MAPPING THE PRODUCT DEVELOPMENT PROCESS TO IT SOLUTIONS THROUGH USE MODELS

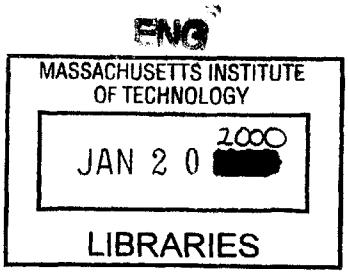
by

SAMIR PATIL

MS
University of Massachusetts, Amherst, 1994

SUBMITTED TO THE SYSTEMS DESIGN AND MANAGEMENT PROGRAM
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN ENGINEERING AND MANAGEMENT

AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
February 2000



Signature of Author: _____
ystem Design and Management Program
January 4, 2000

Certified by: _____
Dr. Daniel E. Whitney
Senior Research Scientist, Center for Technology, Policy and
Development, MIT
is Supervisor

Accepted by: _____
Thomas Kochan
hool of Management
M/SDM Co-Director

Accepted by: _____
Paul Lagace
Professor of Aeronautics and Engineering Systems
LFM/SDM Co-Director

MAPPING THE PRODUCT DEVELOPMENT PROCESS TO IT SOLUTIONS THROUGH USE MODELS

by

SAMIR PATIL

Submitted to the Systems Design and Management Program in Partial Fulfillment of the requirements for the Degree of Master of Science in Engineering and Management

ABSTRACT

Product development, especially the design of large complex systems, is a special type of business process. What makes it special is the highly coupled nature of design decisions and the large size of product development teams. It is not unusual to have several hundred multi-disciplinary members taking millions of design decisions over the life of the project. From the Information Technology (IT) perspective, the special nature of design presents a challenge of coordinating the interdependent tasks of the team and integrating them with the efforts of other engineers and designers. In general, IT solutions for product development have had far less success in creating comprehensive models of the design process than of products. One of the main reasons for this failure is that fully capturing the knowledge that goes into a complex system design is hard. This thesis explores the role of IT, specifically software tools, in improving the product development process through a framework that better represents and captures design knowledge. In order to overcome the limitations of the current implementations the concept of "Use models" is developed. A use model consists of the methods, tools, and a map of the design process. This map is in the form of a design structure matrix that represents tasks and relationship knowledge at a system level. By combining this detailed process view with tools and methods the Use model creates a better IT representation of the product development process.

Thesis Supervisor: Dr. Daniel E. Whitney

Title: Senior Research Scientist, Center for Technology, Policy and Industrial Development, MIT

ACKNOWLEDGEMENTS

I would like to thank Dr Whitney for his kind and patient guidance during the course of this research work and the writing of this thesis. Without his insights and suggestions I would not have been able to complete my work.

I would also like to thank many others at MIT who helped me during my studies. Especially Shaun Abrahamson, Steven Smyth, and Nader Sabbagian who not only provided the much needed entertainment but also were ideal audiences for bouncing off new ideas. Prof. David Wallace helped with the DOME software and provided crucial constructive criticism. Gaurav Shukla, without whose help my computer models would not have worked. Finally I would like to thank the SDM staff that has been a constant source of support during these two years of the Masters program.

TABLE OF CONTENTS

List of Figures

1 INTRODUCTION

1.1 Problem Statement

Product development, especially the design of large complex systems, is a special type of business process. What makes it special is the highly coupled nature of design decisions and the large size of product development teams. It is not unusual to have several hundred multi-disciplinary members taking millions of design decisions over the life of the project. From the Information Technology (IT) perspective, the special nature of design presents a challenge of coordinating the interdependent tasks of the team and integrating them with the efforts of other engineers and designers. In addition to the task of coordination there is the creative aspect of design which needs IT support as well [kappel]. These twin challenges make the development of appropriate solutions extremely difficult.

In general, IT solutions for product development have had far less success in creating comprehensive models of the design process than of products. There have been tremendous improvements in product modeling and analysis technology such as solid modeling, finite element analysis etc. Process modeling on the other hand has been slower to evolve. One of the main reasons for this failure is that fully capturing the knowledge that goes into a complex system design is hard. It is doubly difficult because a lot of this knowledge is tacit and keeps evolving over time. Another issue is that the product and process knowledge is interconnected. The product knowledge relates to "what" while the process knowledge has to do with "how" and "who" in the product development process. Capturing all this in a single system or representation has proven to be difficult.

This thesis explores the role of IT, specifically software tools, in improving the product development process through a framework that better represents and captures design knowledge. In order to overcome the limitations of the current implementations the concept of "Use models" is developed. Use models map the design process to an IT solution. As an example, this develops use models for three complex automobile systems, which form a subset of the product development process at any automotive company. Developing an efficient use model for designing hoods reveals many of the broader issues that IT systems for product development must address. Also, since the overall vehicle development process has several such use models, developing one highlights a common set of IT capabilities required for developing others. The main goal of this thesis is to generalize from these use models and define a framework that can be utilized in other complex system design contexts.

1.2 Product development

In recent years the importance of rapid product development and faster time to market has increased significantly. Arguably, product innovation is far more important today than it ever was before because of increased competition and higher customer expectations. This has led to greater focus on the product development process and methodologies to improve it.

In their book on product development Eppinger and Ulrich describe it as the "set of activities beginning with the perception of market opportunity and ending in the product, sale, and delivery of a product". The process by which products are designed and developed can been seen through several different lenses though in most cases these views are not mutually exclusive –

1. **Product development as a search** – Development of complex products by a team is in many ways a search through multi dimensional product attribute space. Selecting the right attributes requires problem solving and complex interactions between designers, engineers and managers. The development process for products like Automobiles, aircraft, machine tools that do not depend on technology breakthroughs could be categorized as predominantly a constrained search which is complicated by the fact that it is performed by a heterogeneous group of people rather than an individual.

2. **Product development as a social process** – According to this view design is a social process that requires negotiations and collaboration between different participants. In a study [Henderson] observed an organization which introduced a new CAD system and inadvertently destroyed the social communication practices that normally served "to repair the problems and misunderstandings that are part of the work process". As in that study, new tools sometimes destroy the traditional power structure and boundaries between organizations. The view of design as a social process therefore emphasizes that "more useful technology will follow from a better understanding of the natural, visually oriented communication process of design"[Kappel].

3. **Product development as a creative process** – In many situations product design is an ill-structured problem and the role of a designer is to convert it into a set of structured sub problems. The conversion process from concept to design requires creativity and judgement. Both these cannot be easily incorporated in a general-purpose program [Kappel]. This view of design as an exclusively creative process is perhaps true only in a limited number of situations like the development of radically new products.

1.3 Information technology for product development

IT led Improvements in the product development process (PDP) can be classified as those arising from –

- Business process re-engineering

- Improved modeling and analysis tools.

Process re-engineering in the context of product development has had limited success. This is mainly because most companies do not have detailed maps of the PDP. In fact in many cases it is the re-engineering effort that generates for the first time a very high-level set of guidelines and framework for the PDP. Otherwise "deep" process knowledge usually resides with people. The Product Development System (PDS) at a large automobile company was studied as an example of an initiative that aims to systemize PD knowledge. Intended to close the design productivity gap between Japanese competitors and US, the PDS in its first four years succeeded in codifying a large number of best practice guidelines. However, as a study by [Whitney, Qi] reveals, the amount of knowledge, even in a relatively simple sub system, is much greater than what is currently available in the form of design handbooks and the intranet knowledge base. This formal top down view of product development draws inspiration from system engineering. The military and NASA originally developed this approach for better management of complex missile and space programs [Whitney]. However it may not have as much success in the commercial product development space where requirements are driven by hard to anticipate dynamic markets and performance has multiple, subjective criteria. Another weakness of the PDS is that in practice it tends to be component centric because such knowledge is more easily articulated. System level knowledge on the other hand is unevenly distributed throughout the organization and largely in the form of unarticulated experience of experts.

Improvements in product modeling and analysis tools are well documented. The ability to create three-dimensional models, stress analysis simulations, high speed visualization and walkthroughs, are just some of the well known advances. However, even as these techniques have vastly improved the time, quality and cost of individual engineering tasks, the evidence on global improvement in PDP is not so clear[1]. Amongst the large automakers Toyota is the leader in fast product development but it is not the most aggressive user of IT. Part of the reason for a low correlation between the adoption of IT tools and a system level improvement is the difficulty of product development process re-engineering as described previously. Many companies tried to overcome the limitations of the traditional CAD solutions by developing Knowledge Based Engineering (KBE) programs that attempt to capture product development knowledge. The KBE efforts too have had limited success [Whitney, Qi]. A big reason seems to be that though KBE is very effective in capturing well-structured localized knowledge it is not so good for non-local knowledge.

The combinations of business process re-engineering and better software tools clearly have been less successful in the PD context than in improving other business processes. This thesis identifies some of the reasons for this and develops the use model framework for creating better IT solutions for PD.

1.4 Information technology trends

Information technology is one of the key enablers of high quality, cost effective and rapid product development. The past few decades have seen the development and deployment of several key pieces of the IT infrastructure for product development – powerful graphics workstations, broadband networks, CAD/CAM/CAE software, enterprise wide information systems and a host of other related technologies. Figure 1-1 presents a framework that classifies this evolution on the basis of potential benefits. The focus of this thesis is on "level 3" where "the business process is redesigned to maximally exploit the IT capabilities" [Venkatraman][Wagle][Rolf][Bobby].

[1] Interview with Dr Whitney

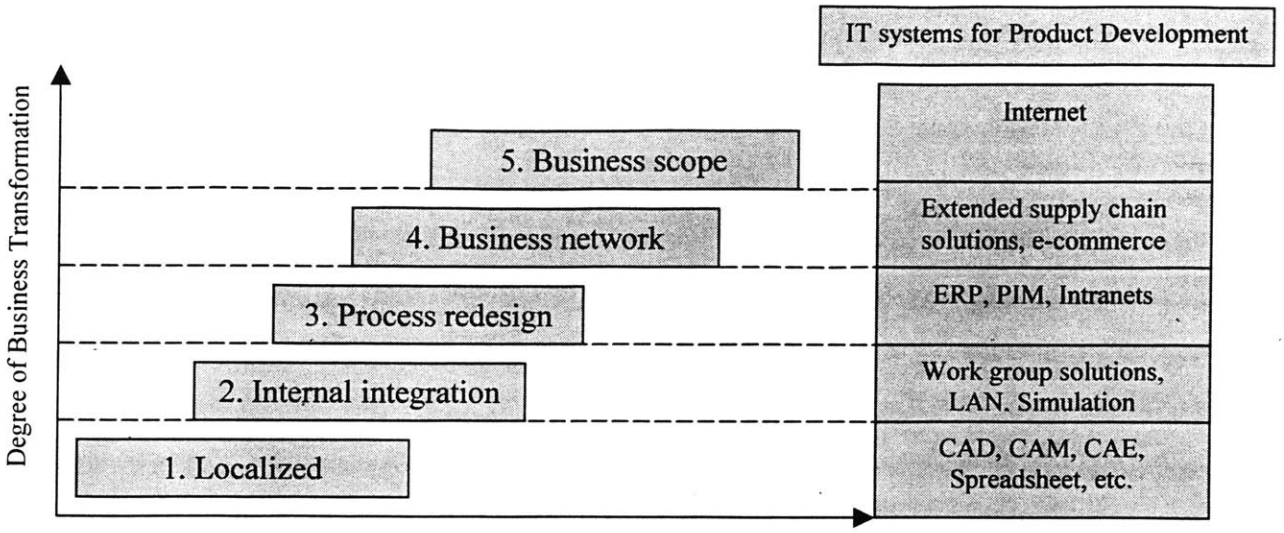


Figure 1-1 Five levels of IT-induced reconfigurations [Venkatraman]

Historically, IT support for product development has been synonymous with CAD/CAE tools. However, most large-scale product development efforts have always been supported by data management solutions. In many cases these were homegrown solutions. In recent times design software companies have started offering Product Information Management (PIM, or Product Data Management PDM) software to replace these proprietary IT solutions. These products are targeted as enterprise wide solutions for managing the product data as well as implementing a well defined design process. The commercial systems have eased many of the problems associated with large-scale configuration management, data security and access etc. They have not been as successful in design process management [Collier]. Apart from the product development process re-engineering difficulties identified earlier there are two other important technological reasons. One is the bottom up evolution of the product development IT systems in most companies, which has led to a collection of mutually incompatible applications. The second reason is that in the software vendor community the design process is not as well understood as are some of the processes for which enterprise wide information systems have been successfully developed. (Enterprise Resource Planning is one example [Wagle][2])

[2]The success of ERP systems is also not a matter on which there is universal agreement

According to [Arturo] the academic initiatives in this field have focused mainly on "developing software applications that support the implementation of specific process improvement techniques and in developing frameworks that allow the capture and sharing of cross-functional information". There are a number of such tools and they are usually classified as computer aided simultaneous engineering systems. These are essentially software tools that help in coordinating the work of teams and provide links with commercial geometric modeling software. Most of them have not been adopted for widespread commercial use, largely because of concerns about support and scalability.

1.5 Overview of the thesis

This thesis is divided into six chapters. The second chapter describes use models in product development. The product development process is a function of several factors and the Use model in each industry varies depending on these factors. Chapter 2 describes these factors in terms of the product and process modeling requirements. Chapter 3 further elaborates the concept of use models in the specific context of the automotive product development process. Chapter 4 and 5 present a methodology for implementing Use models through a software application. The final chapter presents a summary and a discussion of future work.

2 USE MODELS IN PRODUCT DEVELOPMENT

2.1 What is a Use model

A use model comprises the tools and processes by which teams of designers and engineers develop products[3]. Use models depend on the nature of the PDP. Since product design issues differ across industry segments clearly there are development processes that are more suitable for some situations than others. The fundamental cause of differences in PDP's could be summarized as those arising from –

1. **The nature of the market** – Some product markets are more competitive than others are. In some, a dominant design might already exist making incremental improvements the focus of the development efforts. Some markets may show strong network externalities forcing companies to think about standards and platforms rather than point solutions. Thus there are several dimensions along which markets could differ and they in turn affect the nature of the "ideal" product development process. In a competitive environment the product development process gets optimized (over time) to respond to the kind of market the company operates in. For example, a software company in a network driven market concentrates on fast time to market, perhaps even at the exclusion of everything else. On the other hand a space launch vehicle company might build its PDP around reliability and meeting schedules for a single customer.

2. **The type of the product** – Products that are assembled from discrete parts have a very different development process as compared to others like chemicals that are not assembled. Assembled products by themselves vary significantly in terms of the modularity or integrality of their architecture. Complex assembled products like cars have the paradoxical need for a great deal of functional specialization and at the same time sophisticated technological integration. This makes the PDP quite distinctive as compared to that for the development of new pharmaceutical drug.

   Products also differ in the degree of technological uncertainty. Some like satellite telephones depend critically on breakthroughs while others like cars are based almost entirely on proven technology.

[3] A formal definition is presented in the next chapter

3. **Company culture, organization and design philosophy** – One of the most important features of product design is creativity. Different cultures and organizations approach the design problem with their own unique blend of knowledge, aesthetic choice and philosophy. This deeply affects the nature of the development process. Cultural traits, depending on the context, can also make it more or less easy for companies to adopt radical changes in the PDP. For example, a history of centralized decision making might serve as a brake on effective adoption of IT tools that empower designers at lower levels.

All the above variables affecting the PDP also impact the type of use model. This relationship between the nature of the PDP and the use model however is complex and not a simple cause and effect phenomenon. For example, the concurrent engineering philosophy of product development has some obvious implications for how the use model for that process should be. However, in many cases an entrenched use model could also define the product development process. At least part of the resistance to concurrent engineering could be attributed to the old generation use models arising out of the "over the wall" design process. The legacy use models were resistant to change for several reasons. These had to do not just with the high switching cost in terms of human resource and process but also the difficulty in changing the tools of the previous generation that embodied the old process.

The use model concept is intended to analyze the tools and methods that enable knowledge exchange during the product development process. It is in that sense a map of the design process. It differs from other approaches in that it is –

1. Based on detailed and specific information maps for "vertical"[4] system design projects.
2. Designed to evolve as the development process changes

[4] The term vertical here does not imply that these systems are independent and do not have any horizontal linkages. It merely signifies that this system requires rich and distinct knowledge in order to complete the design.

3. Includes a networked IT framework that offers greater flexibility and ability to capture local and non-local knowledge.

2.2 A brief history of use models

Undocumented use models – collection of tasks, methods and tools by which teams and individuals develop products – have been around as long as humans have been conceiving and designing products. In its simplest form all the information and skills needed for a use model could be with a single individual. This was fairly typical of early product development where the innovator, designer and manufacturer of the product were almost always the same person. As the complexity of products increased there was a greater need for specialization and standardization. Specialization was a necessity given rapid technological development that created richly distinct bodies of knowledge. Mastering all was beyond the cognitive abilities of most practitioners of product design. Standardization allowed people from different disciplines the re-use of products and knowledge from other areas of expertise. A typical use model in this early age of simple specialization was based on the "over the wall" mass production design method. Each designer in the PDP would complete his or her part of the process and pass on the results to the next person. An obvious limitation of this approach was that the design process being inherently iterative and interdependent, a linear process led to lower quality and a great deal of rework. In a paper on the history of concurrent engineering [Smith] points out that product developers were aware of these problems as early as late 19th century. The persistence of "silo" based product development organizations was therefore not just a matter of lack of insight in the PDP but also a function of the limitations of the tools of those times. This is consistent with the view that the use model is both a creation of a particular product development process and at the same time the limitation of tools might create a use model that eventually drives the PDP.

The availability of cheap computing power in the early 80's and development of inexpensive CAD packages at around the same time changed the product design practice at many engineering companies. They engendered new use models as well. The new hardware and software that replaced the drafting boards of corporate design offices led to an exponential increase in their ability to effectively visualize and analyze parts. Further advances like solid and surface modeling made it possible to accurately create mathematical models of products that were so close to the real things that sometimes they eliminated the need for physical prototypes. The virtual prototyping and modeling helped firms simultaneously analyze a product from a variety of functional perspectives [Funk]. This made possible better tradeoffs between manufacturing cost, product performance, and quality.

In the late 80's and early 90's in addition to the rapid improvements in computing and design software another fundamental change took place in the form of the explosion in computer networks. To begin with these were small clusters or Local Area Networks (LAN) that enabled cooperation between small groups. These early clusters then grew into networks connecting all of the internal corporate departments ranging from accounts to design. The Internet took this a step further by creating a standards based network of networks. The new worldwide web of companies and consumers will almost certainly have a big impact on future product design processes. In terms of use models the beginnings of LAN greatly increased the ability of designers to exchange information and simultaneously work off the same basic data. A concurrent engineering "use model" was further enabled by the emergence of corporate networks whereby not just the designers but the marketing executives, cost accountants, almost everyone could participate in the PDP. Software to support such collaboration too was developed. During this time Product data management (PDM) and Enterprise resource planning (ERP) emerged as two of the largest software markets underlining the need and desire for integrated product development.

The improvements in communication technology and the ability to share "rich" information at very low cost is a fundamental new factor that will create entirely new use models. Until recently computer technology was a bottleneck in the adoption of at least some of the theoretical "best practices" in product development. For example, in the past true concurrent engineering was difficult because of scalability and reliability issues. These limitations have disappeared fast in the last decade. In the absence of these roadblocks the new challenge is the development of business processes and tools that fully utilize the capabilities of modern computing environments. The evolution of such a use model has been slower than the relatively rapid adoption of CAD software. Largely, because it involves far deeper changes in the organization – in the way product development teams are organized, in their incentives, and in the software tools. The slow speed of change does have a historical precedent – the move from steam to electric power in factories. The development of the electrical motor altered the basis of production facilities design. Historically factories were designed with the overriding requirement of power transfer from a large centralized power plant. This constraint had a greater bearing on the layout even than the needs of the production process. The electric motor eliminated this constraint and afforded far greater flexibility on how the factor floor could be arranged. However it took more than forty years before new factories truly began to exploit this flexibility [Erik]. The emergence of computer networks affects the nature of information flow and product design use models in a similarly fundamental way and organizations have been just as slow in adopting the new technologies that might completely transform the nature of design work.

