INTEGRATING SUPPLIER DESIGNED COMPONENTS INTO A SEMI-AUTOMATED PRODUCT DEVELOPMENT ENVIRONMENT

by

Jared Alden Judson

B.S., Mechanical Engineering, State University of New York at Buffalo, 1988

Submitted to the Sloan School of Management and the
Department of Mechanical Engineering
in Partial Fulfillment of the Requirements for the Degrees of

Master of Business Administration
and
Master of Science in Mechanical Engineering

in Conjunction with the
Leaders for Manufacturing Program

at the
Massachusetts Institute of Technology
June 1998

Signature of Author: _____
Sloan School of Management
Department of Mechanical Engineering
May 8, 1998

Certified by: _____
Janice Klein
Sloan School of Management
Thesis Advisor

Certified by: _____
Daniel Whitney
Department of Mechanical Engineering
Thesis Advisor

Accepted by: _____
Lawrence S. Abeln
Director of Master's Program
Sloan School of Management

Accepted by: _____
Anthony T. Patera
Acting Chairman of the Graduate Committee
Department of Mechanical Engineering

INTEGRATING SUPPLIER DESIGNED COMPONENTS INTO A SEMI-AUTOMATED PRODUCT DEVELOPMENT ENVIRONMENT

by

Jared Alden Judson

Submitted to the Sloan School of Management and the
Department of Mechanical Engineering on May 8, 1998
in Partial Fulfillment of the Requirements for the Degrees of
Master of Business Administration
and
Master of Science in Mechanical Engineering

ABSTRACT

This thesis presents a demonstrated method for achieving rapid, iterated product development between an automobile manufacturer and a component supplier. The work occurred in the context of a semi-automated design process being developed at Ford. The thesis also discusses the organizational and cultural context of this new product development process and their implications.

Designing an automobile demands the coordinated design of thousands of parts. Systems engineering proposes that this challenge can be simplified by decomposing the car into systems, sub-systems and components: manageable pieces which are designed individually and then reintegrated into the whole, a process repeatedly iterated until final design approval. Complications arise when suppliers design some parts, and engineers must manage supplier communication at each iteration.

To meet pressure for increased speed, a Ford group is instituting their view of a "next generation" design process, a semi-automated software-based methodology which embodies knowledge management and systems engineering. This thesis addresses a crucial next step: integrating suppliers into this new, faster process. Working with a team applying the new design process to a Throttle Body, the research culminated in the successful demonstration of a method for integrating supplied part design into assembly design.

The technical and cultural setting for this work was also studied. Organizational issues influencing immediate acceptance as well as the longevity of the proposed process change were assessed using a Core Capabilities/ Rigidities framework. Introducing the new semi-automated design and communication process delivered an order-of-magnitude reduction in design iteration turn-around time. However, the complexity of the software tools may impair long-term adoption. And the process demands many changes to the existing product development culture, resulting in significant resistance to this innovation in product development.

Thesis Supervisors:

Janice Klein

Title: Senior Lecturer, Sloan School of Management

Daniel Whitney

Title: Senior Research Scientist, Department of Mechanical Engineering

ACKNOWLEDGMENTS

The author gratefully acknowledges the support and resources made available to him through the MIT Leaders for Manufacturing Program, a partnership between MIT and major U.S. manufacturing companies.

In addition, I would like to thank the many people I worked with during my internship for their assistance and flexibility in accommodating my research. Kevin Deacon, Scott Lehtonen, and Dave Wise at Ford/Visteon played a central role in incorporating the ideas for this thesis into an actual process and making it work. Bob Humphrey at Ford offered moral support and aided in the testing and prove-out of the process. Gerald Baker and Bob Pavlin at Michigan Spring were open to trying something new and always found the time to work with me. Their interest and active participation made the supplier component of my research a pleasure. And I must thank the DIRECT ENGINEERING℠ management team at Ford's Advanced Manufacturing Technology Development center. My supervisor, Pete Sferro, Greg Burek, Ken Kuna, and Sean O'Reilly all offered support, advice and an audience for new ideas. Ken deserves special thanks for reviewing my ideas in detail. I wish everyone at Ford and Michigan Spring the best of success in the years ahead.

I also would like to acknowledge and thank my advisors at MIT, Janice Klein and Daniel Whitney. Their advice and insight played a much-needed part in clarifying and explicating the ideas herein.

I must express my appreciation and gratitude for the understanding and helpfulness of friends and family who've put up with the focus needed to complete this work. Thanks to all- I'm in your debt.

