been modified to preserve confidentiality. All dates have been left unchanged.)

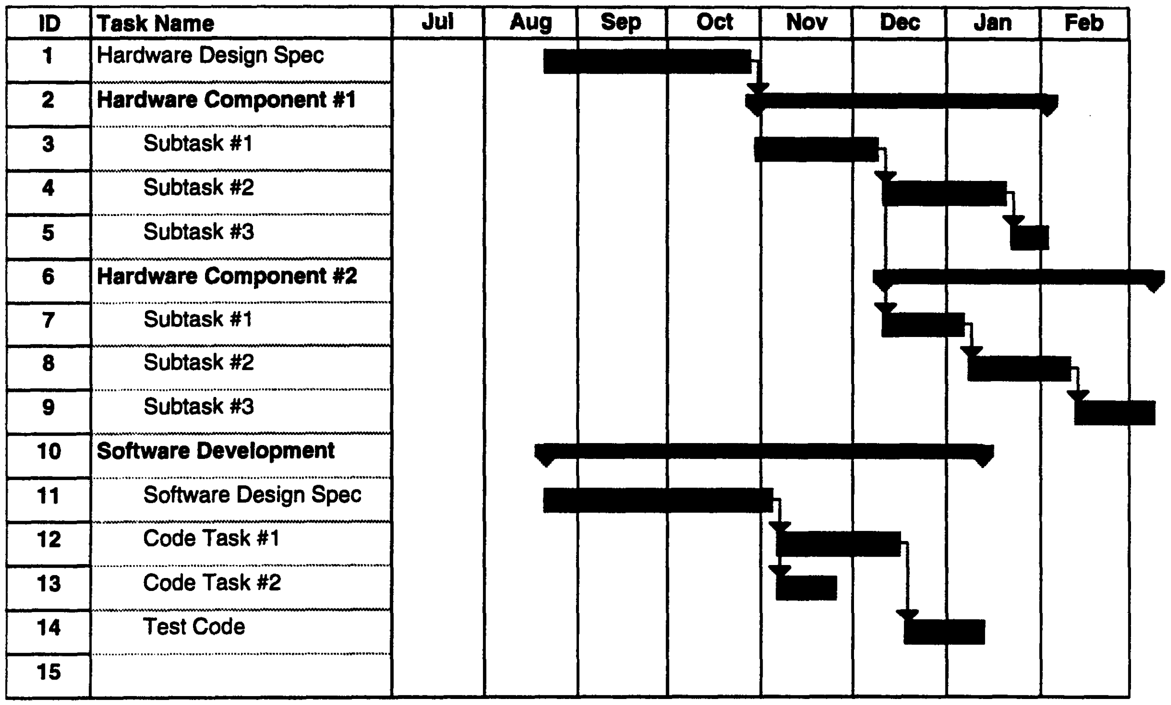
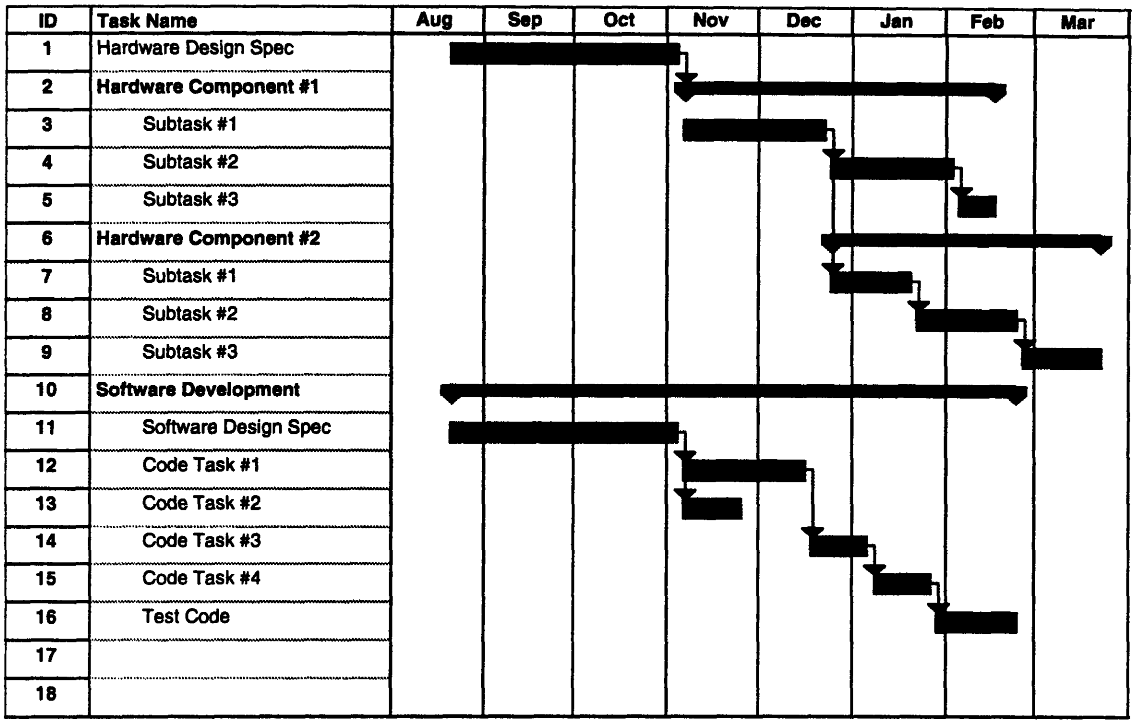