2.3 New use models enabled by IT

The use of information technology in product development is ubiquitous. However, the benefits in terms of improved system design are not so clear yet. One contrary indicator, described earlier, is that some of the most successful complex system design companies like Toyota are not the most aggressive users of IT. At the same time it is clear that IT enables new ways of creating products – new use models, that were previously unimaginable. For example for simple products it might become common that customers themselves "design" products, especially the cosmetic aspects. This is already true in the computer industry. Companies like Dell allow customization of their product to a very high degree. In more complex situations customers might help in product development by clicking through preferences on a company's web site. Toyota recently announced that for a particular car model the company would be able to deliver made to order cars. Though this does not directly affect the product development process it gives reliable and current data to the product development organization about customer preferences.

The software products industry presents some interesting examples of IT enabled use models. Nowadays commercial software products are complex enough to justify comparisons with traditional engineering systems design. To take one example, Windows 2000 has 40 million lines of code and reportedly 3000 engineers working on it. The Microsoft product development process described by [Cusumano] as the "synch and stabilize" has been extraordinarily successful. In terms of the use model it is characterized by daily builds which makes it possible to evaluate the product performance even as it is being developed. This procedure makes it easy to integrate the code of several different software engineers working independently. On the downside, the relative ease of integration leads to low attention towards developing elegant software

architecture. Microsoft and many other companies have found it difficult to maintain the integrity of their software as new features are added. Another distinctive feature of this use model is that the inherent modularity of software products[5] allows for easy testing and evaluation of product before and after integration. This is in sharp contrast to the situation in automotive development where system level testing and performance evaluation has to wait till the very end of the process. Unlike software product development, the IT support for such testing is limited because the performance of a car cannot be completely quantified and tested by a software model. This restricts computer-based evaluation to only a limited set of features of the product. Even the types of features that can be evaluated, such as the packaging in a car, are problematic because of lack of suitable tools. For example, at a major automotive company engineers are forced to wait, sometimes for days, to retrieve designs because the use model of "checking in parts" to the master car model periodically is not well supported by their IT architecture.

The comparison between the software product development process and vehicle design reveals that these have radically different use models. Some of the differences arise out of the limitations of the tools and others have to do with the nature of the product. The set of variables that make them different can be classified as –

1. **Frequency of synchronization** – The ability to synchronize or integrate the product frequently can decrease the time required to complete a design. This is because more information is available to everyone early enough so that incompatible changes are not made and the product does not vary too much from the agreed upon design. As pointed out earlier in case of products like software it is common to integrate everyday.

[5] Modularity of software products can be analyzed at two levels. One is the level of product features and the other at the implementation level of software code. Truly modular and independent software code is rare. Despite advances in object oriented programming, code reuse in most commercial development environments is low. However, product features modularity achieved through separate libraries and executables is commonplace. The one being compared here is feature level modularity.

2. **Number of participants and complex interfaces** – In case of products such as automobiles, aircraft's etc the PDP has thousands of participants. In addition the interfaces between the product components and sub-systems are complex and interact in ways that cannot be easily modeled[6]. In this case it is difficult to create effective use models because there is a lot of knowledge/information exchange during the PDP and a significant part of this is tacit knowledge.
3. **Product Architecture** – Use models for modular products are different than those for integral products (all other things being equal). Modularity of products can imply a more modular and decoupled process. This in turn would mean that the use model is fairly linear and non-iterative. For integral products the reverse would be true because each part/sub system of the product would affect several other systems.
4. **Tools** – The relationship between tools and use models is complex. In general, tools should support efficient use models for PDP. Many times they tend to create inefficient use models. Also, not all tools are purely software related. Some like rapid prototyping are dependent on the development of new hardware and materials technology. Figure 2-1 summarizes the different factors that affect use models.

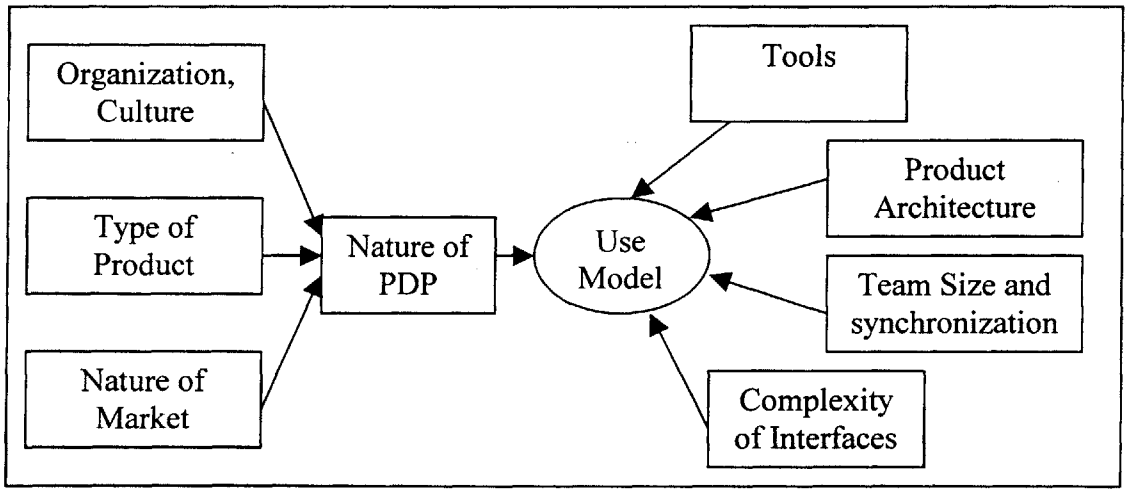


Figure 2-1 Factors affecting the Use Model

[6] This limits the incentive for frequent synchronization. Another limitation is that design tasks in traditional products tend to be longer than programming tasks. However, frequent integration would be attractive for early discovery of packaging issues.

Table 2-1. Comparison between two "use models".

PDP	Elements of Use Model		Factors affecting Use model
	Methods	Tools	
Commercial application Software Example Word	• Object-oriented architecture • Abstraction, encapsulation, operator overloading and Inheritance	• UML • Rational rose modeling tool • Java, C++ • Source code control software	• Modular product architecture • Availability of OO languages • Ability to synchronize everyday
Mechanical System design Example Machine tools	• Integrated product modeling • Axiomatic design	• CAD/CAE • Axiomatic design software • Intranets, email	• De-coupled product architecture • Geometry modeling tools • Inability to create comprehensive models • Number of multi-disciplinary participants • Complex interfaces between sub systems

The comparison in Table 2-1 can be repeated for several types of use models. For example use models for mechanical systems might differ depending on whether the system is integral or modular. The machine tool example is predominantly modular and performance driven. However a car is integral. The performance of a car is not as quantifiable as that of a machine tool. Consumers make car purchase decisions on a variety of dimensions, and pure engineering performance is a subset of that process. In creating the use models for either case these factors affect how the designers interact, the extent and need for collaboration, the uncertainty in the process etc.

2.3.1 Requirements for a new architecture for product development systems

The special nature of design creates the need for special use models and IT tools. There are five key dimensions along which software tools and the PDP differs fundamentally from other business processes such as Bill of materials creation, accounting, or financial transactions.

- **Concurrency** – For any complex product development project usually more than one designer is involved and often a team of designers, engineers and analysts require access to the data. In addition several other users might want to access the data for non-engineering reasons such as cost analysis, documentation, etc. Many times this access is required during the work in progress stage much before the final design is complete. In terms of software requirements this implies that it is essential for several people to have read/write access to the same data simultaneously.
- **Collaboration** – Not only do designers need to have concurrent access to data, they need to exchange "rich" information to carry out their tasks. The product development system therefore needs to support a "distributed, multidisciplinary, team oriented" effort. [Robert].
- **Iteration** – A design process is inherently iterative. According to [Whitney] "Dramatic improvements in development time require a sophisticated understanding of interrelationships and iteration drivers". Studies show that a significant part of designer's time is spent on negotiations – the process of establishing requirements through iterative consultations with others in the team [Crabtree]. A product development system needs to be able to model the iterations in a design process.
- **Complex data models** – The data that designers exchange is extremely complex. [Cleetus]. Typically these are solid and surface models of components, simulation results, and other files containing hundreds of data elements and their inter relationships. To make matters worse most of the data is in proprietary formats and there is low inter-operability between various systems that generate the data.
- **Complex interaction** – A typical designer working with a CAD system has a very open ended and complex interaction with the software. If the CAD system is seen as a component of the larger product development system then it is clear that as compared to other enterprise wide solutions the nature of the interaction of users with the system is highly unstructured.

2.4 Product and process modeling

The methods and tools that teams use depend on the product and process modeling requirements of the PDP. Thus use models for simple products and processes also tend to be simple. Figure 2-2 shows a matrix that classifies products and their design processes according to their complexity. As pointed out earlier, there are several elements that contribute to the complexity. Product complexity is a function of the number of attributes, the nature of interactions between them and the architecture. The number and diversity of participants, and the project time line could be some of the drivers of process complexity. The focus of this research is the type of use model in the upper right hand quadrant.

Another factor that affects use models is the stage in the life cycle of the product and market. Figure 2-3 presents this idea in terms of the rate of major innovation in products and processes. The key notion is that as a product matures process innovation gains in importance. This research will develop use models for automotive design, which clearly falls in the category of mature products.

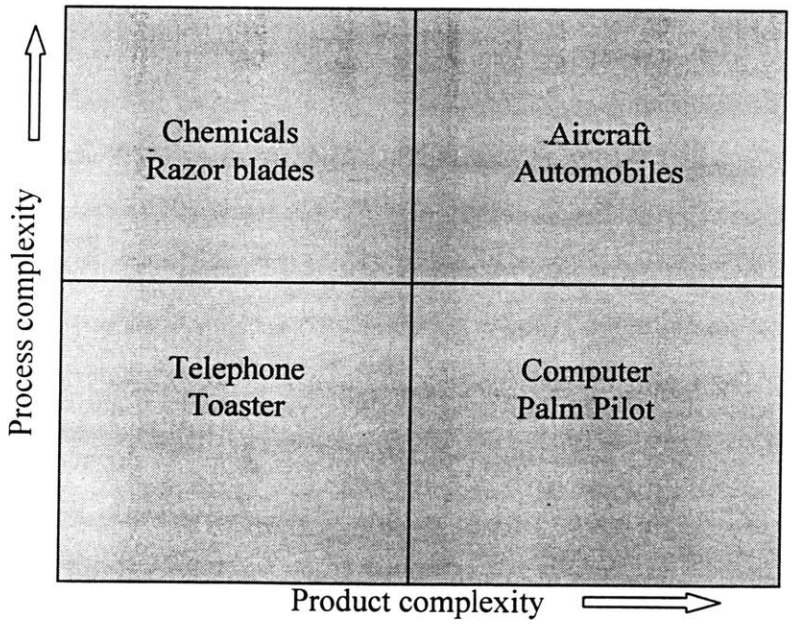


Figure 2-2 Complexity of Use Models

For mature as well as new product lines there are differences in the complexity of architecture and in the rate of knowledge change. In some segments the integral nature of the product might imply that a highly inter-dependent design process is required. In some cases the rate of knowledge change in the field could be very high. Figure 2-4 adapted from [Allen] captures this idea. It shows the rate of change of knowledge, task duration

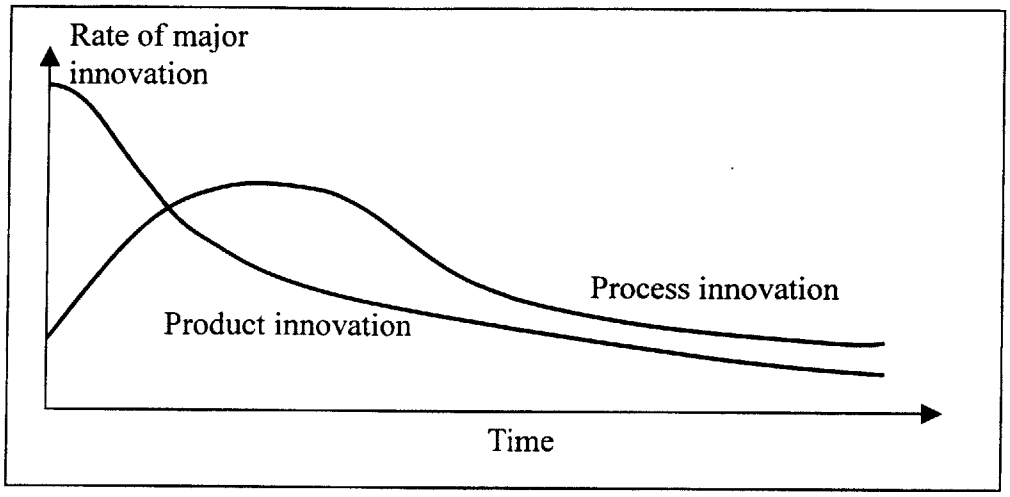


Figure 2-3 Rate of product and process innovation as a function of time [Abbenathy]

and interdependence. The implication is that if there is a lot of task dependence and knowledge changes then a project structure is suitable for the product development team. On the other hand if the process involves relatively modular tasks with low rate of knowledge change then a functional organization is more suitable. In terms of the use model the automotive process can be described as long tasks, very high interdependence and moderate knowledge change. Thus in this case the PDP is comprised of a search through a fairly well defined design space. What makes it complex is that a large, distributed, multi disciplinary team performs the search.

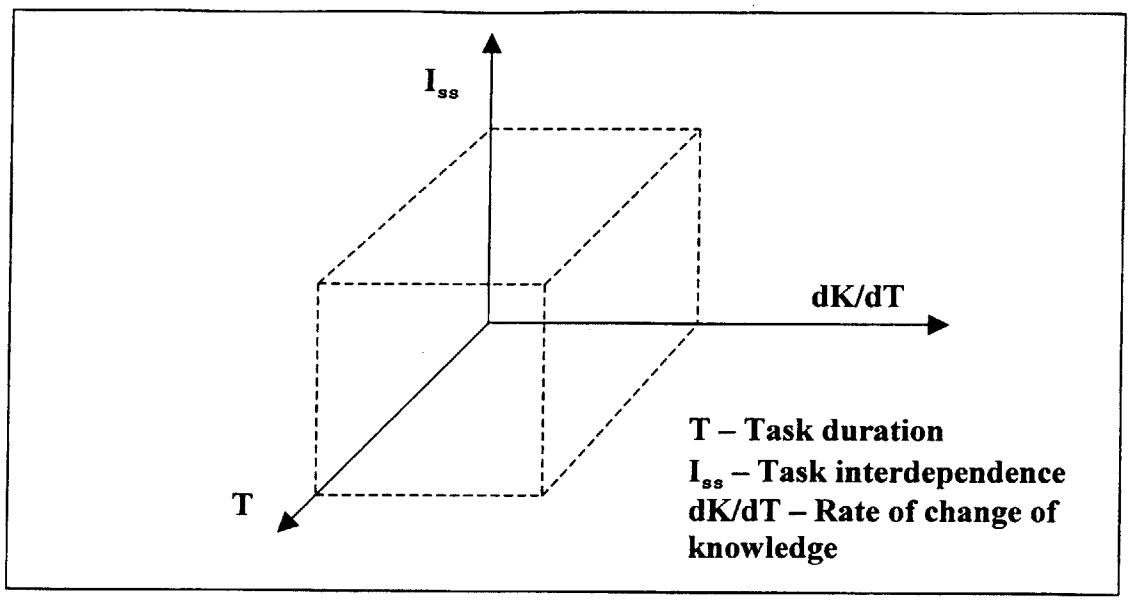


Figure 2-4 Rate of change of knowledge and Use models (reproduced from [Allen])

2.5 Design Structure Matrix

Alexander [Alexander] proposed a method for creating diagrams representing the interaction of different elements of a design. Following on Alexander's work, Steward introduced the Design Structure Matrix (DSM) for information flow analysis. In its simplest form, the matrix can be compared to a directed graph, which represents a traditional flow.

This research uses the DSM for modeling the product development process. This approach involves "mapping an existing or proposed design procedure into a simple array representing the complex inter-relationships among the many design tasks which must be accomplished." [Whitney]. The key benefit over other types of process maps is that the DSM can model iterations and couplings that are peculiar to design projects. For example the DSM in figure 2-5 shows six tasks and their dependencies. The row with task C has three marks corresponding to tasks A, B and D. This implies that in order to complete C data is needed from the other three. Since A and B are before C they do not present a problem. However D is after C and that means C cannot be completed before D. Such feedbacks are common in design and the DSM is an effective tool for capturing these.

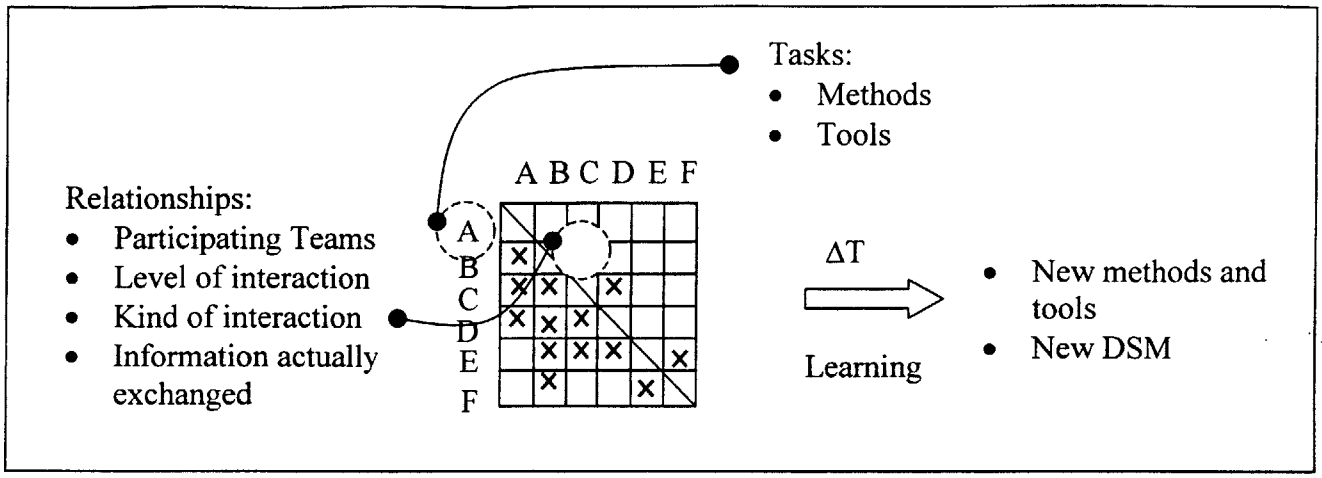


Figure 2-5 DSM and use models

2.5.1 Use models and DSM

The DSM by itself is a process map while use models are meant to be more than that. Figure 2-5 shows the critical parts of a use model. It includes the methods and tools that appear as rows and columns in the DSM. The crosses that signify relationship between tasks are also part of the use model. It is important to know not only that there are dependencies between tasks but also the level or magnitude of these, the way they are handled, and who is responsible for them. The use model changes over time, as better modeling techniques and tools evolve. This change could lead to improvements in isolated methods and tools or it could even make the whole DSM redundant. An example of a disruptive technology that had a similar effect was the CAD workstation. This virtually eliminated the old paper-pencil use model of creating two-dimensional drawings.

2.6 Limitations of current tools

Product development for complex systems is special for several reasons. According to [Whitney] – "The design of automobile, aircraft or computers can involve thousands of engineers making millions of decisions over several years. None of these many tasks is performed in isolation. Each design choice may be a tradeoff affecting many other design parameters". Several studies have confirmed the special nature of the system design process [Banares][Huang][Cleetus].

In developing software solutions for complex product design it is important to map the desired design process to the tools that can help a team implement that process. The mapping is difficult as compared to other IT solutions because of the special nature of the development process. This thesis develops a mapping scheme that treats PD tasks as objects and creates a network of these based on the DSM. The main insight that the DSM gives us in developing the software solution is that the design process is complex and to develop effective solutions require information models at a greater level of detail than currently available.