And I would like to extend a special acknowledgment to my wife, Judith, whose encouragement and flexibility throughout my participation in the Leaders for Manufacturing program has been extraordinary. She has endured relocation, geographic distance, no little amount of stress on my part, and many late hours of studying and writing with patience and continual support. Thank you.

TABLE OF CONTENTS

TABLE OF FIGURES

1. INTRODUCTION

The problem this thesis addresses is how to realize the speed advantages of a new product development process when a significant part of the product is designed outside the immediate organization. The solution presented is a demonstrated method for implementing rapid, iterative product development between an automotive customer and a component supplier which embodies a systems engineering approach. The method was created and tested over the course of an internship at the Ford Motor Company. A team of engineers and managers at Ford have been developing and deploying a proprietary "next generation" semi-automatic product development process and had identified the need to integrate suppliers and supplied components into their new process.

For this research, a candidate supplier was chosen, and a method for integrating the supplier into the new product development process was developed and demonstrated. The method, which is the focus of this thesis, is sensitive to the technological and cultural setting and the differences in skills, capabilities, and tools available to the customer (Ford) and the supplier. While the method was developed within the new proprietary process, the writer believes the results are generally applicable to other supply-chain product development relationships.

In addition to the supplied-component development process, the research revealed many organizational and cultural issues which impact the proposed change from current product development practice to the "next generation" process. These are discussed throughout.

This thesis is organized as follows. The remainder of Section 1 outlines the organizational setting, the product used to demonstrate the supplier process, and the new semi-automatic product development process at Ford. Section 2 presents a short summary of assembly design and systems engineering, and asserts that this methodology belongs at the core of the "next generation" process. Section 3 then discusses how supplier-designed components can be integrated into systems engineering and the product-development process. Section 4 describes the demonstration of the integrated supplied-component development process between Ford and the candidate supplier. An assessment of the types of supplied components to which this process can be applied is presented. Section 5 discusses the observed organizational and

cultural issues that will influence the long-term success of the new semi-automatic product development process. Section 6 concludes with some suggestions for future research.

1.1 ORGANIZATIONAL SETTING

The research that led to this thesis was conducted during an LFM internship at the Rawsonville, Michigan plant of Visteon, an enterprise of the Ford Motor Company. The LFM (Leaders for Manufacturing) Program is a partnership between twenty U.S. manufacturing firms and the Massachusetts Institute of Technology. Visteon was announced during the internship, bringing all Ford's components divisions under one corporate banner. The Rawsonville plant designs and manufactures a variety of automotive components, including alternators, fuel rails, intake manifolds, and the subject of this thesis- Throttle Bodies.

1.2 OVERVIEW OF PRODUCT STUDIED

This thesis presents a product development methodology using a Ford/Visteon Throttle Body and Throttle Return Spring as an example. Both are shown in Figure 1-1. The Throttle Body is in essence a butterfly valve operated by the driver's foot that modulates the flow of intake air to a car's internal-combustion engine. The Throttle Return Spring provides a closing