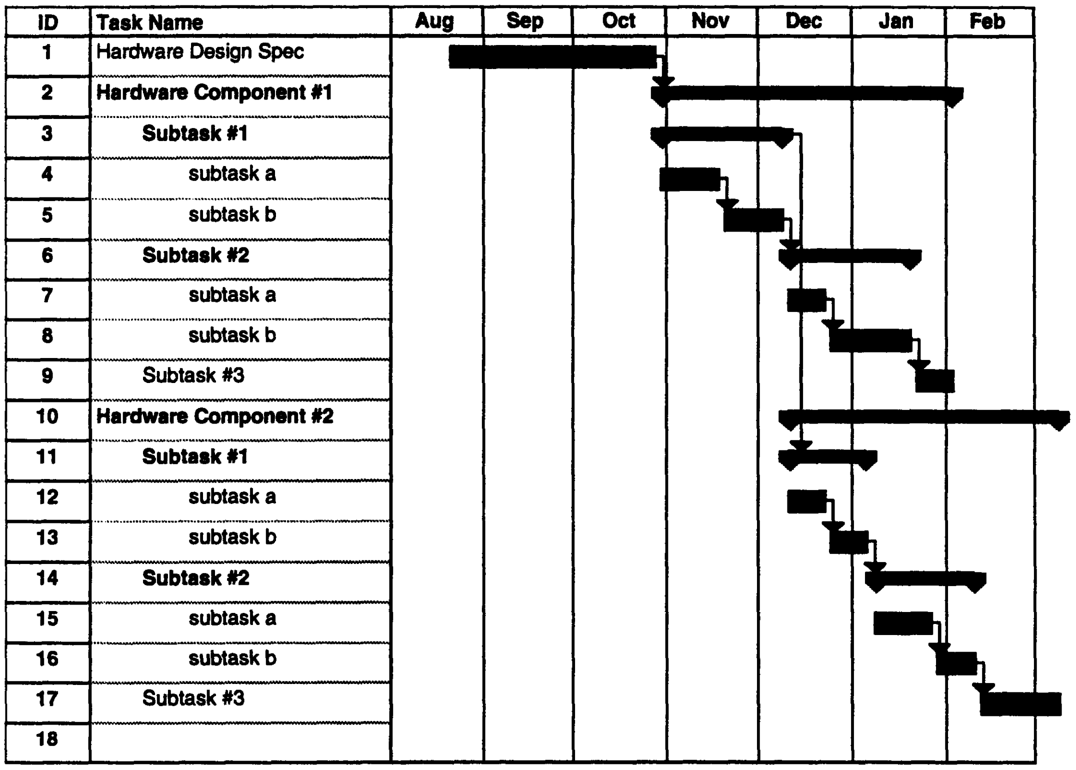
Figure 3: Gamma Corporation ProdX+ Schedule, 10/5/95