Flexibility in process Modeling – The IT solution needs to be flexible enough to allow re-configuration[7] as the design process evolves. Figure 2-6 shows this point graphically. The nature of Product development in general is such that both information exchange and network topology (defined as the network view of the DSM) is partially known. For example, at the beginning of the design process it is not completely known who will participate in the design and what information will they need to exchange. This is in contrast to most other systems. For example, consider an application that allows the creation of a new checking account. In this case the network topology (the tasks that need to be performed and the order in which they need to be performed) is completely determined. (In fact "hard coded" in the software). The information that needs to be exchanged is also completely structured and well defined. Such business processes have had some success in implementing Enterprise Resource Planning (ERP) solutions. However, for product development the sequential, transaction oriented approach is not suitable.

[7] Re-configuration is used here in the sense that the software should be able to be customized for the new process without programming.

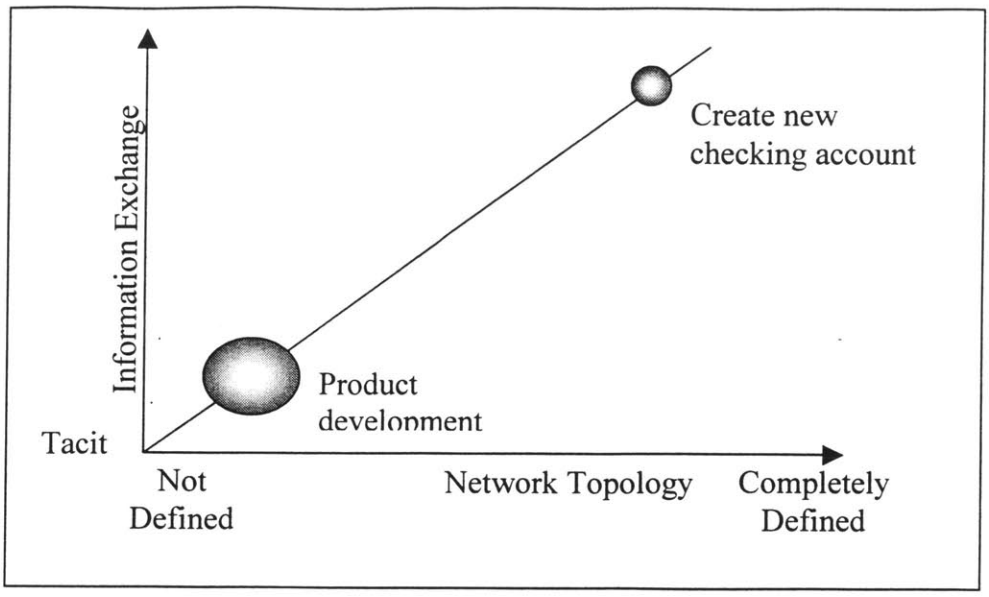


Figure 2-6. Comparison between design process and other business process.

The learning loop – As the Use Model evolves with better methods and tools the software implementation needs to change as well. Current solutions are too rigid and require programmatic changes to implement the new developments. This results in the separation of those who use the system and those who create it. The disconnect leads to inefficient solutions which do not reflect the actual design process and furthermore do not change at the rate at which designers improve their methods. A flexible software architecture that allows visual changes to the process model can significantly improve the "learning loop". There are two kinds of knowledge that a design process requires – local and non-local. The learning loop for non-local knowledge, which is essentially the "relationships" information, can be "captured" by a suitable process modeling tool based on the frequency of use. The local knowledge is too diverse and detailed for a single method to incorporate the learning loop in the software architecture.

Tractability and Visualization – For a typical system design problem the use models are likely to be large and the tractability of these models is essentially dependent on how effective the visualization tools are. One possible solution, that is widely used, is to use "visual indexing" to create a product view. A single product structure is created for the entire product and this corresponds to a complete three-dimensional model of the product[8]. The advantage is that most people find it easier to remember the visual mock-up of products than any other abstract representation. Visualizing processes, especially the iterative product design type, is harder. The DSM is a good candidate. It is capable of capturing and succinctly displaying the process. Current IT tools for PD use workflow software to visualize the process. These rely too heavily on the linear view of the project.

Associativity – A key feature of the software architecture has to be associativity at various levels. There is a trade-off between associativity and user flexibility. For example seat belt designers at the automotive company under study do not associate the cad data to the assembly configuration because they want the flexibility to change their files. The fact that seat suppliers are outside vendors and the belt designers depend on their data further complicates the situation. An object oriented architecture for encapsulating design tasks can help solve this problem. Such a mechanism would allow for synchronous and asynchronous execution of multiple tasks. Another issue with associativity is visualization. Complex models with hidden associativity are almost unusable by everyone except those who created them. Current KBE solutions have this limitation. As the relationships get complex the KBE software becomes harder to use and understand without extensive training in its internal working.

[8] Most commercial CAD systems also support different views of the product structure. The as designed, as manufactured, as assembled to name a few.

2.7 Summary

This chapter developed a general definition of a Use Model and also described the key factors that affect Use models. Since there are many different kinds of product development processes based on the industry context, there are also many different kinds of Use models. The Use models discussed in the next few chapters concern complex products, which have complex development processes. In terms of rate of change of knowledge these Use models are in industries which have moderate changes and the bulk of the innovation is the creating new and more efficient processes. As discussed here such use models have special needs in terms of IT solutions. A framework is presented that addresses these requirements.

3 AUTOMOTIVE PRODUCT DEVELOPMENT: USE MODELS

3.1 Introduction

The automotive development process is a collection of many use models (Figure 3-1). This development process is by nature very complex and integrative. However, the technology uncertainty is relatively low and most vehicle projects do not require any new breakthroughs in order to succeed. In terms of the framework described in earlier chapters it can be characterized as a problem solving and search process. The social interactions are extremely important as well because of the large size of teams and the cross-functional and cross cultural nature of large auto firms.

Some of the issues in developing suitable IT solutions for automotive projects are the same ones describe earlier. In addition because of legacy software and process complexity there are some other factors as well –

- The use model and software needs to support several different levels of decision making. The hierarchy could be in terms of component, sub-system and system level.
- Automotive companies have a huge portfolio of disparate software applications. Many of these applications need to "talk" to each other in order to speed up development. One approach for achieving integration across application has been to replace legacy applications by software from a single vendor. Another is to introduce data exchange standards like STEP and IGES. Despite these measures it is likely that a lot of different applications will survive. The approach proposed here is to enable process integration through event propagation across applications. This is complementary to the data exchange effort and it has several advantages over a pure data exchange strategy. The primary benefit is that not all vendors need to agree and deliberate over a common standard. That's why in practice integration via published

interfaces (called API's) has proven to be very successful. In addition, by allowing event or services based integration between applications a greater level of concurrency can be achieved. Also, services cane be "exposed" to other application through standardized web interfaces.

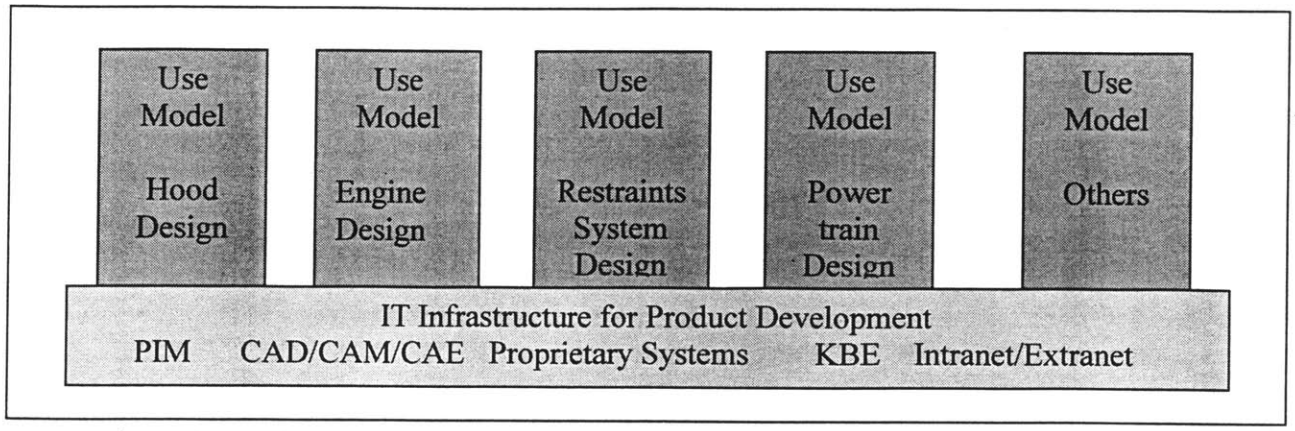


Figure 3-1 IT infrastructure for Product Development and "vertical" solutions

3.2 Definition of use model

A use model is a set of tasks, methods, software tools and a map of the process by which a team in a product development project uses these

Tasks – Actions that need to be performed for completing the project

Methods – How the actions are/should be performed. These could be software implementations of heuristics, analytical solutions, mathematical models, and/or best practices that designers are required to follow.

Software tools – The applications that allow the engineers to carry out the methods.

Map of the process – The product development process is a complex web of inter-linked tasks, the map is a snapshot of these relationships. As the design process evolves the map changes as well.

A use model as it is defined here is intended to be different from the conventional user models that are associated with individual application software. For example the Win95 look and feel is a user model associated with many windows applications. It is also meant to be different from use cases associated with customized software. These user scenarios are usually static, hard coded process models, which are not suitable for most product development situations.

Table 3-1. An abstract view of a Use Model [Whitney]

Use Model		
Type of Task	Method	Tools
Know **what** facts	Database Search, Query, Match	PIM, CAD, Intranet, etc.
Know **how** procedures	Search, Match, Query, Create a new one	Intranet, Online Design Guides, Text, Video, Audio, People
Know **who** relationships	Remember relations, decide which are important, Search process models, Search project plans, ask experts	Process modeling software, project plans, People
Know **why** "Deep knowledge"	None available	People

Use models for "vertical" design processes collectively make up the product development use model. Each of the vertical system design projects such as hood design or engine design requires very specialized product and process knowledge. There is however a common thread among all of them. Table 3-1 shows it terms of the nature of task and knowledge. A typical design process has some or all of these characteristics. The first item in the table is well supported by current IT tools. However, relationship knowledge for completing a design task is something that is mostly with people and which they learn over time. For in-experienced engineers the lack of this information also limits their ability to use the procedures since they need to know the relationships in order to know what procedures to use. In some ways formal product development processes that define clear milestones, procedures, and requirements are supposed to mitigate this difficulty.

However, most formal design processes are too high level and do not adequately reflect the rich information exchange that typifies complex system design. A study of throttle body design at a large automotive company by [Whitney] is a good example. The paper describes a two-stage process of preparing a DSM. In the first step they used the design manuals and the documented process to create a DSM. This was refined after interviewing experts and the resulting "real" DSM had five times as many marks as the first one. Quite clearly very little IT support exists for the rich information exchange that occurs during even a relatively simple system design such as throttle body.

3.3 Developing a Use Model

It is clear that for complex product development a complete DSM breakdown may not be available, either because it has not been formally documented and/or because it has changed radically since last documented. Therefore the first step towards creating a use model is to document a DSM view of the design process. This reveals not only the rich knowledge exchange and uniqueness of the process but is also a good empirical measure of the complexity.

Figure 3-2 shows a methodology for developing a Use model. Once the DSM is documented it can be "optimized" by reducing coupled tasks and eliminating non-value added ones. This can serve as the baseline for mapping to an IT solution. The resulting Use Model and DSM have to change with time as new methods and tools become available or if new business processes are adopted. In addition to the DSM each tasks can be associated with a method and tools that automate or assist in carrying out that method.

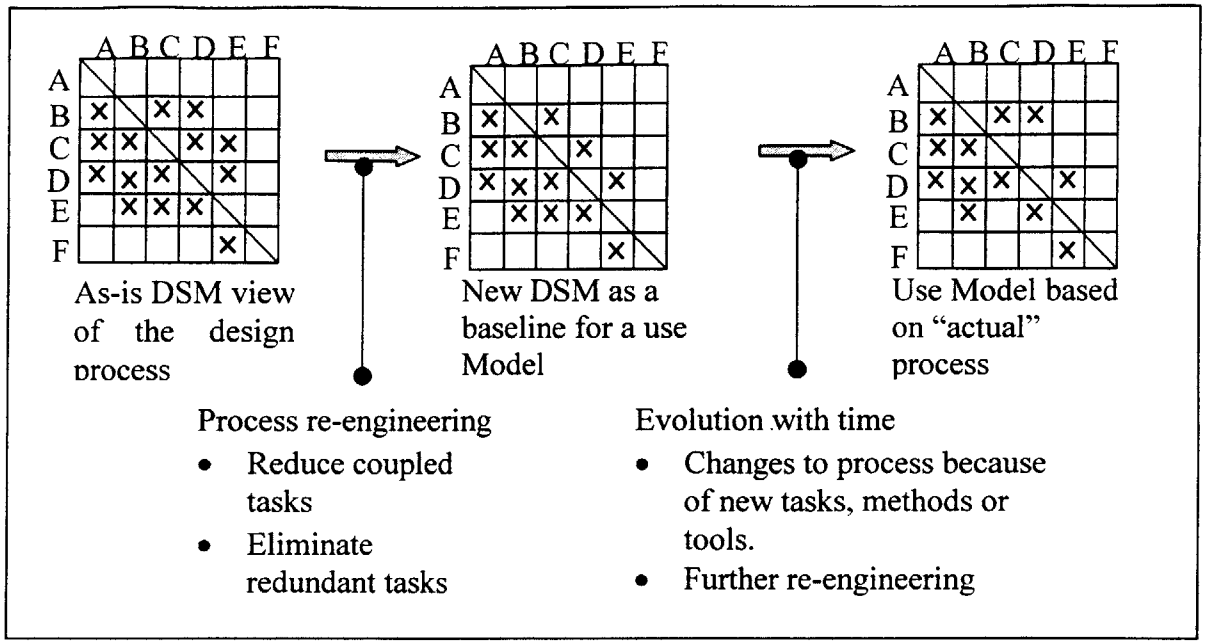


Figure 3-2 Methodology for developing a Use model

3.4 Automotive development process

The automotive product development process is a large and complex enterprise involving hundreds of engineers, designers and managers. IT systems that support the vehicle development process have traditionally evolved bottom up with very little support for the highly coupled decision making process involved in the product development effort. Most companies have their own internal design rules and processes that they use in addition to the general product and process modeling capabilities provided by the CAD/CAM/CAE/PIM tools. These implicit "use models" tend to be inefficient since they are not created to fully realize the potential of IT for large-scale product development. Also, being company specific and proprietary, they have not impacted the evolution of the CAD/PIM tools to the extent that they should have.

Figure 3-3 shows one view of the vehicle development process. According to this view the vehicle design begins with a strategic intent. At this early stage of design, vehicle level parameters such as over all size (compact, sub-compact), price point etc. are fixed. As the design is further detailed more system level parameters are frozen and more people get involved in the design process. In general, at the vehicle level the decisions are made by a small number of people who are, in most situations, in close touch. As the design moves "down" to the level of subsystems several hundred functionally diverse (and sometime geographically dispersed) design engineers participate in the process. Table 3-2 shows the differences between these stages of the vehicle development process from the viewpoint of developing a Use model.

At each of these levels the decisions taken in the previous level impose constraints and new design decision cause iterative loops that might imply changes at higher levels. The sequential view of the vehicle development process depicted in Figure 3-3 does not reflect these complex sets of inter dependencies. It also assumes that at each stage one "final" design is passed on to the next stage. If the development process follows the so-called "set based approach" [Sobek] then the number of iterations and coupled tasks increase greatly.

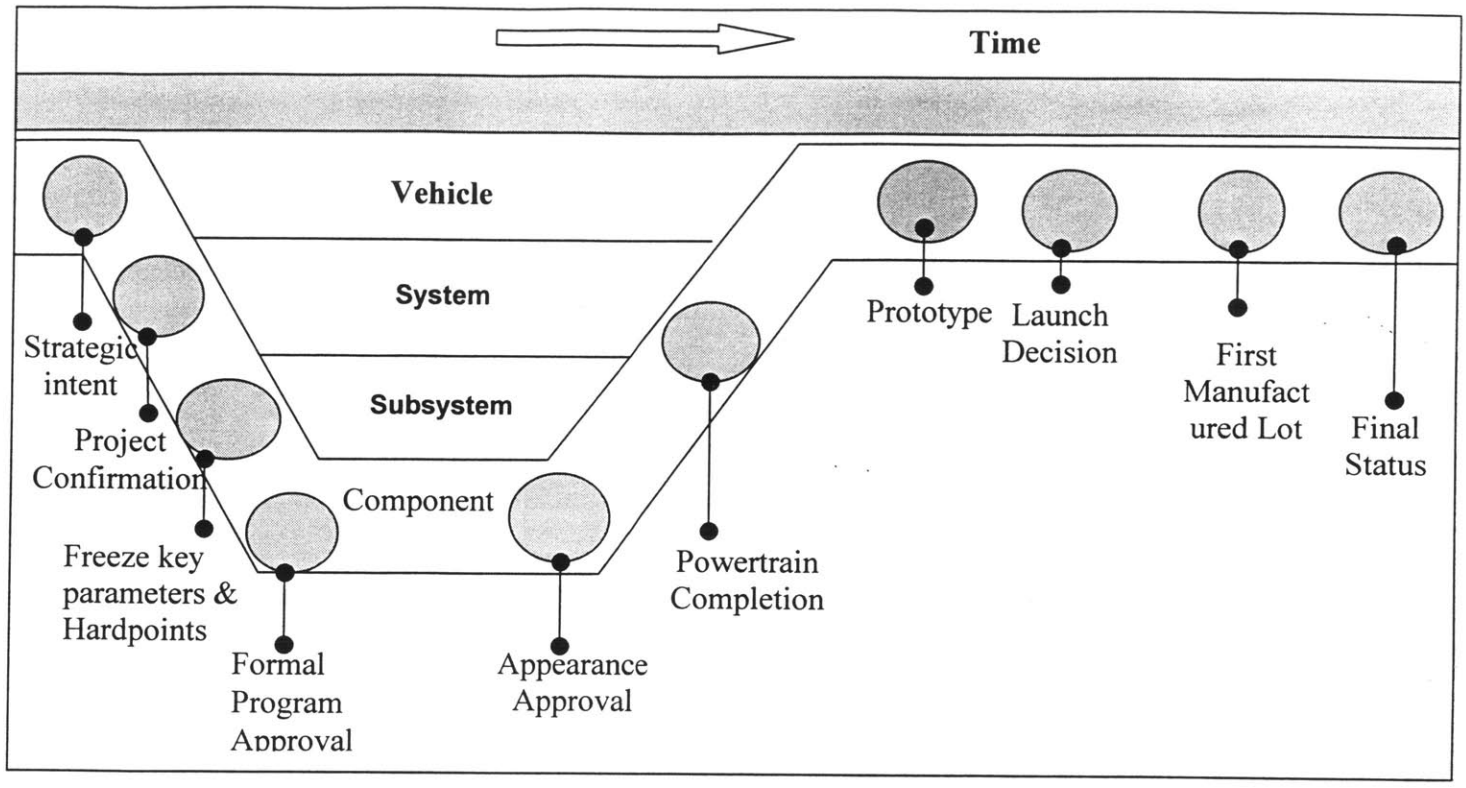


Figure 3-3 "V" shaped automotive development process [9]

It is clear from Table 3-2 that different process and product-modeling requirements distinguish use models, at each stage of the vehicle development process. In the early definition stages the requirements of collaboration and concurrency of decision making dominate other aspects. While at the component design level the product modeling (geometry definition, analysis etc.) is more important. For sub-systems such as a hood, or a door, all the factors that make product development a special business process are discernible. The number of designers and the number of decisions to be made are too large, complex, and inter-related to be addressed by typical transaction-oriented IT systems.