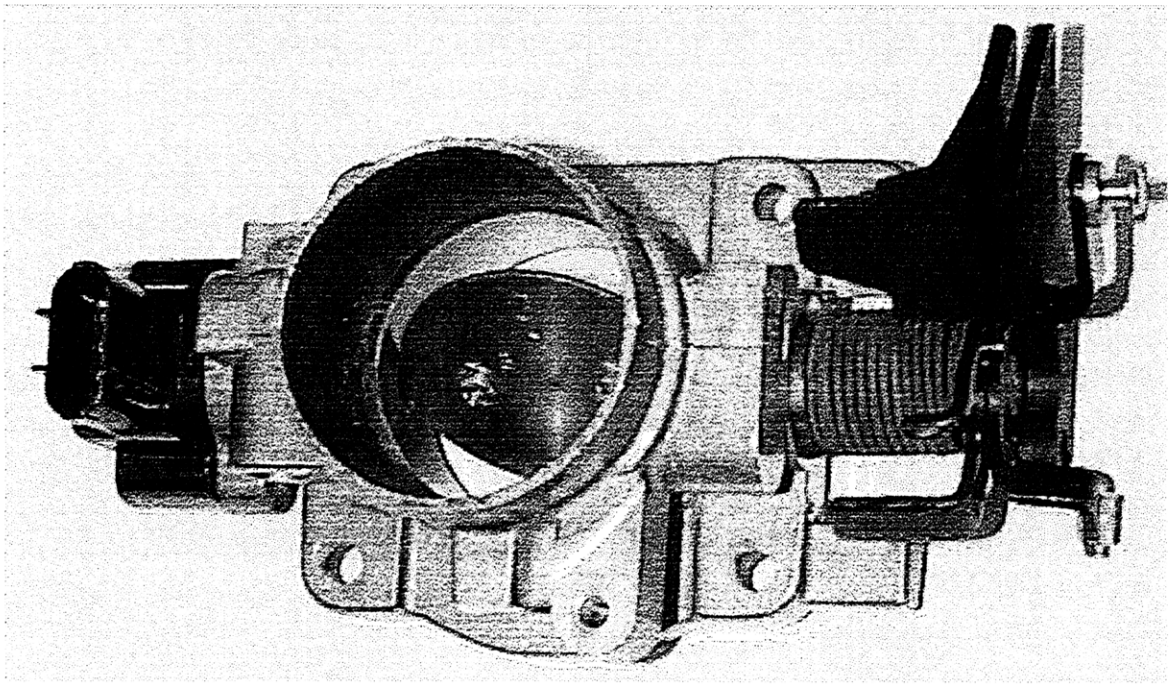

Figure 1-1 Throttle Body, shown partially open (Throttle Return Spring is the coil to right)

torque that shuts this valve as the driver's foot relaxes.

All Throttle Body components are assembled into a die-cast aluminum housing. Inside this housing, the butterfly valve is comprised of a round, flat plate supported across its center by a shaft. The plate fits snugly in a round bore machined through the aluminum housing; as the shaft and plate rotate from the closed position, the cross-sectional air flow area gradually increases. The shaft rotates in bearings which are pressed into the housing at either side of the plate. One end of the shaft is attached to an input lever, typically a cable-operated cam. The other end of the shaft engages a rotary potentiometer- the Throttle Position Sensor- which provides throttle plate position (i.e. throttle opening) signals to the electronic engine controller.

The Throttle Return Spring surrounds the shaft between the housing and input lever. This torsion spring contacts the housing and lever, and is wound to produce a torque on the shaft which closes the butterfly plate. This is an important safety feature, insuring that the engine can not "run away" if it becomes detached from the accelerator pedal input. The return spring also contributes to the load pushing the accelerator pedal back at the driver's foot.

The Throttle Body controls the amount of air traveling into the internal-combustion engine. When the driver presses the accelerator pedal, a mechanism (most commonly a cable) transfers this motion through the input cam to the shaft, which rotates and opens the plate to admit more air to the engine. In modern internal-combustion engines, injectors deliver fuel in proportion to the air flow reaching the cylinders. Therefore, overall vehicle throttle response is controlled by the Throttle Body and its design influences the "performance feel" of the entire car. The "performance feel" of the car also depends on accelerator pedal effort, which is influenced by the Throttle Return Spring's preload and spring rate.

At Visteon's Rawsonville plant, Throttle Bodies are always custom-designed for each new engine, and are commonly modified for each unique combination of a given engine and car platform. The Throttle Return Spring, designed by a supplier named Michigan Spring, is similarly adapted to each new vehicle and/or engine application. Because the car's "performance feel" often isn't known until pre-production builds are completed, Throttle Return Spring design changes can occur very late in the car design process.

As is discussed in later sections of this thesis, Throttle Bodies are currently designed in a traditional engineering process. But prior to and during the internship, Ford and Visteon engineers have been applying a "next generation" semi-automatic product development process (described below) to the development of Throttle Bodies. Introducing this process will dramatically increase Throttle Body development speed. However, the Throttle Return Spring is tightly integrated within the Throttle Body, so speeding up Throttle Body design necessitates speeding up the development process for this supplied component. This thesis presents a demonstrated approach for accomplishing this goal.

1.3 FORD'S SEMI-AUTOMATED PRODUCT DEVELOPMENT ENVIRONMENT

A group of managers and engineers at Ford are developing and applying a vision for how products will be designed in the future. They call their concept for a semi-automated knowledge-based product development process the DIRECT ENGINEERING^{SM} process (or, the DE^{SM} process)[1]. The details of the DE^{SM} process are considered proprietary by Ford, and are not a focus of the thesis. However, its development led to the need for this thesis, so it will be helpful to discuss in general terms what the DE^{SM} process is about.