The level of detail provided in the schedule varies considerably between the hardware and software sections. In fact the schedule was actually presented to engineers as two separate schedules on October 5, 1995 (see Figures 3 and 5). The first schedule is best described as an overview of the project and includes three major divisions: two hardware development sections, and a software development section. The second schedule is a more detailed presentation of the hardware development schedule provided in the overview schedule. The inclusion of a second schedule devoted exclusively to the



ID	Task Name
1	Hardware Design Spec
2	**Hardware Component #1**
3	Subtask #1
4	Subtask #2
5	Subtask #3
6	**Hardware Component #2**
7	Subtask #1
8	Subtask #2
9	Subtask #3
10	**Software Development**
11	Software Design Spec
12	Code Task #1
13	Code Task #2
14	Code Task #3
15	Code Task #4
16	Test Code
17	
18	

Figure 4: Gamma Corporation ProdX+ Schedule, 1/23/96



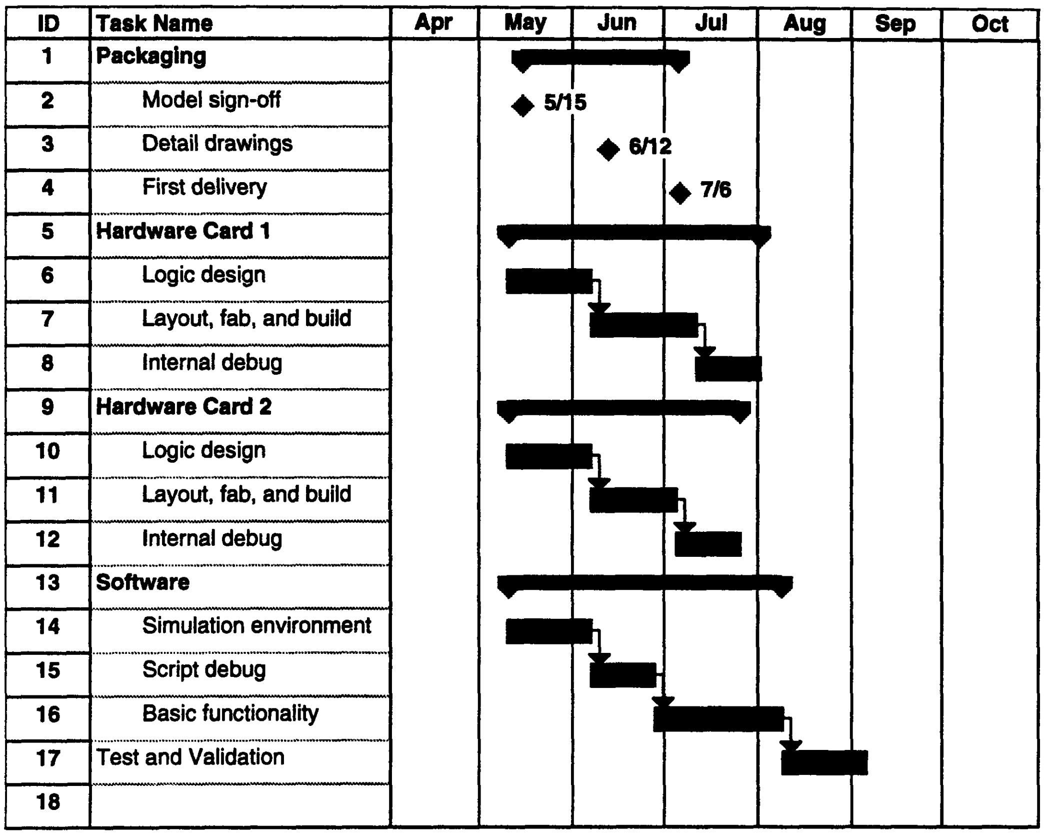
ID	Task Name
1	Hardware Design Spec
2	**Hardware Component #1**
3	**Subtask #1**
4	subtask a
5	subtask b
6	**Subtask #2**
7	subtask a
8	subtask b
9	Subtask #3
10	**Hardware Component #2**
11	**Subtask #1**
12	subtask a
13	subtask b
14	**Subtask #2**
15	subtask a
16	subtask b
17	Subtask #3
18	

Figure 5: Gamma Corporation ProdX+ Hardware "Micro"-Schedule, 10/5/95

hardware portion of the project with no such schedule provided for software development, provides a strong indication of the comfort (or discomfort) that management at Gamma had with estimating and controlling the software development process.