[9] Based on systems view presented by two large auto companies at MIT

Table 3.2 Determining the complexity of a Use Model

Level	Number of Design/Technical Decision makers	Number of decision variables / actions	Use Model driven by following factors
Vehicle	Low	Low, but with high impact on subsequent decisions	• Concurrency • Collaboration
System Level (E.g. Body, Power train)	Medium	Medium	• Concurrency • Collaboration • Iteration
Sub-System (Hood, Engine, etc.)	High	High	• Concurrency • Collaboration • Iteration • Complex Data and performance Models • Complex interaction
Component	Low	Low/Medium[10]	• Complex Data and performance Models • Complex interaction

3.5 Examples of Use models in restraint systems, throttle body and hood design

The "meta" use model for automotive product development is made up of several overlapping use models. Chapter five presents DSM studies of three sub-systems – restraint design, throttle body and hood. These were developed through interviews carried out a large automobile company [Qi][Lavine][Zambito]. A comparison of these DSMs reveal a lot of commonalties from the IT architecture point of view (these are discussed in the final chapter). It also shows that a huge amount of PD knowledge is undocumented. At the company where these studies were carried out there was little appreciation of their complexity until the DSM was made.

The main objectives of the restraint system design project is to develop a seat belt system that meets the safety and regulatory requirements while satisfying the packaging and styling needs of the vehicle. In this case, design interactions often crosses organizational boundaries. The process of selection of the appropriate seat travel requires negotiations between the Full Service Supplier (FSS), and the designers and engineers at the auto company. One of the problems in the current process is that each organization has a unique and sometimes conflicting perspective on the necessary tasks and who should be doing them. [Lavine]. At least part of the reason for the conflict is that before the DSM was made no one had a clear idea about the process. The creation of the DSM produced a useful map and also made it possible to analyze and re engineer the process.

The next logical step in the improvement process is the creation of an appropriate use model. A big opportunity in this situation is to improve the co-ordination of tasks between the four participants. This can be done within the framework of the existing CAD/PDM applications. However, such a solution is unlikely to be effective for the reasons identified in earlier chapters. Another approach is to create a use model that maps the DSM to an IT solution. In this approach each task in the DSM is treated as an object that has certain data associated with it and can provide some services. The services are essentially the methods that can accomplish that task. Not all services are computer programs and some (perhaps many) are performed by human engineers and designers.

To take one example from the safety belt DSM consider the task "determine seat travel". The object representation of this task is shown in Figure 3-4. The design service of finding out the seat travel is dependent on several tasks such as defining allowable anchorage locations, determining anchorage points, CAE performance evaluation etc. A general-purpose algorithm for this task is therefore unlikely. However, given the DSM view and linkages with other objects from the DSM it is possible for a designer to figure out –

- Which tasks are dependent on this one (for input or output)

[10] This might not hold for certain components, which are parts of several sub systems.

- Who is responsible for those tasks
- Trade-offs between different options and constraints on the design.

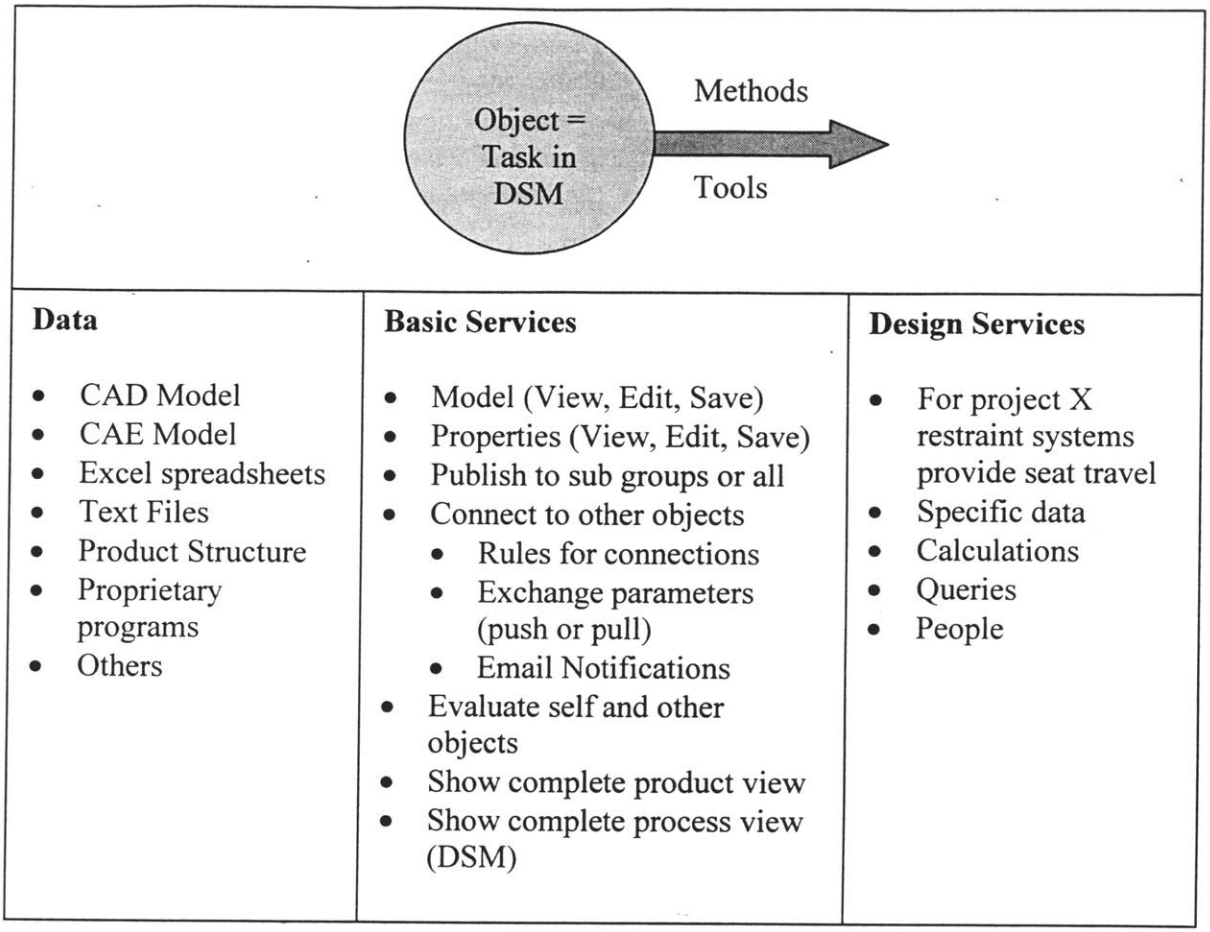


Figure 3-4 Tasks mapped to object with data and services

The are three other examples of complex systems in chapter five. Appendix A is a DSM view of the design of throttle bodies [Qi Dong]. In this example it is clear that this relatively simple system being part of a larger system makes its design dependent on several other sub-systems. Modeling these interactions is difficult because complete math models for them do not exist. In order to reduce the time of development the automotive company in this study developed a generative knowledge based engineering (KBE) solution. This software presents the user with a starting point and guides through design steps as the user defines the attributes and constraints [Whitney]. The underlying engine is ICAD, a KBE tool-set that provides engineers with libraries for programmatically creating

geometric models. These general-purpose libraries can be used in conjunction with design rules to create the type of KBE solution that the Auto Company has deployed. This approach has several limitations. The ones observed by [Whitney,Qi] can be listed as –

- Only a fraction of the design knowledge is actually captured by the KBE
- Local knowledge is easy to program but non-local knowledge is much more difficult.
- The KBE systems are difficult to use by new members of the team. And only a few "experts" are currently able to use it. This seems to defeat the very purpose of 'knowledge capture".

A use model for the throttle body can be constructed in the same way described earlier. All the tasks can be represented as objects, which provide data and services. Their inter relation ships form the DSM. The use model concept does not replace the KBE system. It fills the gaps by providing services that the KBE systems are weak at representing. Such a use model will have several advantages over the KBE approach –

- The use model will provide the designer with a view of the process. This is important to provide a context to a new user. Current tools are extremely good at providing "product context" in the form of three-dimensional geometric models. Designers can navigate these models and get a clear picture of where their part of the product fits in space given the overall product context. However, a similar process context is missing. The use model provides this missing piece in form of the DSM.
- System level knowledge and relation to other sub system DSM's can also be modeled
- Designers can visually change the inter relationships between objects, add objects or change their properties during the PDP. This will enable the capturing of as practiced design process and provide rich data for further re-engineering.
- If new methods are developed then can be included in three forms – as new relationships, as a set of instructions that need to be followed, and as independent computer programs and models that are added to the Use model. A complex new method might be a combination of all these mechanisms.

The goal of the use model in this context is therefore to provide the building blocks for creating an effective IT representation of the product development process. These building blocks could be seen as objects that encapsulate product development knowledge.

3.6 The relationship between the top down approach and Use models

The over all PDP at an automotive company can be analyzed from two extreme points of view. One view holds that all requirements and processes can be defined top down. Once that is done, the designers, engineers, and managers would follow these guidelines and that would be lead to a successful PDP. The other view is that the PDP is a market place for services where designers, engineers, manufacturing engineers and managers can come together and look for what they need to get their job done. A computer network can enable these interactions and this would create an efficient market model of the PDP within the company and its suppliers. In practice of course the PDP lies somewhere between these two extreme views. Relatively lower level design details do have a flavor of the market in the sense that designers, engineers negotiate options and trade-offs. At the same time each vehicle project does have an explicit goal and stages that are defined top down.

The use model approach can accommodate any combination of the above philosophies of PD. In case of automotive development the top down approach cannot capture all that is needed to successfully complete the project simply because of the magnitude of knowledge required. However, in the interest of managing the project within cost by measuring progress it is important to have stages at different states of completion. Also, some top-down rules are required for managing the data[11].

[11] There are more than 4,000 CAD software users and more than 3,300 PDM users at the company under study. Additionally, more than 3,000 participating supplier have similar systems. Currently there are nearly 30 vehicle programs that have implemented the new IT system that includes CAD/PDM and other tools.

3.7 Summary

This chapter presented the concept of use models in the context of the automotive development process. In view of the scope and complexity of automotive PDP, this is a rich area for exploring the IT architecture requirements and the way Use models can be applied in practice. An important aspect of any use model for this type of a PDP is the several different levels of knowledge it encompasses. There were four types identified here. The first one is "what" – the knowledge about the tasks and product data. This is strongly supported by current IT solutions. The other types – "who" and "how", are not as clearly a part of the current IT system for PD. In developing the Use model a central idea is to incorporate all these types of knowledge in a single framework. Such a framework would support a much richer representation of the PDP and capture the true complexity of system design. It would also serve as a compromise between a completely the top down system engineering view of PDP and a bottom up details driven approach.

4 DEVELOPING AND IMPLEMENTING USE MODELS

This chapter describes the process of developing use models and suitable software architecture for implementation. The first section is a discussion of some differences between the current "data modeling" approach in product development with the proposed use model method. It is followed by a description of the old and new software tools required for implementing a use model. This framework is essential for the following chapter regarding the development of use models for restraint system, throttle body, and hood.

4.1 Use model Vs Data models

A unified data model for an automobile that completely defines all product data has for long been a goal of many designers and design tool developers (especially CAD companies). The advantages of such a comprehensive model are many – it would make data sharing across various engineering disciplines easier, a single standard file would simplify integration between computer applications, etc. In practice this goal has been difficult to realize and might be unattainable even in the long run given the speed with which new applications are developed. The other primary difficulty is adopting a single representation that satisfies all. Efforts such as IGES and STEP have had some success by limiting the focus towards CAD data exchange and product modeling. The larger question of a comprehensive data model for all product development activities – marketing analysis, financial trade-offs, besides the traditional engineering modeling requirements, defies a simple solution. The Use model methodology is proposed to overcome some of these issues. The two approaches can be compared and contrasted along the following dimensions –

1. **Product Vs Process focus** – The data modeling approach is focussed on product modeling. It does not directly address the issue of product development process modeling. However, adopted data models do affect the process in ways that are not originally foreseen. For example, at a large automotive company, the data model for assemblies does not sufficiently separate the geometric information from the part positioning information. This data model results in a process whereby engineers are forced to download entire assemblies when all they needed is the positioning matrix for one part.

   A use model approach on the other hand is focussed on representing the process. An efficient data model in this case is just one aspect of the modeling effort and not the central theme. If IT tools are developed with a use model in mind, the solutions are likely to lead to a more efficient product development process because instead of the tools driving the process the process would dictate the type of tools that are needed. It would also ensure that the underlying data structures are best suited for the intended use model.

2. **Development process** – The data model approach depends on a single team or committee to create a unified model. This approach is essentially top down and is driven primarily by the IT requirements and capabilities rather than the needs of the designers and engineers. The IT limitations that have primarily influenced the current use models are the need for scalability, speed of access, lack of standards for data exchange, and network bandwidth limitations. A use model approach is a better guide towards eliminating these IT limitations. The proposed development processes for a use model is a combination of the top down and bottom up approach. Product requirements are defined top down while the detailed use model is actually developed by those who use the tools. The other difference is that a use model is developed for specific system design tasks. For example, a use model for seat belt design requires knowledge that is unique to each company and its design philosophy and methods. All firms have their own methodology developed over the years. A data model based

approach, which attempts to boil all this down to a single representation, faces three big problems –

a. The single representation needs to be generalized to include every possible requirement making it less efficient.
b. Most companies outsource their IT solutions yet it is not in their interest to reveal all the proprietary knowledge required to make customized tools and this limits the utility of the resultant "standard" data model.
c. The data model implementation effort depends on the CAD vendors. However, they do not have enough knowledge about complex system design processes. This is essential for developing process sensitive software through customization of general IT packages.

The use model solution is to build process models from basic building blocks in the same way that CAD tools are used to build product models. These basic object frameworks could be developed by the IT vendors and enhanced as the technology improves. One example of such a framework is the Distributed Object-based Modeling Environment (DOME) developed at MIT. This software, and use models implemented in it are discussed in the later chapters.

3. **Changes with time** – The rate at which new techniques and methods develop is much faster than the rate at which these are translated into commercial tools. This is because those who develop the tool and those who use it are different groups and many times different organizations. A data model approach overseen by a single committee that drives the standard is therefore unlikely to evolve as fast as the business practices might want them to. The use model is not maintained by a central organization and so it can mitigate this problem to some extent by allowing end users to make changes to the model. In order to maintain a minimum standard of process quality, some checks and balances are required. Monitoring product performance as well as post project analysis of the process could enforce some checks.

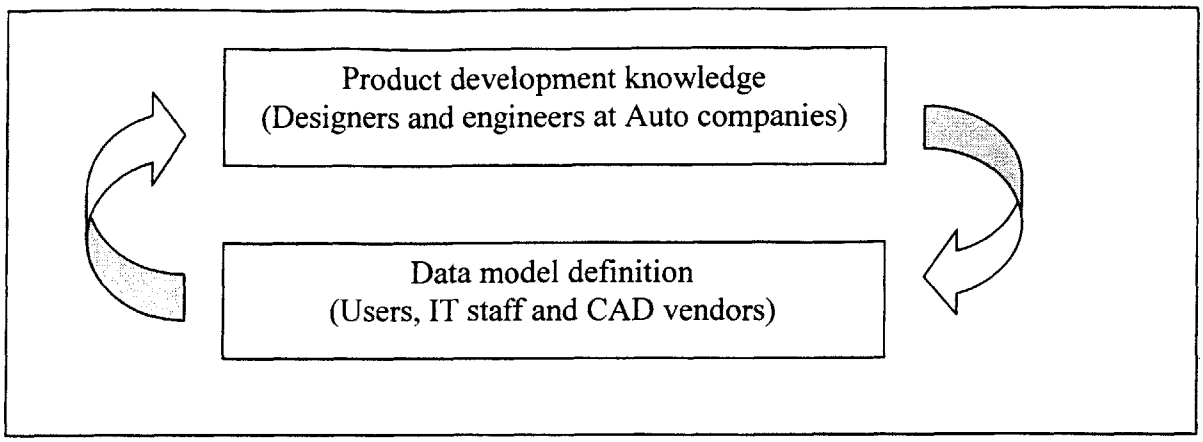


Figure 4-1 Time lag between new PD knowledge and changes in data model

4.2 The role of existing IT systems in implementing Use models

Manufacturing companies have made substantial investments in IT for product development. As described in Chapter 2 the IT tools for product development have evolved from stand alone design workstations to large networks that manage design data and processes. The use model approach relies on these systems for many services such as geometric modeling, walkthrough simulation etc. The services used from the legacy IT tools can be classified as –

- **Computer Aided Design (CAD)** – In the Use model context CAD tools provide the product modeling services. Companies commonly use solid and surface modeling for creating realistic representations of actual parts and assemblies. These are used extensively by designers and engineering to resolve issues such as packaging, stress analysis, etc. In addition to part creation many CAD packages support associative modeling. Such parametric modeling services can be used to automate simple design tasks and this service too can be linked into the Use model for complex system design. In addition to geometry lots of other data can also be made available from the CAD system. This could be materials and their properties, topological connectivity for assemblies, interfaces with other sub systems.

- **Product Data Management (PDM)** – The most important service provided by PDM systems is a safe centralized database for all CAD and other product related data. In many cases this repository or vault is presented to the user as a hierarchical tree structure. This product structure view is the main metaphor used by PDM systems to arrange product data. Companies use the product structure to create a product hierarchy with assemblies, sub assemblies and parts. Each node in the product structure can contain CAD files, bill of materials information and other data. The big PDM vendors also offer integrated software for configuration management and project management. In the use model framework PDM software is an important piece because all product related data has to be accessed from the company's PDM database. In addition the product structure service is also utilized as the primary product view. This tree data model is however is not sufficient to present all the inter relationships of the product. For example the as assembled view of the product might have different hierarchy. These different product views can be derived as long as the use model captures all the relationships.

- **Workflow** – The commercial workflow software that is currently available is not widely used in automotive product development. It is likely that the lack of scalability and a linear process model unsuitable for PD are some of the main reasons for the low deployment. An even more important reason perhaps is that most companies simply do not have detailed maps of their design processes that are needed for implementing effective workflow systems. In the use model framework the DSM view of the process serves as the workflow view.

- **Analysis** – CAD and PDM software tools provide the services required for modeling the product. A variety of Computer Aided Engineering (CAE) tools are used to analyze these models. Finite element analysis, computational fluid dynamics, kinematics analysis etc are some of the common methods. Companies have developed a rich portfolio of proprietary applications built on the basis of these methods. They perform a variety of simulations, analysis and other internally developed procedures. In the use model context these tools and methods correspond to individual tasks in the DSM. In many cases these tools and the way they are used would distinguish one company's use model from another.

Table 4.1 shows the gaps in services for the different features of the PD such as concurrency, iteration etc. The combined effect of the above gaps is that the as modeled PDP is much poorer in knowledge than the as practiced process. This low quality representation has been confirmed by several DSM studies as well as by the lack of progress on system level knowledge based engineering tools [Whitney]. A software implementation of the use model framework that supports concurrency, collaboration, iterations, complex data models, and complex interactions would overcome many of these limitations.

In addition to the software tools a formal method for knowledge capture is needed. At the component level data structures and simple math models can sometimes be acceptable. At the system level complete analytical models are rare and no single data model is sufficient to express all the complexity. The use model approach is therefore to gather system level relationship data through the DSM. Once the DSM is identified it serves as a meta-map of the knowledge that is missing from typical point solutions. Combining this process view with the existing methods and tools therefore presents a powerful system level model. Users can navigate the model through the DSM map and develop a much better understanding of the process context.

Table 4-1 Gaps between services provided by existing software and those required for implementing use models.

Tools	Service	Gaps
PDM and CAD	**Concurrency** • File based • Simultaneous changes by Workgroups • Configuration management through product structure views • Secure database (Vault) • Multiple views of product structure for different users	• Parameters and feature level concurrency are missing • Changes by distributed workgroups are supported for very small groups. Large-scale development needs are not met. • Constraints and relationships that cross the nominal product structure are not supported. • Inability to re-chunk the data dynamically according to different criteria.
PDM, CAD Workflow	**Collaboration** • Uniform product structure view • Linear process model • Pull model for design change notification. Limited push capabilities	• Process view reflecting the true complexity and knowledge in PD is missing • Information push and pull features based on the project context are missing. • Sophisticated design change notification is not fully integrated with process and product models
PDM, CAD Workflow	**Iteration** • Not provided for, considered and anomaly	• Can't model iterations in the process model • Can't display iterations and dependencies between products and processes, and between participants.
CAD and PDM	**Complex data model** • Standards for data exchange (STEP, IGES)	• Not sufficient support for Integration based on event propagation independent of a single data model. This type of integration should be visual rather but most vendors

		require programming. • Data exchange standards are focussed on products not on the process.
CAD	**Complex interaction** • Simplified user interface • Design guidelines • Knowledge based engineering at component level • Product structure view	• Process context in the form of DSM is missing. This hides he complexity of interaction and the as modeled process is much poorer than as executed. • Knowledge based engineering at system level is not supported.