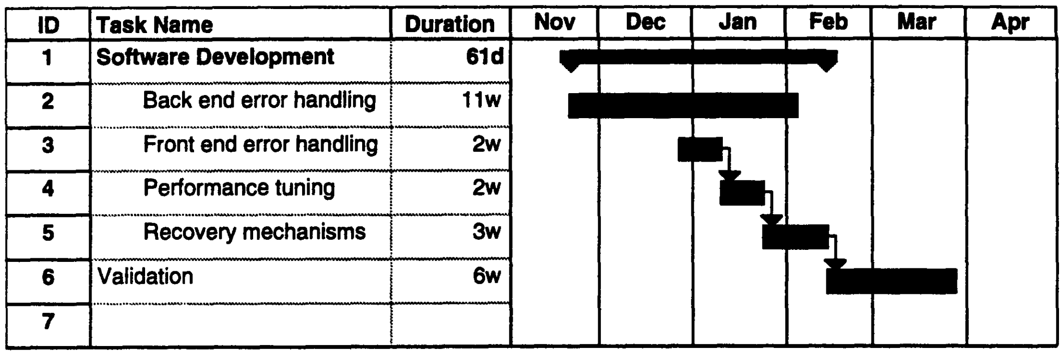
As the schedules evolved over the months, it is seen that the hardware portion slips two weeks over a period of four months whereas the software portion slips six weeks over the same time period. In fact, the final overview schedule on January 23, 1996 (see Figure 4) included handwritten dates in place of the formatted ones presented in the early schedule -- an indication of the waning confidence the project manager had in estimating dates as time went on. At the time of this writing, the project (ProdX+) for which this is the schedule has been put on indefinite hold for lack of resources.

The second set of schedules analyzed provided more insight into the management culture at Gamma Corporation. The project schedule shown in Figure 6 represents the first large-scale hardware and software project ever undertaken at Gamma. This initial master schedule (on June 7, 1994) was primarily devoted to scheduling hardware development, with only about a quarter of the items on the schedule relating to software development. The initial schedule set a completion milestone of September 15, 1994, and assumed that software development would occur concurrently with software development. As it turned out, the hardware portion of the product was completed (albeit one to two months late) while the software development continued into March 1995. A later schedule (see Figure 7) was entirely devoted to software and had a completion date in February 1995. It shows all of the software portion shifted from May-August 1994 to the new time period following hardware completion, November-February 1995. The original schedule developed in June of 1994 was virtually devoid of any preparation or understanding of the nature of the software effort involved.



ID	Task Name	Apr	May	Jun	Jul	Aug	Sep	Oct
1	**Packaging**							
2	Model sign-off							
3	Detail drawings							
4	First delivery							
5	**Hardware Card 1**							
6	Logic design							
7	Layout, fab, and build							
8	Internal debug							
9	**Hardware Card 2**							
10	Logic design							
11	Layout, fab, and build							
12	Internal debug							
13	**Software**							
14	Simulation environment							
15	Script debug							
16	Basic functionality							
17	Test and Validation							
18								

Figure 6: Gamma Corporation ProdX Schedule, 6/7/94



ID	Task Name	Duration	Nov	Dec	Jan	Feb	Mar	Apr
1	**Software Development**	61d						
2	Back end error handling	11w						
3	Front end error handling	2w						
4	Performance tuning	2w						
5	Recovery mechanisms	3w						
6	Validation	6w						
7								

Figure 7: Gamma Corporation ProdX Schedule, 1/3/95

Gamma Corporation has learned little from these project failures and slippages. An informal survey of Gamma software engineers provided a list of three reasons as to why many of Gamma's software efforts are often redone in order to reduce errors and increase efficiency and reliability:

1. Lack of time to "code it right the first time."
2. Amorphous requirement specifications.
3. Failure to enforce strict interface checking and structured coding practices.

All of these reasons point to significant problems with management practices at Gamma Corporation.