4.2.1 The industry structure of IT tools companies for product development

The previous section focussed exclusively on the product and process modeling limitations of the existing IT solutions for PD. However, another issue is the way in which IT solutions for PD are deployed in companies. Good analogs for PDM/CAD systems are the Enterprise resource planning (ERP) solutions for operations/Manufacturing. ERP systems have been widely deployed and the ERP software industry grew prodigiously in the mid-1990's slowing down only in 1999. The ERP market is therefore a relatively well-developed and mature business with a tried and tested implementation model. Figure 4-2 shows the comparison between ERP and PDM industry structure. A typical ERP implementation cycle includes a phase of business process re-engineering followed by a long IT system analysis and deployment phase. The industry structure reflects this phenomenon. There is a very large base of trained consultants with deep knowledge of business processes and the ERP application suite. The "implementers" usually partner with the target company for rolling out the system. A similar deployment of PDM system usually follows a somewhat different path. To begin with there are no companies with the size and scope of big ERP implementing firms. Nor are the tool developers themselves as large as ERP companies like SAP. The total size of the market is smaller as well but the complexity of implementation is comparable to ERP. In addition there is the issue of the distinctive nature of product development discussed earlier. In a typical ERP implementation scenario the system integrators "hard code" a single way of performing a particular process. These process models are based on scripts

developed during the BPR phase. A similar approach would not work in product development because it would overly constrict designers and engineers and make the process less efficient.

Even as the PDM/CAD software industry evolves, it is likely that deep product development knowledge will primarily remain with companies that actually develop products such Ford, Boeing, or Sony. IT system for PD therefore would need to be more flexible than their ERP counter parts have been[12]. This flexibility is essential so that users of the system can evolve the process and product models on their own without programming. It is also important for being able to effectively model the product development process in the first place because the knowledge resides overwhelmingly with the users of the system and only they can create an accurate model. The flexibility is also important for continuos process improvements.

[12] ERP systems have faced a lot of flak for lack of flexibility and many ERP implementations have struggled with the difficulty of changing "hard coded" process models [Collier][Wagle][Bobby].

IT tools for Manufacturing/Operations **Enterprise Resource Planning (ERP)** *SAP, Baan, PeopleSoft etc.* Competency • Developing software • Business Process Re-engineering (BPR)	**Implementers** **Big 5 Consulting/Accounting firms** *Anderson, KPMG, E&Y etc.* Competency • BPR • IT implementation	**Users** **Fortune 500 Companies** Investments – $10 Billion/Year Competency • BPR
IT tools for Product Development **Product Data Management (PDM), CAD** *PTC, SDRC, Dassault, etc.* Competency • Developing software	**Implementers** ? (The users are the implementers in most cases) Small compared to users and ERP market	**Users** **Fortune 500 Companies** (Ford, Boeing, etc.) Investments – $2 Billion/Year Competency • Implementation • Developing software • IT implementation

Figure 4-2 Comparison of CAD/PDM and ERP industry Structure

4.3 Software architecture for implementing use models

The previous sections and Chapter 2 described many of the limitations of current software tools for product development. In light of the object representation of the DSM developed in the Chapter 3 it is clear that a similar approach can be followed to identify the gaps between existing software and that required to create efficient use models. Table 4.1 presents such an analysis. The majority of the tools for performing individual design tasks that require local knowledge in product development are already available. The challenge is to create a network of these tools and methods that not only effectively

captures the PDP but also is flexible enough to incorporate new learning. The next sections in this chapter describe the Distributed Object Modeling Environment (DOME) that can serve as glue between the heterogeneous applications used in product development. In addition it provides functionality that can link local knowledge and tools into a system level network.

4.3.1 Distributed Object-based Modeling Environment (DOME)

The DOME software is an integrated modeling tool developed at the CADLAB in MIT [Francis]. In this thesis the DOME software was used to create use models, and demonstrate the utility of such models. The choice of DOME was driven by the fact that it fulfills several of the gaps identified in Table 4.1.

The DOME software allows creation of complex system models made up of inter-related objects. The objects can encapsulate CAD Data, Excel spreadsheets, or any other applications for which plug-ins are available. A plug-in is a software module that is developed based on the application programming interfaces (API's) provided with most commercial software packages. For example, a CAD software application typically has API's that allow programmatically querying the part for geometric data, for material properties etc. Thus an object representation of a CAD model in DOME has access to most of the functionality of the CAD package. DOME users – designers, engineers, and managers – can visually connect such models together and create large system models. DOME employs three basic mechanisms to facilitate the building of such models –

- **Publishing** – Designers, engineers, and others can publish their models to the rest of the group. A published model exposes the services offered by that model. Figure 4-3 shows a model with two objects – an Excel costing model and a few design parameters part. The published models can be connected to each other or other objects.

- **Subscribing** – The DOME software provides a facility to subscribe to services from other objects. Thus the excel cost model can subscribe to a "size of the part" service from the CAD model. If the size of the part changes the Excel object would automatically recalculate the cost.
- **Creating relationships** – The objects can also be explicitly linked to each other through relationship objects. These relationships are of different types – numeric, logical, textual etc.

A big part of the DOME architecture is the ability to distribute objects over the network. So, a vendor could create an object with certain services published to the rest of the world. An Original Equipment Manufacturer (OEM) could subscribe to these services over the network. Thus a distributed design supply chain can also be modeled using the same architecture. Table 4-2 shows the design goals of the DOME software.

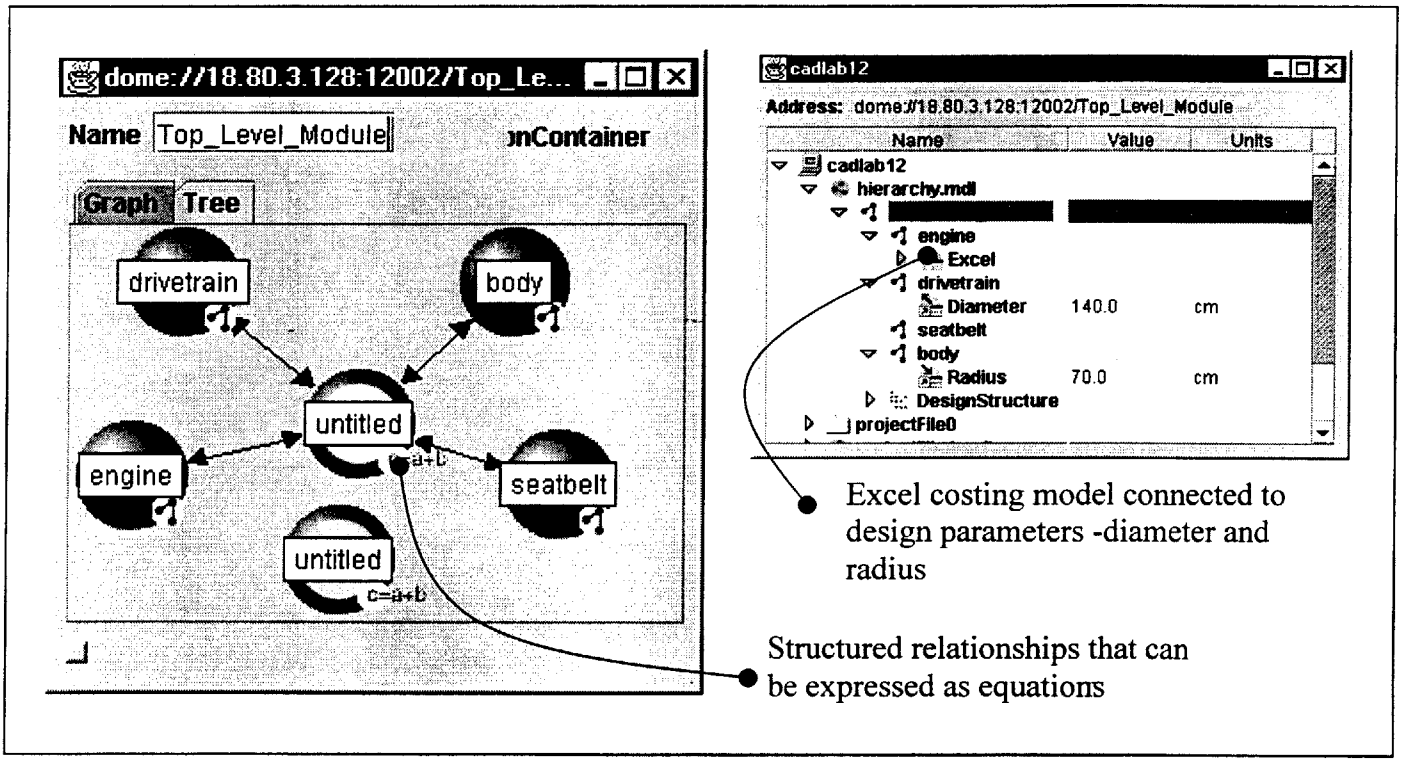


Figure 4-3 Object representation of the DSM in DOME for developing Use Models

Table 4.2 DOME functionality (Reproduced with minor changes from [Abrahmson]'s analysis)

Modeling	Evaluation
1. *Lowering the barriers to participation:* Creating an information system architecture that allows distributed users in different environments to participate in a transparent manner without investing significant time in learning the systems, tools, or a modeling language outside of their own expertise. 2. *Domain specific choice of tools:* Allowing each design participant/expert to use the tools, representations, simulations, heuristics or models which are most suitable within their domain. 3. *Flexibility at the user level:* flexibility to allow for the spontaneous and robust growth, extension, change, revision and reuse of integrated models, tools, or resources to solve evolving or new problems. 4. *Mixing of level of detail and quality of models:* Allows incorporating a mix of detailed models and incomplete or approximate models to support both top down and bottom up design. 5. *Support different collaboration models:* Accommodates both tight and loose collaboration ranging from co-located colleagues to outside consultants or suppliers while respecting a diverse set of intellectual property and work-synchronization needs.	1. *Real-time simulation:* Provides the ability to explore the design solution space, concurrently elicit trade-offs or causality between disciplines and goals, and monitor design evolution—in both a manual and automated fashion. 2. *Decision support:* support decision making through visualization and evaluation.

4.3.2 DOME and Use models

DOME is a general-purpose modeling tool that can be used in a variety of ways to model processes and products. It can be used to create an unstructured marketplace for design services or, at the other extreme, for implementing a rigid top down design methodology. This thesis advocates a combination somewhere between these two extremes. Figure 4-3 shows the method followed in subsequent chapters for creating use models in DOME. According to this approach, for each system level design task, such as hood design, a DSM view of the process is developed. The DSM is re-engineered and purged of redundant tasks through clustering, re-sequencing of task etc. The new DSM is mapped to an object representation. Each object represents a task in the DSM and embodies methods and tools needed to complete the task. Not all objects are based on software packages. Some objects might simply send an email to the relevant human expert.

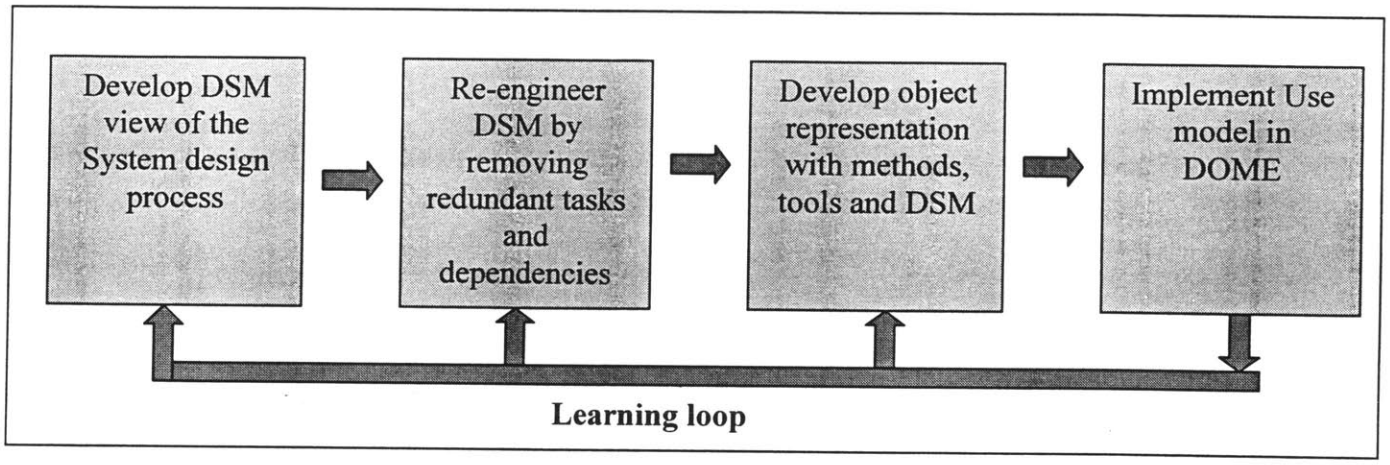


Figure 4-4 Using DOME for developing Use Models

A fully implemented use model in DOME has the following features –

- **Process view** – A DSM view of the process is associated with each use model. This view shows the designers and engineers the context of their tasks. It also helps them in visualizing dependencies and future tasks. Figure 4-4 shows the process view in DOME. It's a design structure matrix where the rows and columns represent tasks in the system design project. The marks in the matrix indicate dependencies between tasks. The marks off the diagonal represent information exchange between two tasks. As the use model is executed over several similar projects, the process view becomes richer and more accurate. The software facilitates this by allowing users to change the process and capture the interactions between the participants. This flexibility creates the possibility of non-value added and sometime harmful changes. Some steps that can mitigate this problem are –
  - Permissions based on roles, responsibilities, and fixing the scope of the system that the participant is responsible for.
  - Post project analysis and "cleaning" of the as executed use model.
  - Automatic tracking of interactions and data collection on infrequently used methods, tools and relations.

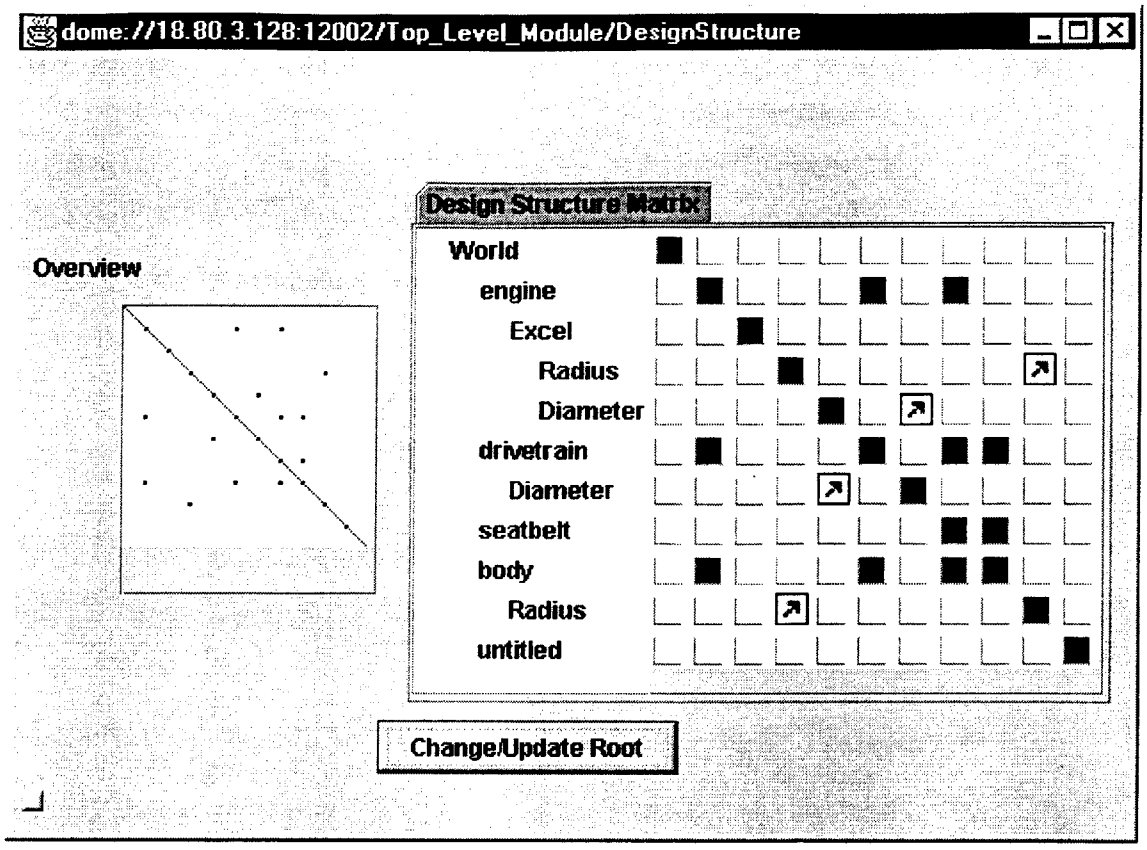


Figure 4-5 The Design Structure Matrix (DSM) as the Process view

- **Product view** – The product view is based on the services provided by CAD and PDM tools. A 3D view of assemblies and parts is commonly provided by CAD software. A product structure view is available from the PDM application. Figure 4-4 shows a VRML[13] viewer as the product view.

Figure 4-6 Product View in Netscape (can also be shown in a CAD/PDM package)

- **Resource View** – A resource view provides the list of people who are responsible for the tasks in the DSM [Appendix C shows an example view]. In a corporate setting the resource view could be a service provided by Directory services software[14]. Each task in the DSM and every interaction would have a group assigned to it. The directory services software will keep all this information current. Thus if roles and responsibilities change they would get reflected in the new resource/process view. Also tools that balance resource capacity and requirement balancing can be used.
- **Schedule view** – The schedule view is a top-level linear view of the process. It summarizes the milestones and completion gates in the PDP along with the dates.

[13] VRML - Virtual reality markup language. Most CAD packages provide translator from proprietary formats to the standard VRML. It is useful for visualization on the web when geometric accuracy is not vital.

[14] Large firms consolidate all employee roles and responsibilities information by using the directory services application. The main vendors of such systems are Novell, Oblix, and Microsoft.

- **Object representation** - This is a view of all objects in the Use model. All other views are derived from this one. Users can interact with this view and add/remove objects or change relationship between the objects.

4.4 System Engineering versus design marketplace

The Use Model framework described above is flexible in terms of the type of design philosophy that a company wishes to follow. A top down system engineering use model as well as an unstructured market place of design services can serve as a Use model. The examples developed in the next chapter, however, make use of a combination of these two extremes. The top down system engineering process is primarily responsible for driving the requirements and establishing key milestones. The Design structure matrix provides the baseline process context as opposed to being completely decided by the participants. As the project is executed the designers and engineers can alter the baseline as needed. This has the flavor of the marketplace idea. The implemented use model does not restrict the users. It does need to have a protocol for making the changes in a way that do not deteriorate the model.

4.5 The limitations of integrated modeling

The Use model described above is in many ways an integrated model of the product and process. However, it is not a complete analytical model. Thus it does not necessarily evaluate to any single "solution". It's an integrated map of the process and a model of the product. In general, complete integrated modeling of the product development process is considered infeasible because [Whitney] [Cutkosky] –

a. Methods for navigating, modeling and displaying complex tradeoffs are missing
b. Analytical models of several basic phenomena like dynamics of physical systems, complex system behavior etc. are missing. Even when models are available their validity is difficult to establish.
c. The different product development domains use different tools, data models, and often different data management systems.