Summary of Gamma Corporation

As a case study, Gamma Corporation provides insight into how hardware and software are viewed and managed in an organizational setting. From the data, it is evident that while the developers at Gamma recognized that the company was going through a major transition, the organization had not adapted its procedures and practices to reflect this realization. At Gamma little if any changes were made to such practices as budgeting, estimating, organizing, and other standard operating procedures. As a result, Gamma Corporation was not properly positioned to transition from hardware to software production. Hence, most of the software projects at Gamma failed or took considerably longer to complete than originally estimated. Furthermore, engineers who were aware of the problems as well as upper management's failure to adapt the organization's practices became disgruntled and left. Gamma Corporation, an organization well-versed in hardware, ignored many of these problems and by default chose to manage software in the way that they had always managed hardware.

GENERALIZATIONS AND IMPLICATIONS

This chapter provides suggestions for what issues computer organizations must address to make the transitions necessary to support the changing nature of computer systems development from a primarily hardware orientation to a primarily software one.

The results of this case study have demonstrated that the approaches and skills required to excel in hardware and software development are essentially the same. However, what is necessary is a different set of processes and practices to manage the fundamentally different nature of hardware and software. In particular, processes used to manage the development of hardware are simply inappropriate and ineffectual when applied to the development of software.

What is thus needed is a focus -- not so much on understanding how individuals approach hardware and software development -- but on understanding the organizational contexts in which engineers can be expected to effectively develop software, hardware, or both. In the following, personnel as well as organizational issues are discussed.

It is recognized that the nature of the hardware development done at Gamma Corporation does not represent all classes of hardware development. Comments such as "if it's broke, it's broke" certainly do not universally apply to all hardware development efforts. Therefore, it is important to bear in mind that the following remarks are suggestive only, and generalizations beyond Gamma can only be made cautiously.

Personnel Issues

Hardware and software are characterized as different forms of engineering, with a good engineer capable of handling either or both. When transitioning from hardware to software, therefore, the fundamental "engineering" skills that are required can be carried over by existing personnel.

Although the mindsets required for effectively developing hardware or software may be similar, the processes required to effectively work on a hardware or software project are not. Certain individuals may adjust easily to the new processes introduced by a company shifting from one form of development to another. Training may be required to introduce new tools and methodologies to the engineers who will be using them. For example, one software engineer noted during my study:

> *I don't think it's a difference in the type of person [required to do hardware or software development], but rather just in training and what one has been doing for the last ten years.* (SW Engineer)

Furthermore, any repositioning of management practices for shifting from hardware to software development can have effects on individual engineers. It is to be expected that a certain number of capable engineers will find the transition too difficult or jarring to accept. Such turnover, however, generally accompanies any organizational change.

Organizational Issues

The primary areas of change are the organizational processes and practices around projects. Gamma Corporation failed to do this in shifting from hardware to software development, and, as a result, encountered numerous problems as it attempted to migrate to primarily software intensive projects.

Organizations must consider the contexts within which they ask people to work. The contexts required to effectively develop hardware versus software are very different. Furthermore, the production processes required for development in a hardware world are quite different than those required for development in a software world. When migrating from a hardware to software development environment, processes should be changed using a step-wise approach. It is important that the upper management of the organization provide the motivation for process change. Without such support, it is unlikely that the organization will make the necessary transition.

Upper management is responsible for recognizing that a hardware to software shift is occurring within the organization and communicating this awareness to the employees. Communication prepares the employees for the process changes that are to occur, and also reassures them that the organization is aware of the changing environment in which it is operating. Upper management should also require that project managers and project leaders rethink the ways in which they approach and schedule software projects, recognizing that they need to be framed differently than hardware projects (see below).

Both test and production processes -- areas not involved with the actual software development, per se -- may require the most radical change. These two areas require processes that are suitable to the increased complexity and changeability seen when moving from hardware to software products. Test and production fixtures and procedures may become instantly obsolete when the shift from hardware to software is made. Furthermore, the personnel involved in test and production may not be as flexible in their ability to adjust to new paradigms given that such areas are generally much more procedure driven than engineering development.

In summary, proper software engineering processes need to be designed to yield quality software with predictable schedules. However, the necessity for such processes on the software side needs to be recognized by management before they can be effectively implemented. Gamma Corporation fell short in that it attempted to develop sophisticated software in an organizational context that still understands development from a hardware perspective. Unfortunately, it is not clear from this study whether a proper management structure, although necessary, is sufficient to effectively allow a company to make the hardware to software transition.

Project Management Recommendations

It is important that managers have respect for the software development side as well as the hardware development side when preparing development schedules and allocating personnel on projects in which both software and hardware are used. Efforts should be made to reduce the likelihood of software being placed in the final critical path to product delivery. Software/hardware codesign is an idea that allows for parallel development of both software and hardware to help ameliorate just that problem.

The respect given to the processes and procedures required for the development and release of hardware should also be given to the processes and procedures required for the development of software. At Gamma Corporation, software releases were made when the developers hand someone a "magic diskette" with the code executable on it. On the other hand, hardware releases went through a rigorous approval and documentation process. Such a discrepancy in procedures and attitudes provides insight as to why so many of Gamma's software projects slipped or failed.

Management expectations should be adjusted when thinking of software given its likely increased complexity and lack of predictability and testability as well as its high likelihood of undergoing "function creep." Resource allocation may also need to be rethought. Whereas five years ago, two engineers in a back room were capable of producing a board-level product for Gamma in three months, two engineers in such a situation are utterly inadequate at producing the type of software required today.

Summary

The effectiveness with which an organization manages the transition from hardware to software lies with the recognition that the very nature of hardware and software are different along the five dimensions of predictability, functionality, testability, complexity, and changeability. Gamma Corporation provides an example of the problems that result when an organization does not recognize the different processes and contexts necessary for proper development of software, even when individuals understood the differing issues involved with the development of software versus hardware.

REFERENCES

Abdel-Hamid, T., and S. Madnick 1991. *Software Project Dynamics: An Integrated Approach,* Prentice Hall, Englewood Cliffs, New Jersey.

Argyris, C., and D. Schon 1978. *Organizational Learning.* Prentice-Hall, Englewood Cliffs, New Jersey.

Berger, P. L., and T. Luckmann 1967. *The Social Construction of Reality.* Anchor Books, New York.

Boehm, Barry 1981. *Software Engineering Economics.* Prentice-Hall, Englewood Cliffs, New Jersey.

Brooks, Frederick P. 1995. *The Mythical Man-Month: Essays on Software Engineering, Anniversary Edition.* Addison-Wesley Publishing Co., Reading, Massachusetts.

Constantine, Larry L. 1995. *Constantine on Peopleware.* Yourdon Press, Englewood Cliffs, New Jersey.

Curtis, Bill 1995. *Foreword for Constantine on Peopleware.* Yourdon Press, Englewood Cliffs, New Jersey.

DeMarco, Tom, and Timothy Lister 1987. *Peopleware: Productive Projects and Teams.* Dorset House Publishing, New York, New York.

Dougherty, D. 1992. "Interpretive Barriers to Successful Product Innovation in Large Firms." *Organizational Science 3*, 2, 179-202.

Douglas, M. 1987. *How Institutions Think.* Routledge and Kegan Paul, London.

Eisenhardt, K.M. 1989. "Building Theories from Case Study Research." *Academy of Management Review 14*, 4, 532-550.

Hargrave, Steve, and John More 1996. "The Increasing Importance of Software." *Electronic Design 44*, 1, 109-113.

Hennessy, John L., and Norman P. Jouppi 1991. "Computer Technology and Architecture: An Evolving Interaction." *Computer*, September, 18-29.

Knuth, D. E. 1968. *The Art of Computer Programming, Vols. 1-3.* Addison Wesley Publishing Co., Reading, Massachusetts.

Kooshian, Sarah 1995. "Managing Software Teams in a Hardware Company." *Proceedings Embedded Systems Conference*, April.

Leonard-Barton, D.A. 1990. "A Dual Methodology for Case Studies: Synergistic Use of a Longitudinal Single Site with Replicated Multiple Sites." *Organizational Science 1*, 3, 248-266.

Miles, M.B. and A. M. Huberman 1982. *Qualitative Data Analysis: A Sourcebook of New Methods.* Sage Publications, Newbury Park, California.

Mittag, Larry 1996. "Trends in Hardware/Software Codesign." *Embedded Systems Programming 9*, 1, 36-45.

Mohr, L.B. 1982. *Explaining Organizational Behavior.* Jossey Bass, San Francisco, California.

Orlikowski, Wanda J. 1989. "Division Among the Ranks: The Social Implications of Case Tools for System Developers." *Proceedings of the Tenth International Conference on Information Systems*, Association for Computing Machinery, New York, New York, 199-210.