The Use model approach is based on the assumption that in the absence of comprehensive integrated analytical models, the next best option is to create models that can complement the human participants in the design process. The proposed solution is the creation of large distributed system models with a focus on the user and the development process. Several different technology strategies have been tried for creating such solutions. The different schemes can be seen to polarize around two basic philosophies. One requires participants to represent their views in a common way that can be understood by other people or machines (data model approach). The other is integration by facilitating communication between different domain specific elements, which need to interact. In practice integration can be viewed as a continuum, bounded by these two approaches [Abrahmson]. Use models implemented in DOME fall in the category that tries solving the problem of communication between different multi-disciplinary "objects". While most commercial systems have adopted a file-based transaction oriented system, this approach is not suitable for representing a use model since it does not provide information at the task/design parameter level. In comparison, the object-oriented representation described here has several benefits– it maps on well to all the software requirements listed earlier, it is scalable and extensible and easy to integrate with existing software. In fact there has been an increasing trend to integrate through a commonly agreed upon suite of services instead of just data compatibility.

5 USE MODELS FOR THREE COMPLEX SYSTEMS

This chapter describes three examples of Use models within the context of the automotive design process – restraint systems (seat belts), hood design, and throttle body design. All the use models are for complex sub systems that are intricately related with other vehicle systems. In identifying the methods, tools and design maps for these system design projects the goal is to uncover the common conceptual and computational building blocks that can be re used in other situations as well. These examples are based on thesis research by [Qi][Zambito] and [Lavine].

The first case study is of throttle body design. This is the most detailed of the studies available and reveals how a relatively small and simple system can have complex relationship with other system models. The Use model is characterized by the need for coordination, performance modeling, and integration between different sub systems. The second use model discussed is for restrain system design problem. This project was dominated by the need for effective communication between suppliers and internal teams. It emphasizes the support for concurrency and iteration identified in earlier chapters. The last case is of the vehicle hood. The hood design problem requires not only collaboration, but also complex performance modeling. The use model is therefore defined by a need for coordination, integrating complex data models, and reconciling conflicting objectives.

5.1 Use Model for throttle body design

A throttle body is a part of the car's air intake system. It is a simple device consisting of only seven important parts that performs two basic functions [Whitney, Qi] –

- Metering air flow to the intake manifold
- Reporting the degree of throttle plate opening to the engine computer

5.1.1 The current method of designing throttle bodies

A detailed study of the product development process and supporting IT tools for this system by [Qi] shows that the key requirements for a throttle body can be classified as shown in Table 5.1. The external product requirements refer to the sub-systems that are

not part of the throttle body but affect its performance. Six such external systems were identified [Appendix A]. The key challenge that this system design project presents for the PDP is the issue of modeling the product and process at a level of detail that is accurate and easily understood by all participants. A Knowledge Based Engineering (KBE) system was implemented at the large automotive company where this case study was carried out. The KBE system solved some of the process challenges. For example, it provided a mechanism to exchange design parameters with the supplier. However, not all issues could be addressed with the KBE. Some of the limitations of the system are –

- Complex performance modeling such as that between the accelerator sub subsystem and the throttle body
- Lack of process view that can provide a context to a new participant. A novice user of the system typically had to speak to experienced designers and engineers before they could gain enough knowledge to use the KBE system.

Table 5.1 Requirements for throttle body design

Product requirements		**Process Challenges**
Internal	**External**	
• Maximum air flow • Leak-tightness • Resistance to back flow from accidental ignition • Safety requirement – Two self-acting methods of closing by itself	• Pedal feel or pedal response • Packaging and space limitation under the hood.	• Outsourcing of certain key components • System modeling required for pedal feel • Knowledge management

In addition to the above limitations from the perspective of Use models some other problems are evident –

- The KBE system under study takes a generative approach. It starts with a basic design, provides suggestions, and constrains the user to pre specified rules as he/she walks through the design steps. This approach works well at component level or for sub systems that are sufficiently decoupled. However, as the study by [Qi] showed the throttle body is a much more complex problem and the KBE system captures only a fraction of the knowledge that goes into system design.

- The KBE system is implemented as a rule base that can be changed only by programmers. This limits the ability of users to change the system for new process improvements and it also makes it difficult for a new user to understand the internal logic. This is in contrast to the use model approach where the system model is visually represented as a network of objects that exchange data and services. Design rules are represented as relationships between objects.

5.1.2 Developing a Use model for the throttle body

The throttle body use model was developed using the process described in Chapter 4. The Design structure matrix documented by [Qi] provided the baseline design process. Each task in the DSM was mapped to objects in DOME. The dependencies and relationships between tasks were modeled as relationship objects. Figure 5.1 shows the network of objects and the DOME view and Figure 5.2 shows the product and process view for the Use model. The product view is an IDEAS[15] parametric solid model. The key features of this implementation of the Use model are –

- **Product and Process interaction** – Design is both a product and a process. The complex interaction between the product model and the manner in which designers and engineers create it is therefore important. The fact that for large system several hundred simultaneous users work off the same model further complicates the dependency between the product model and the process. Two basic types of interactions can be listed

  1. Structured – Certain types of interactions are exchange of design parameters. In the throttle body case study a good example is the relationship between the bore diameter and the plate diameter [Appendix A shows the complete DSM]. The two need to be related because change in one necessitates change in the other. Traditional parametric CAD models are useful

[15] IDEAS is a registered trademark of the SDRC corporation.

for capturing such dependencies. However, they suffer from two crucial drawbacks – a distributed team cannot easily visualize the parametric dependency and the process of creating a constraint is hidden from other users. Use models implemented in DOME rely on a basic CAD parametric model however the difference is that the key parameters are explicitly linked to the DSM Process view. Thus all participants can clearly see that certain parameters are related to others and also who in the process is responsible for ensuring that they meet the requirements.

Apart from geometric design parameters there is other type of structured information that is exchanged during the PDP. This includes material properties, test results data, analysis data etc. All these are modeled as objects in DOME with data and services that can be linked to each other.

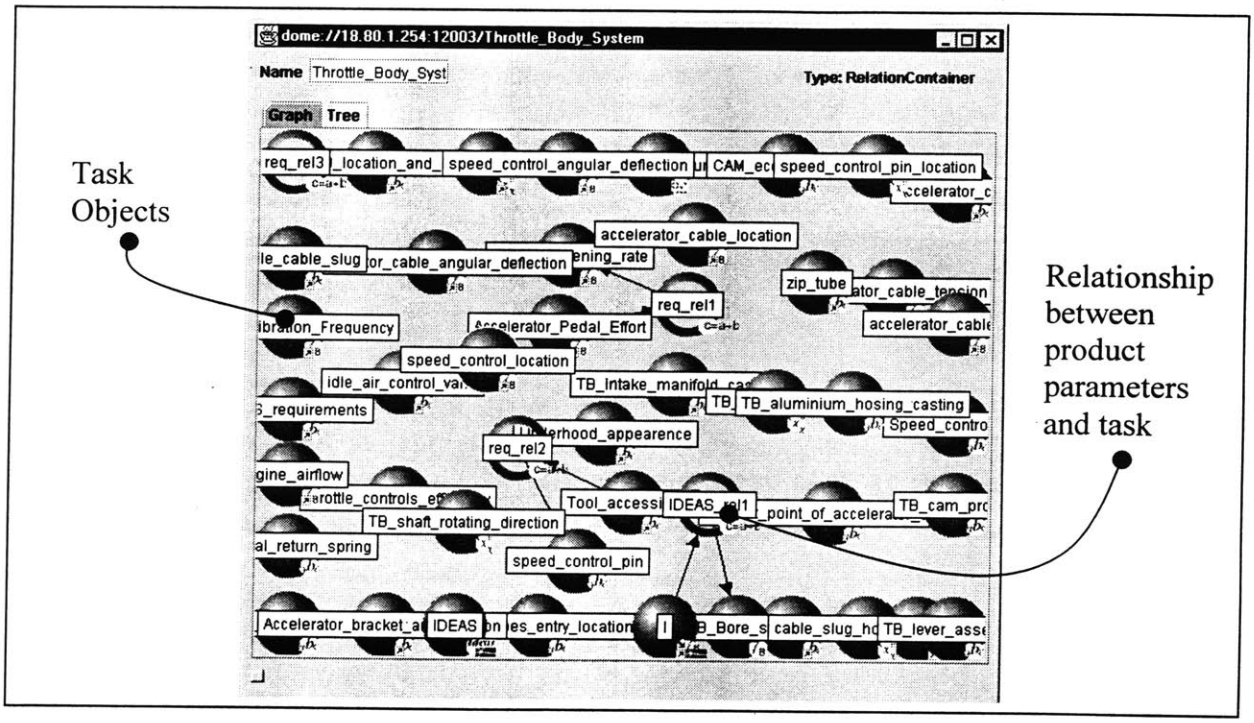


Figure 5-1 DOME view of the object network for throttle body design

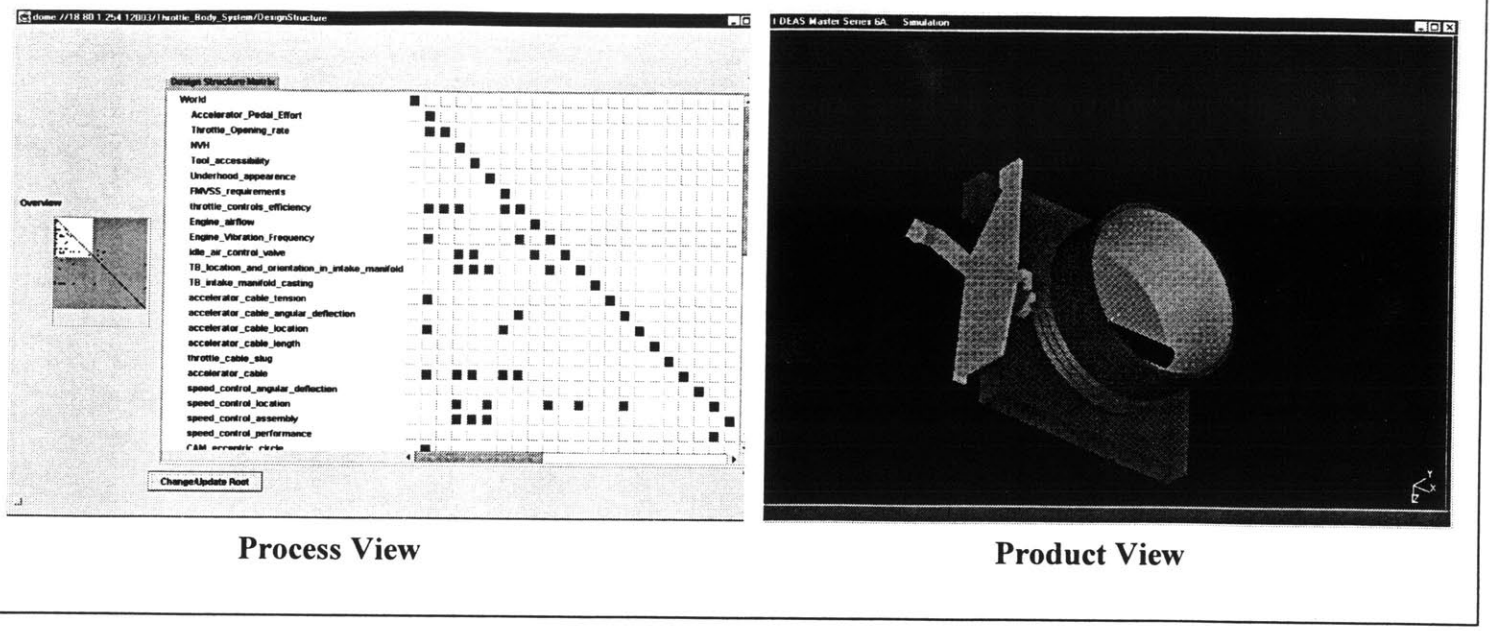


Figure 5-2 The Use model view for the throttle body

2. Unstructured – A significant part of the PDP deals with unstructured data such as emails between participants, subjective views expressed in other documents on various design and organizational choices made etc. There are two major limitations of current product models as implemented by CAD/PDM systems. One is that in many cases non-structured data is not captured at all and the other is that the documented design guides are not linked to the IT enabled process. Therefore even though design guides are made available on the Intranet, the particular context in which these have to be used is not clear to an inexperienced participants in the PDP.

In a use model the DSM process view captures these unstructured relationships at a greater level of detail. For example in the throttle body case the under hood appearance task which is a top down requirement can be linked to a web page that discusses past experience and guidelines. By providing a suitable context the DSM greatly enhances the value of the existing documented knowledge base. An implementation in DOME can link these relationships to not only Intranets but also other communication systems such as email, document management systems etc.

In summary both the structured and unstructured interactions can be modeled in DOME (or a similar) software. The structured data exchange is automated and creates a system model that can be evaluated and optimized. The non-structured information can be used to guide participants through a combination of emails, web page updates etc.

- **Knowledge management and representation** – The process of implementing a Use model consists of a cycle of creating a DSM, mapping it to a use model and implementation. This process creates a knowledge base that can be used and refined in subsequent projects. The DSM representation provides a succinct map of the knowledge that goes into that particular system design task. The underlying methods and tools provide a baseline to improve upon.
- **Workflow and project management** – The as executed Use model can provide useful data for further process re-engineering and improvements. Two kinds of changes are possible. New marks (dependencies) in the DSM might emerge. Also new tasks might get added. In both cases since the users can change the process the IT system will quickly reflect these improvements. The project learning will therefore get more easily incorporated into tools.
- **Collaboration in distributed teams** – The use model implemented in DOME is inherently multi-user. Each separate sub-system was implemented as a separate model. A different team can work simultaneously on each of these models. (Figure 5.3). The subsystem themselves are subdivided into objects that can be shared. The combination of publishing and subscription mechanism ensures that team members have the flexibility of trying options before they impact the complete system model.

- **Leveraging existing IT assets** – The Use model created for the throttle body is based on services of existing IT systems such as CAD. A product data management (PDM) system, the other widely used application, can provide services like storage and retrieval, change management and requirements tracking. All these form an important part of a Use model in a commercial setting where managing huge amounts of data requires established enterprise database systems. The key value added by DOME based Use models are the creation of detailed process models that link the existing services. This object-based network of data and services is something that is not supported by current IT tools.
- **Resource View and Schedule view** – These are the two views that were only partially implemented but are part of the Use model framework [Appendix C shows an example]. In an industry setting the resource view would be based on the roles and responsibility chart. It would show the links between tasks and responsible team members. The schedule can be a simple Gantt representation. The four views together provide a complete picture of the PDP. They can be used to create derived views such as resource utilization, expected versus actual finish times etc.

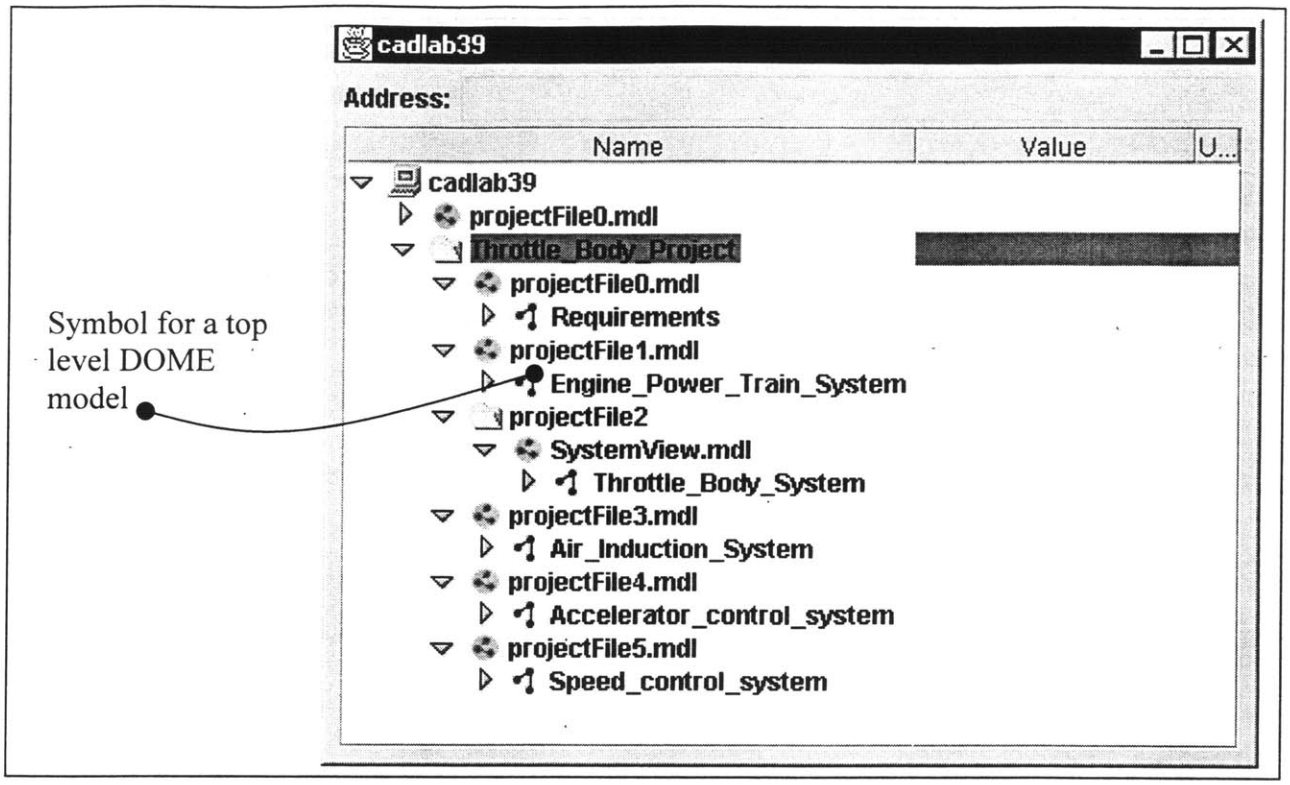


Figure 5-3 Use Model as a collection of DOME sub-system models

5.2 Use model for restraint systems

The restraint system is co-designed by the Auto Company with suppliers. [Lavine] describes a study of the interactions between the principal participants. This system differs from the previous case study in the limited number of tasks and interactions. The key factors that make this use model unique are summarized here –

- There were four main participants in the restrain system study – Full service suppliers (FSS), engineering personnel from restraint system design and engineering and others from manufacturing.

- Project delays were caused by inconsistent statements of work for the various participants so, responsibilities had to be continuously negotiated. It is clear that one of the foremost requirements for this use model is to develop the correct and complete statement of work and ensure that everyone has access to it. A top down system engineering approach is suitable here. If the product development process is seen as marketplace guided only by the participants then such coordination becomes difficult. This is one of the pitfalls of the design-marketplace school of thought.
- Design manuals and specifications did not provide sufficient levels of detail for developing a set of tasks. This confirms the observation in several other DSM efforts that most traditional documents and methods do not capture the amount of knowledge that goes into a typical product development process. A corollary of this observation is that IT solutions based on insufficient process knowledge are not very helpful and in some cases might even reduce productivity.
- Changes in organization structure are frequent and these were not reflected in the design manuals. Such changes need to be part of the use model implemented as an integrated computer model. As described in Chapter 4 the resource view of a use model tracks changes in the roles and responsibilities of participants in the PDP.
- The vehicle package team, FSS and CAD group performed several activities independently and inconsistently leading to rework. Lack of a well-defined process map and appropriate tools were the primary causes. This resulted in multiple versions of the seat belt anchorage regulatory zone. Vehicle package created it with the vehicle architecture in mind, while suppliers created it to develop mounting brackets, and CAD group developed them to monitor regulatory compliance. Rework created by inconsistent activities can be greatly reduced by existing solutions such as PDM, which provide a single database for all. However, what these tools lack is the concurrency and collaborative layer[16] required for a distributed team. The DOME software provides this added functionality.

[16] The distributed collaborative functionality claimed by most commercial applications provide file level support. They do not support the parameter or feature level details being discussed here.