Orlikowski, Wanda J. 1993. "CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development." *MIS Quarterly*, September, 309-340.

Orlikowski, Wanda J. and Debra C. Gash 1994. "Technological Frames: Making Sense of Information Technology in Organizations." *ACM Transactions on Information Systems 12*, 2, 174-207.

Schein, E. 1985. *Organizational Culture and Leadership.* Jossey-Bass, San Francisco.

Schumacher, E. F. 1973. *Small Is Beautiful: Economics as if People Mattered, Perennial Library Edition.* Harper and Row, New York, New York.

Smircich, L. and C. Stubbart 1985. "Strategic Management in an Enacted World." *Acad. Management Review 10*, 4, 724-736.

Soloway, Elliot, and Sitharama Iyengar eds. 1986. *Empirical Studies of Programmers.* Alex Publishing Corp., Norwood, New Jersey.

Strauss, A. and J. Corbin 1990. *Basics of Qualitative Research: Grounded Theory, Procedures, and Techniques.* Sage Publications, Newbury Park, California.

Thomsett, Rob 1980. *People and Project Management.* Prentice-Hall, Englewood Cliffs, New Jersey.

Vessey, I. and R. Weber 1983. "Some Factors Affecting Program Repair Maintenance: An Empirical Study." *Commun. ACM 26*, 2, 128-134.

Weick, K. E. 1979. *The Social Psychology of Organizing.* Addison-Wesley Publishing Co., Reading, Massachusetts.

Weinberg, Gerald 1971. *The Psychology of Computer Programming.* Van Nostrand Reinhold Co., New York, New York.

Yemma, John 1996. "MIT Brains are Feeling Blue." *The Boston Globe 249*, 48, 1-9.

Yin, R.K. 1989. *Case Study Research: Design and Methods.* Sage Publications, Beverly Hills, California.

APPENDIX A: FIRST ROUND INTERVIEW FORMAT

The following guidelines are to be provided to the interviewee before the interview:

- The name of the company is to remain anonymous for the purposes of this research.
- The name of the interviewee is to remain anonymous for the purposes of this research.
- The interview is voluntary. The interviewee may refuse to do the interview.
- The interviewee may refuse to answer any questions in the interview.
- The management of the company will have no access to the answers given in the interview.

Questions:

1. Do you consider yourself a software person, a hardware person, both, or neither?
2. How would you describe the ways in which individuals who are good at software development think versus the ways in which individuals who are good at hardware development think? How are they similar? How are they different?
3. Do you feel that Gamma is primarily a hardware or software company? (Follow up on this question depending on how it is answered.)
4. Can you describe to me the difference between working on a project that uses only hardware versus one that combines both hardware and software?
5. Is the current culture at Gamma suitable to the kind of products it is trying to deliver?
6. What is the criteria for developing reliable and robust hardware in a timely manner?
7. What is the criteria for developing reliable and robust software in a timely manner?
8. How does Gamma compare with respect to meeting these criteria?
9. Do you have any experiences or anecdotes that either highlight ways in which issues concerning hardware and software development were done right? or wrong?

APPENDIX B: SECOND ROUND INTERVIEW FORMAT

Questions:

1. Do you consider yourself a software person, a hardware person, both, or neither?
2. How would you describe the ways in which individuals who are good at software development think versus the ways in which individuals who are good at hardware development think? How are they similar? How are they different?
3. Do you feel that Gamma is primarily a hardware or software company? (Follow up on this question depending on how it is answered.)
4. Can you describe to me the difference between working on a project that uses only hardware versus one that combines both hardware and software?
5. Is the current culture at Gamma suitable to the kind of products it is trying to deliver?
6. If the culture has changed, how has it changed? What was management like before? What is it like now?
7. What are the critical people issues involved in developing reliable and robust hardware in a timely manner?
8. What are the critical people issues involved in developing reliable and robust software in a timely manner?
9. Do software and hardware people work differently?
10. Are software and hardware people evaluated differently? Are they trained differently? Are different things expected of them?
11. How does Gamma compare with respect to meeting these criteria?
12. Do you have any experiences or anecdotes that either highlight ways in which issues concerning hardware and software development were done right? or wrong?