- Informal channels of communications are common and these are typically focussed on getting the job done rather than following a process. This in itself is not harmful but it has a bad side effect in that it bypasses the learning effect. Knowledge created in the process gets lost. A DSM based approach which tracks as well as guides interaction between groups can go a long way towards capturing interactions that forms a big part of the system level knowledge. Some interactions will perhaps always remain outside the IT system but as compared to the current situation many more would get documented.
- The most common form of communication was informal emails or voicemail. These are of limited use as references for the data. They do not reveal where the data originated from and where it is maintained (the master source).
- Out-dated information was identified as a significant cause of re-work and this problem was addressed by associative modeling in CAD. This solution may have worked well in this situation but it does not scale up to more complex situations. Even when it does work it is hard for a new participant to understand a complexly associated model because the inter-connections cannot be easily visualized. Also if the number of FSS and other parties involved increase, a full associative CAD model becomes even harder to share with a team.
- The pure DSM methodology has a weakness in that incorrect information created because of an error during its generation is difficult to fix. The Use model concept addresses this limitation by including the flexibility of updating the DSM based on actual interactions.

5.2.1 Discussion of the Model

Figure 5.4 shows a complete view of the restraint design process. This is based on the optimized DSM recommended by [Lavine]. The use model implemented for this project did not have a product view (data was not available). The same process as for the previous case was followed for creating the DOME model. [Lavine] describes the solution that was adopted by the company. It is based on re-engineering the DSM and making the requirements available early on and presenting the users with a catalog to select the seat belt design features. This solution is a subset of the use model approach. In certain cases, such as this, when the primary needs are coordination, consistent specifications and assignment of responsibilities between suppliers such an approach works. In other system design projects where subjective performance evaluation is needed to make complex trade-offs the same strategy may not work.

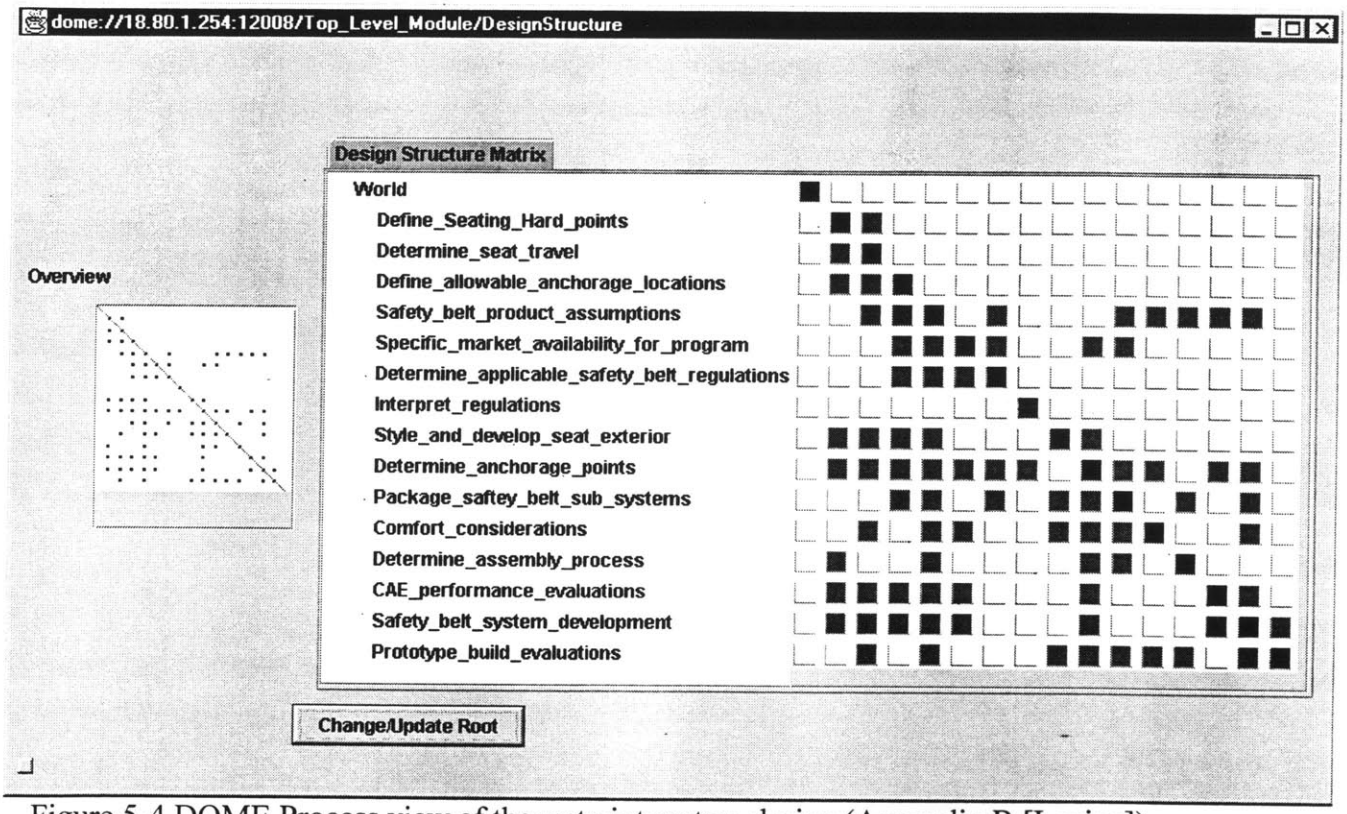


Figure 5-4 DOME Process view of the restraint system design (Appendix B [Lavine])

5.3 Use model for hood design

Hood sub-system design is a complex problem. The complexity arises because of many interfaces of the hood system with the rest of the vehicle designers, safety engineers, CAE analyst, styling, and body side people. The overall packaging requirements are also important and are driven by global vehicle hard points. The use model for hood design therefore depends much more on top down requirement compliance and making complex trade-offs between them. This is different from the earlier two cases, which did not have as many top down requirements. This process was also different in that it was aligned more closely with the system engineering methodology prescribed by the firm under study.

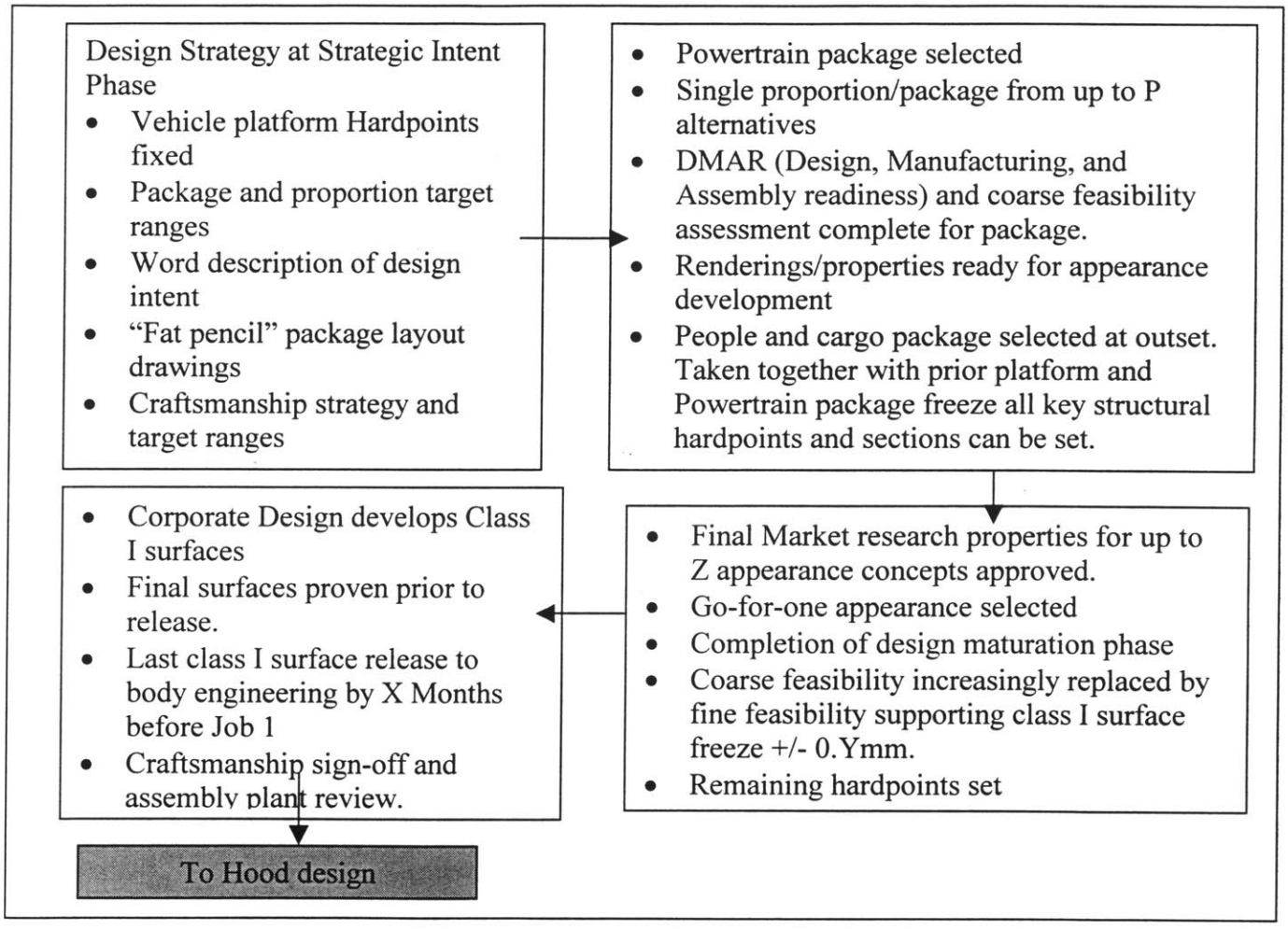


Figure 5.5 The prescribed top down hood design process

5.3.1 Top down approach to hood design

Figure 5-5 shows the vehicle design process leading up to hood design. As the design is refined more of the vehicle hard points are defined. These constrain the hood designer. Table 5-2 shows some of the top-level design goals for the hood. In addition to system level parameters there are several other design parameters specific to the hood. Table 5-3 shows the elements of the use model and classifies the top-level hood design requirements into three types – packaging, manufacturability analysis, and performance modeling. The goal of the Use model is to deliver these requirements. In order to do this, besides knowing the top design objectives, a detailed map of design process is required. A DSM view of the hood design process was therefore developed [Zambito].

Table 5-2. Goals for Hood design[17]

Goal	Metric (units)	Decision Criteria
Cost Target	System Cost ($)	Select a concept that costs lowest absolute variable and fixed cost
Weight Target	Absolute weight (lbs.)	Select a concept that offers the lowest absolute weight
Torsional rigidity specs	Rigidity/weight/cost (index number)	Select concept that offers highest performance index
Deflection (inertial) specs	Deflection/weight/cost index (index number)	Select concept that offers highest performance index
Complexity	No. of Parts (number) Ease of assembly (time, $)	Select concept that offers most efficient assembly and least parts
Quality	Repeatability (Failure rate)	Select concept that allows production with the most repeatable methods
Quality	QFD correlation index such as opening closing efforts (index number)	Select concept that best meets customer requirements.
Quality	Assembly variation (mm)	Select the concept that is least sensitive to assembly variation.
Safety	Energy absorption. (% Crash energy absorbed)	Select the concept that manages energy absorption safety

[17] Based on interview with Tony Zambito

Safety	Geometric considerations such as minimum radii. (mm)	Select a concept that offers best safety characteristics to occupants and pedestrians.

Table 5-3 Elements of the Use Model

Use Model		
Goal	Methods	Tools
Packaging • Weight Target • Complexity • Geometric considerations such as minimum radii. (mm) • System Cost	• 3D Solid and surface modeling • Virtual Mockups • Standard master assemblies and parts • Design guidelines	• CAD • Visualization software • PIM • Intranets • KBE
Manufacturability • Repeatability (Failure rate) • Assembly variation	• Process capabilities catalogs, knowledge base • Connective models of assemblies • Assembly sequence, variation analysis	• CAD/PIM • KBE • DFC based analyzer • CAM • Email/Phone experts
Performance (Objective & Subjective) • Energy absorption. • Rigidity/weight/cost • Deflection/weight/cost • QFD correlation index	• Fit & Finish – 3D visualization, styling • Thermal, stress, aerodynamic performance • System performance (feel of car, ease of use in consumer electronic etc) • Environmental impact	• Styling software • CAE • CAM • Simulation software

5.3.2 DOME model and discussion

The Use model for hood design was different from the other two in the breadth of interaction between disciplines. Marketing requirements, business decisions, design issues and manufacturing capabilities are all inter linked in a complex manner. This makes the use model more general than that for the restraint system and throttle body which focussed primarily on design compliance. The common requirements of

coordination, performance evaluation etc are evident in this case as well. Some observations from the hood design use model that are different from the previous cases and are general enough to be applicable in other situations as well are –

- The hood design DSM is large, complex, and not a lower triangular matrix. This means that the design has a lot of iterations and requires complicated negotiations between different groups. A DOME use model helps in two fundamental ways – support for complex math models that can be used for decision-making and reducing the asymmetry of information between groups. The last point is especially important when the decisions are subjective. The research for this thesis at a large automotive company revealed that many interactions between groups got bogged down in non-value added "gaming" situations. By increasing the information available to all parties use models can help in guiding the discussions towards the real issues and in making better decisions.
- Product development process in the large requires the integration of many disciplines. Many decisions are based on non-engineering criteria. In the case of the hood for example the selection of materials, the target market and production rates are all dependent on each other. The use model allows for linking of such diverse models through the objects and services paradigm. Such an integrated model represents a more accurate system model.
- A significant number of tasks in the hood use models are supported by existing solutions like CAD and PDM. The use model leverages these and creates a richer representation by including other tasks and relationships which are traditionally seen as outside the scope of the IT system for product development or are not documented at all.

6 CONCLUSIONS

6.1 Summary

This thesis developed a framework for implementing IT solutions for product development. It defines a new concept called "Use models" that is focussed on how designers, engineers and managers use different methods and tools to accomplish complex system design. This user-centered approach was contrasted against the predominantly tools oriented and data modeling based approach in existing IT implementations for product development. The Design Structure Matrix (DSM) was used as a design process engineering, modeling and knowledge management tool. A method was developed for mapping the product development process to an object-oriented network of methods and tools. This combination of a design map, software and other tools form the basis of the use model.

As an example, use models were developed for three complex systems from the automotive product development process – hood design, restraint systems, and throttle body. Developing efficient use models for these sub-systems revealed many of the broad issues that IT systems for product development must address. These are the need for–

- Support for an object based modeling environment that can link to existing IT solutions through services. These services should include PDM, CAD, workflow, and analysis.
- Support for a Design Structure Matrix based process model
- Support for visual linking of services through relationships allowing participant in the PDP to create and change process models.
- Support for integration of design services through standard web interfaces
- Support for resource and schedule views in addition to process and product views.
- Support for open and easy collaboration through a "publish and subscribe" mechanism.

Also, since the overall vehicle development process has several use models, developing one highlighted a common set of IT capabilities required for developing others. One of the results of this thesis is the generalization from these use models and definition of a framework that can be utilized in other complex system design contexts. The framework has two main parts –

- A methodology for implementation of use-models. This includes the first step of documenting the DSM, then mapping it to an object based network and finally implementation.
- The elements of the Use model that can be common across different product development processes – process views, object based modeling, and standard web based software for implementation.

The feasibility and utility of the use model concept was tested through implementation in the Distributed Object-based Modeling Environment (DOME) software developed at MIT. These use models created in DOME present the participants in the product development process with a 3D-product view and a process view in the form of a DSM. These computer models help in capturing the system and sub-system level design knowledge in new ways that are not currently available. Since the focus of this research was complex system design and the special nature of the design process, the thesis takes an integrative approach that combines elements of system engineering, knowledge management, integrated product and process modeling and organizational theory.

6.2 Future work

- **Use Models and the PDP**. The use model approach recommends a different approach along three features. First is the focus on the participants in the development process rather than on the data. Second is the higher level of detail at which design knowledge is represented. And the third important element of the Use model framework is the emphasis on learning over time. All these difference require further investigation. For example, with every iteration of the design process new methods become apparent and the use model framework provides a mechanism for incorporating these into system models. The mechanism would need to be formalized in order to prevent non-value-added activities from creeping in to the use model. Research is required in deciding on the right amount of control that maintains the process quality without overly constraining the users.

- **Creating and comparing use models.** Once use models are implemented and used in production for a particular system design project they will provide a rich set of data for analysis. Comparison can be made between projects and also between different methods and tools applied to the same task. An important area of future work would be developing metrics for comparing the performance of use models.

- **Abstracting basic building blocks from successful use models.** An area related to the benchmarking of use models is the idea of abstracting basic building blocks from successful use models. This is similar to the notion of re use of software components in object oriented programming. Those objects in a use model that prove to be robust and high performance can be plugged into new projects with a high degree of reliability. A good example would be a cost analysis module. Given a part, a standard cost analysis module can look up a database for information and compute the actual cost based on part geometry or other data. Assuming that for a large number of parts the same variable (like vendor or weight) determines the cost a standard object for cost analysis can be created. New use models can just reuse this without alterations. Similar proven modules can greatly improve the speed of implementation of use models.

- **Production roll out in commercial setting.** A big next step for adopting Use models is the challenge of implementation and use in a commercial setting. Like any other software application that is process sensitive the Use model approach will need significant change management and training effort. The examples in this thesis were all based on case studies of actual systems in a large automotive company and therefore the scale of models was realistic. However, their utility and performance under a multi-disciplinary team with time pressure and in a hierarchical organizational context remains to be seen. This will undoubtedly present some new issues and problems that will have to be addressed. Another implementation issue is the need for maintaining the validity of the model as multiple users change relationships. A locking scheme, some guidelines to team members and other techniques would need to be developed to prevent inconsistent simultaneous changes.

- **Learning**. A key element of use models is the ability to incorporate learning as projects are executed. The primary mechanism for this is currently through participants who can easily make process changes and attach new improved models to the system model. In future, automated learning algorithms could be used to analyze interactions between project participants. These could analyze information such as frequency of interactions, extent of interactions, number of participants etc.

- **Technology for implementing Use Models.** The DOME software that was used to implement Use models provided the ability to create and link objects. It also supplied the necessary software for communicating with applications like CAD, VRML etc. A similar approach is required for commercial installations. The technology is based on web standards and therefore scalable and open. More work is needed in the software in the following areas –
  1. Visualization of large models
  2. Better support for process modeling. For example, creating relationship between objects through drag and drop
  3. Two way integration with PDM systems
  4. Searching for info, resources, tasks that are alike.

5. Support for Email clients, customized emails, linking to corporate database of roles and responsibilities.
6. High level visual language for encoding special rules.

- **Internal and external supply chains for complex systems.** Even though the case studies were from the automotive industry the lessons learnt are applicable in complex design situations in other industries such as aerospace, consumer electronics etc. An important area of development in all these cases is the outsourcing of complex systems. Unlike outsourcing of components, system outsourcing is complicated because of the difficulty of developing complete specifications and of evaluating competing solutions. A suitable use model for such a situation would resemble an electronic market place that has DOME like capabilities. It would act a distributed spreadsheet that can evaluate sub system and system level performance for a variety of variables. Figure 6-1 shows the need for creating such a market place –the value transaction in the engineering system supply chains is far greater than of commodity items that are being traded on such exchanges today.

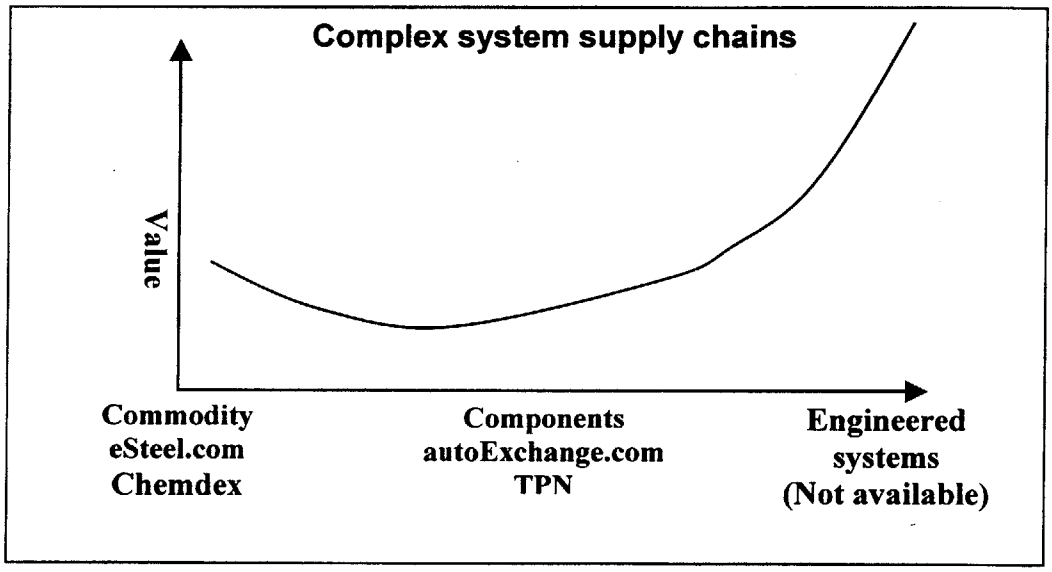
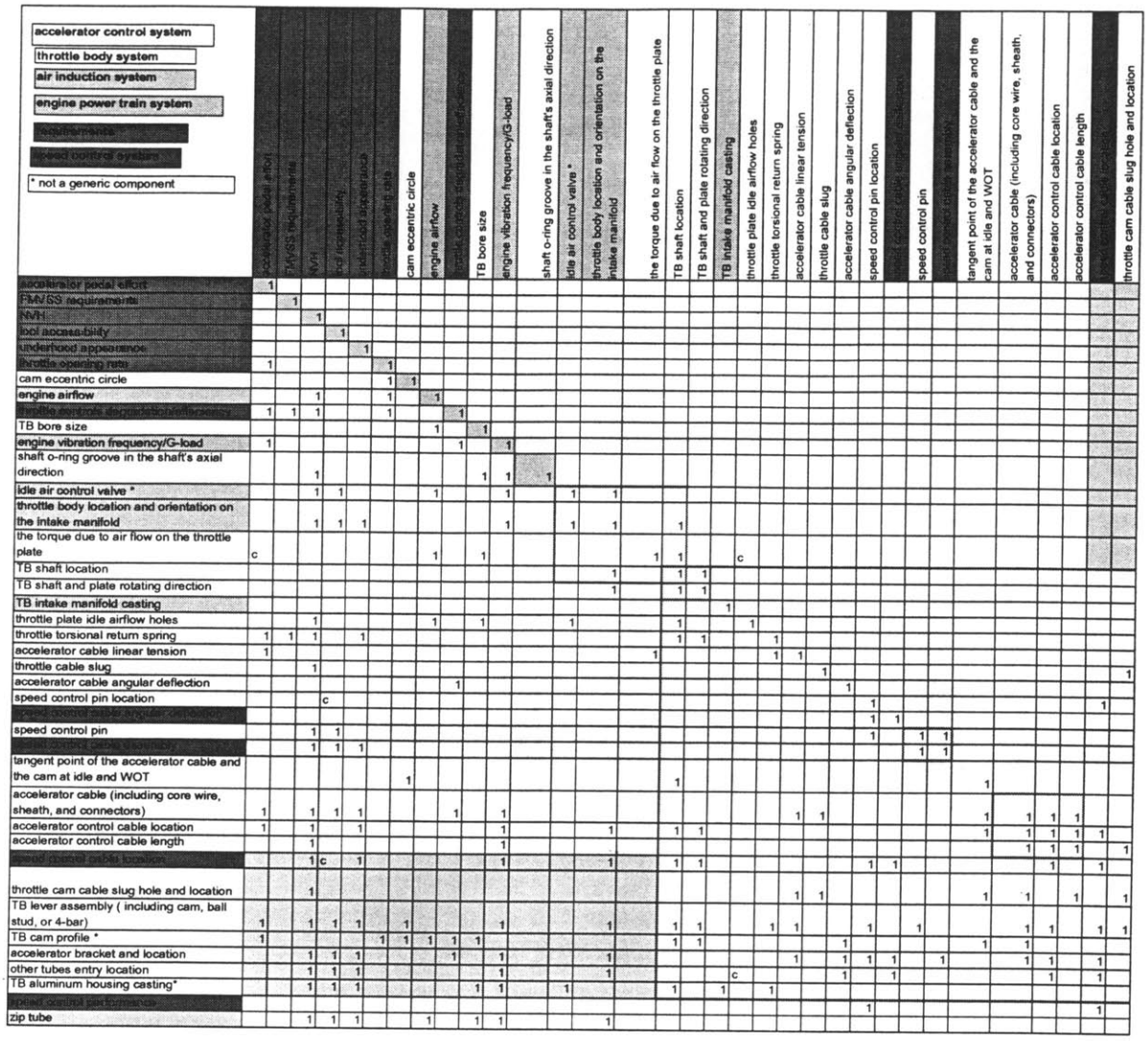


Figure 6-1 Electronic market places for complex engineering systems[18]

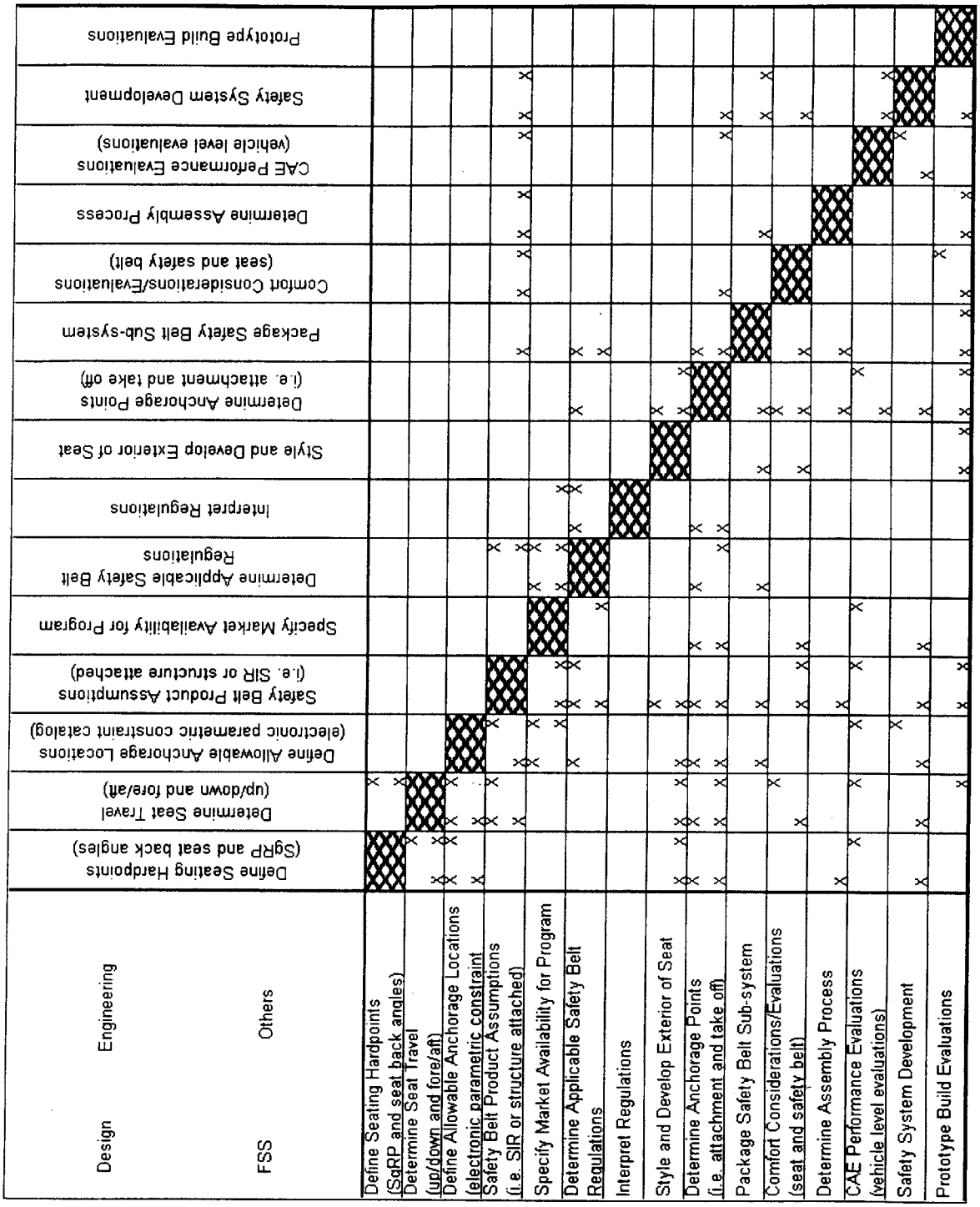
[18] Not based on actual data

APPENDIX A – THROTTLE BODY DSM [QI]



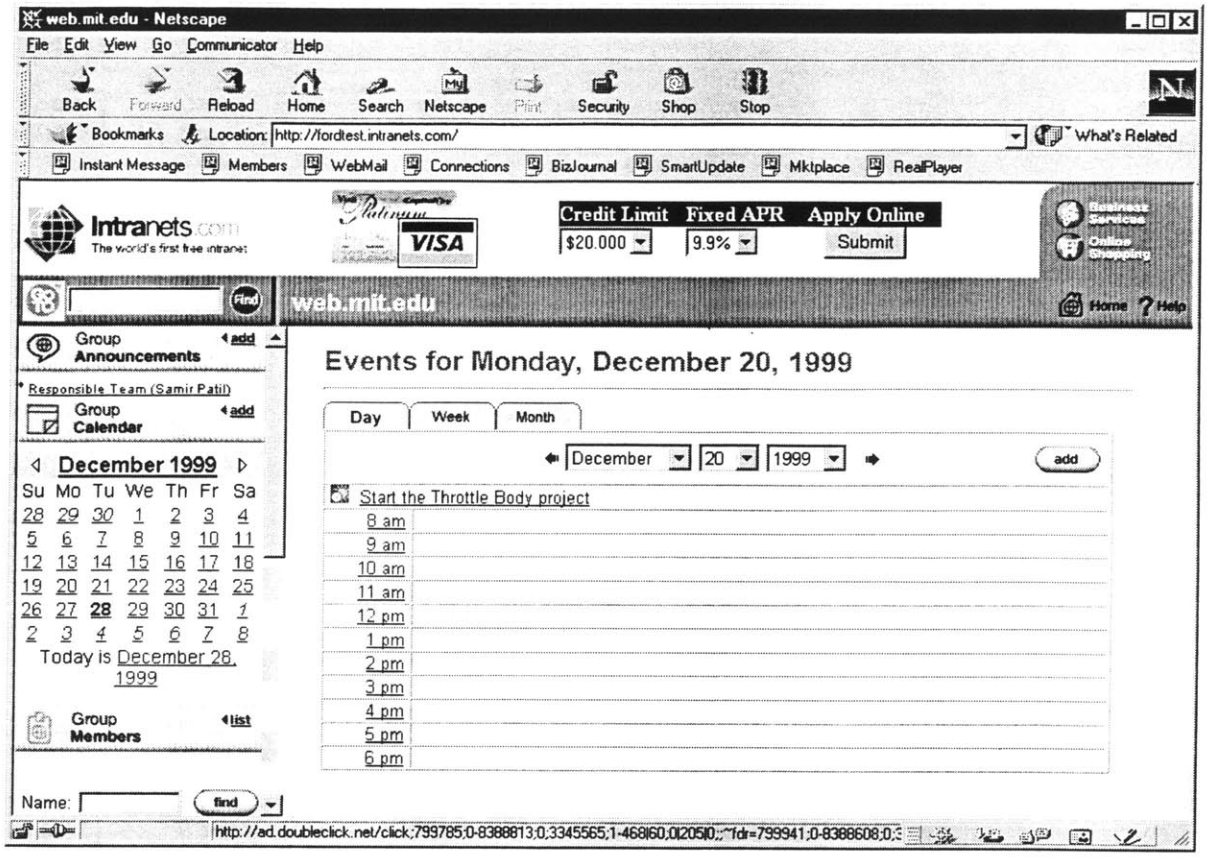
Throttle Body DSM developed by [Qi]

APPENDIX B RESTRAINT SYSTEM DESIGN [LAVINE]



Restraint system DSM developed by [Lavine]

APPENDIX C RESOURCE AND SCHEDULE VIEW



- A simplified resource and schedule view
- Based on a standard web based e-service by Intranets.com.
- This service can keep track of team members, documents and calendar. Such a view linked to the object model can help managers track projects

7 REFERENCES

Abbenathy William J., James M Utterback (1978) "Patters of industrial innovation" Technology Review, June/July

Abrahamson (1999) "Tools for decomposition in an integrated modeling environment", MIT Masters Thesis.

Alexander (1964) "Notes on the Synthesis of Form". Harvard University Press, Cambridge Massachusetts and London England.

Allen Thomas (1990) "The Influence of Communication Technologies on Organizational Structure".

Arturo Molina, A. H. A. -A., Timothy IA Ellis, Robert IM Young and Robert Bell (1995). "A Review of Computer Aided Simultaneous Engineering Systems" Research in Engineering Design 7: 38-63

Banares-Alcantara (1997), R. "Representing the engineering design process: two hypotheses". Computer-Aided Design

Bobby Cameron (1998), "New Life for PDMs - Packaged Application Strategies" Volume Three, Number Nine, Forrester Research

Branki, C. N. (1995). "The Acts of Cooperative Design." Concurrent Engineering: Research and Applications 3(3): 237-245.

Cleetus, K.J (1995) "Modeling evolving product data for concurrent engineering" Engineering with Computers v 11 n 3

Cusumano M, Selby, (1995) "Microsoft Secrets"

Cutkosky, Olsen, Tenenbaum, Gruber,(1994) R. "Collaborative Engineering Based on Knowledge Sharing Agreements" Procedings of the 1994 Database Symposium.

Cutkosky, M. R., R. Engelmore, et al. (1996). "PACT: An Experiment in Integrating Concurrent Engineering Systems." IEEE Computer (January, special issue on Computer Support for Concurrent Engineering): 28-37.

De Pauw, Kimelman, Vlissides (1997) "Visualizing Object-Oriented Software Execution". MIT Press, Cambridge, MA, London, England. Editor John Stakso et al.

Dilip Wagle, (1998) "The case for ERP systems", The McKinsey Quarterly,

Number 2, pp. 130—138

Don Brown and Ken Versprille, (1997) "The OCAI Initiative Open CAx Architecture and Interoperability" CALS Expo INTERNATIONAL 1997 Proceedings

Egidi, M (1992) "Organizational Learning, Problem Solving and the Division of Labour" Economics, Bounded Rationality and the Cognitive Revolution, Edward Elgar Publishing, Vermont USA.

Eppinger S, K. R. M. a. S. (1993). "Managing the Integration Problem in Concurrent Engineering".

Eppinger S, R. P. S. a. S. (1997). "Identifying Controlling Features of Engineering Design Iteration." Management Science 43(3).

Eppinger S, D. E. W., Robert P Smith and David A Gebala (1994). "A Model-Based Method for Organizing Tasks in Product Development." Research in Engineering Design 6: 1-13.

Eppinger S, M. V. N. a. D. E. W. (1997). "Generalized Models of Design Iteration using Signal Flow Graphs." Research in Engineering Design 9: 112-123.

Funk, J.L., (1997) "Concurrent engineering and the underlying structure of the design problem" Engineering Management, IEEE Transactions on Volume: 44 3, Page(s): 305 -315

Harris, S.B. (1996) "Business strategy and the role of engineering product data management: a literature review and summary of the emerging research questions" Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture v 210 n B3, p 207-220

Erik Brynjolfsson and Hitt L (1998) Beyond the Productivity Paradox, Communications of the ACM

Gregory J Burek, (1997) An application for intelligent manufacturing and implementation of direct engineering. Ford Motor Company

Francis Phang, Nicola Senin, and David Wallace, (1998) "Distribution Modeling and evaluation of product design problems", Computer Aided Design, Vol 30 No. 6, pp 411-423

K. Henderson (1991). "Flexible sketches and inflexible data bases: Visual communications, conscription devices, and boundary objects in design engineering, "Sci. Technol. Human values, vol 16, no 4, pp 448-473

Huang, G.Q, Mak, K.L. (1998) "Re-engineering the product development process with `design for X' " Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture v 212 n B4, p 259-26

Kappel, T.A.; Rubenstein, A.H. (1999) "Creativity in design: the contribution of information technology " Engineering Management, IEEE Transactions on Volume: 46 2, Page(s): 132 -143

Kim, J. B. and D. R. Wallace (1997). A Goal-oriented Design Evaluation Model. ASME Design Engineering Technical Conference, DETC97/DTM-3878, ASME.

Kontong Francis Pahng (1998) "Modeling and evaluation of design problems in a network centric environment", Ph.D. Thesis, MIT

Lavine (1999) "Application of the DSM Methodology for Documenting, Evaluating and Improving a Design Constraint Management Process", Masters Thesis, MIT

Les Alberthal, (1998) "The Once and Future craftsmen culture ", The Electronic Marketplace

Love, P.E.D.; Gunasekaran, A (1997) "Process reengineering: A review of enablers" International Journal of Production Economics v 50 n 2-3 Jun 16 1997, p 183-197

Marco Iansiti and Alan MacCormack, (1997) "Developing products on Internet time", Harvard Business Review Article Publication Date: 5/1/97

Michelena, Papalambros (1995) "Optimal Model-Based Decomposition of Powertrain System Design". Transactions of the ASME Vol. 117.499-504.

Molina, A., A. H. Al-Ashaab, et al. (1995). A review of computer aided simultaneous engineering systems. Research in Engineering Design 7(1): 38-63.

Nath B; Gunasekaran, A. (1997) "The role of information technology in business process reengineering" International Journal of Production Economics v 50 n 2-3, p 91-104

Pahng, K. F., N. Senin, et al. (1997). Modeling and evaluation of product design problems in a distributed design environment. ASME DETC'97, Sacramento, California, ASME.

Qi Dong MIT Master's Thesis

Robert A Crabtree, M. S. F. a. N. K. B. (1997) "Case Studies of Coordination Activities and Problems in Collaborative Design" Research in Engineering Design 9(2): 70-84.

Rolf-dieter Kempis and Jürgen Ringbeck, (1998), "The use and Abuse of IT in manufacturing" The McKinsey Quarterly, Number 1, pp. 138—150

Sobek K Durward; Jeffrey K. Liker Allen C. Ward, (1998), "Another Look at How Toyota Integrates Product Development" Harvard Business Review Article Publication Date: 7/1/98

Sobek II Durward K. Allen C. Ward & Jeffrey K. Liker, (1999) "Toyota's Principles of Set-Based Concurrent engineering", Sloan Management Review Volume 40, Number 2

Sussman, G. J. (1975). A Computer Model of Skill Acquisition, Elsevier Scientific Publishing Company.

Thomas H. Davenport. (1998), "Putting the Enterprise into the Enterprise System", Harvard Business Review Article Publication Date: 7/1/98

Thomas Gormley J, III Stuart D. Woodring Ketty C. Lieu, (1997) "Supply Chain Beyond ERP - Packaged Application Strategies" Volume Two, Number Two, Forrester Research

Tomiya, Yoshikawa, Kiriyama, Umeda (1994). "An Integrated Modelling Environment Using the MetaModel". Annals of the CIRP. Vol. 43.

Ulrich, Eppinger.(1995) "Product Design and Development". New York: McGraw Hill.

Venkatraman N. (1991) "IT-Induced business reconfiguration, The corporations of the 1990s", Information technology and organizational transformation, 1991

Wayne Collier (1997) "An integral PIM strategy –Implementation roadmap" 1997 DH Brown Associates Inc. (September 9-11, Dearborn Michigan)

Wayne Collier (1997) Product Vs Process in Design-to-Manufacture Integration Implementation roadmap 1997 DH Brown Associates Inc. (September 9-11, Dearborn Michigan)

Whitney D.E, Qi Dong, Jared Judson, and Gregory Mascoli, (1999) "Introducing knowledge-based engineering into an interconnected product development process"

Zambito T MIT Masters Thesis