The DIRECT ENGINEERING^{SM} process's definition is in flux, as continued learning and development have influenced the Ford team's concept. It continues to develop. The vision is quite broad, as shown by the statement below. This vision for the DE^{SM} process is perceived (and marketed within Ford) as being more than just a tool- it is a process to capture, manage and develop knowledge of products and their production environment, with the goal of enabling rapid design of variant components which are fully compliant with the constraints of all systems that interact with a product- including the product's surrounding assembly and the manufacturing process environment. The DE^{SM} process presents a major organizational shift in terms of engineering roles, activities and focus. Section 5 describes and assesses this impact.

---

[1] DIRECT ENGINEERING^{SM} and DE^{SM} are Service Marks of Ford Motor Company.

The DIRECT ENGINEERING℠ Vision: To provide an environment that allows an engineer to simultaneously consider both product and manufacturing requirements throughout the design/development/manufacturing cycle, resulting in a Total Product Definition built upon the collective intellect of the organization.

Key Principles (selected): An integral, dynamic knowledge management process; Seamless knowledge delivery and application; Rapid Development of a Total Product Definition; Variant Design

The "tool" that enables the DE℠ process of defining, capturing, and encoding the engineering process is a high-level software application. The DE℠ organization considers the choice of software important but has not prescribed a particular tool; they feel that developing a consistent, disciplined engineering process that connects and integrates sources of knowledge is the key to the DE℠ process. Most commonly, DE℠ applications have been developed on Unix workstations using IDL/ICAD, a LISP-based expert-system development package. This is not the only solution- the work in this thesis created a DE℠ application in Microsoft Excel.

The DIRECT ENGINEERING℠ organization is comprised of a core "DE℠ Team" and a number of application groups. The managers and technical specialists on the DE℠ Team are developing the vision and planning for its growth and dissemination across Ford. A major team goal is to generate a product development "mindset shift" across Ford which will overcome resistance to change and initiate a transition to the DE℠ approach. Among the DE℠ Team's activities are coordination of technological advances, detailing a formal Business Case, identifying standard processes, and targeting appropriate projects for applying the DE℠ process. The application groups are distributed across Ford and Visteon, each creating a unique DE℠ application for a specific vehicle component or assembly. Application groups are comprised of a supervisor or manager and several engineers, designers, and software developers.

Additional DE℠ process details will be presented as needed throughout the thesis.

2. PRODUCT DEVELOPMENT OF ASSEMBLIES

Automobiles are assemblies of many discrete components, and a large part of the vehicle engineering challenge is the integration of thousands of parts into a unified whole. Managing assembly design is a core part of automobile design. It is challenging in part because it defies the skill of any one person: "the intuitive resolution of contemporary design problems simply lies beyond a single individual's integrative grasp" [Alexander (1964)]. A typical car contains more than 10,000 parts, and organizing their collective design is perhaps the greatest challenge in manufacturing a motor vehicle [Womack, Jones and Roos (1990)].

The problems associated with assembly design have been the subject of much research and writing. Numerous methods for managing the successful integration of many parts have been presented in the literature. This thesis draws on these writings, but focuses mainly on the details: specifics of what to do within a larger assembly strategy. This section "sets the stage" by presenting the assembly context in which this research was done.

Section 2.1 introduces and defines the concepts of assemblies, components, and architecture, and introduces the problem of architectural evolution. Section 2.2 discusses Systems Engineering, a crucial vehicle design skill. Topics introduced include decomposition and reintegration (and limitations of this methodology), the FPDS V, and two key skills: specification writing and rapid iteration. Section 2.3 discusses Assembly Knowledge, which captures the relationships between parts and the assembly's system model.

2.1 ASSEMBLIES, COMPONENTS AND ARCHITECTURES

Vehicles are made of groups of interrelated parts- called assemblies - that together perform the desired functions. An assembly is not merely a collection of parts. Together, the connected parts in an assembly do things that the parts alone cannot. However, these parts interact in complex, not always anticipated ways, an important point that will be discussed below.

Components are separable physical parts or sub-assemblies [Ulrich (1995)]. Henderson and Clark (1990) define a component as a physically distinct portion of the product that embodies a core design concept and performs a well defined function. This definition confirms that

components need not be single physical entities and can be assemblies in their own right.

These concepts are best illustrated with an example. The approach used in the discussion below draws on Alexander (1964) who uses the illustration replicated in Figure 2-1 to discuss complex system design. The lines represent interactions between components, represented by the heavy dots. Sub-systems are represented by the large circles. Just how the sub-systems are identified is the subject of much research and is discussed in Section 2.2 below.

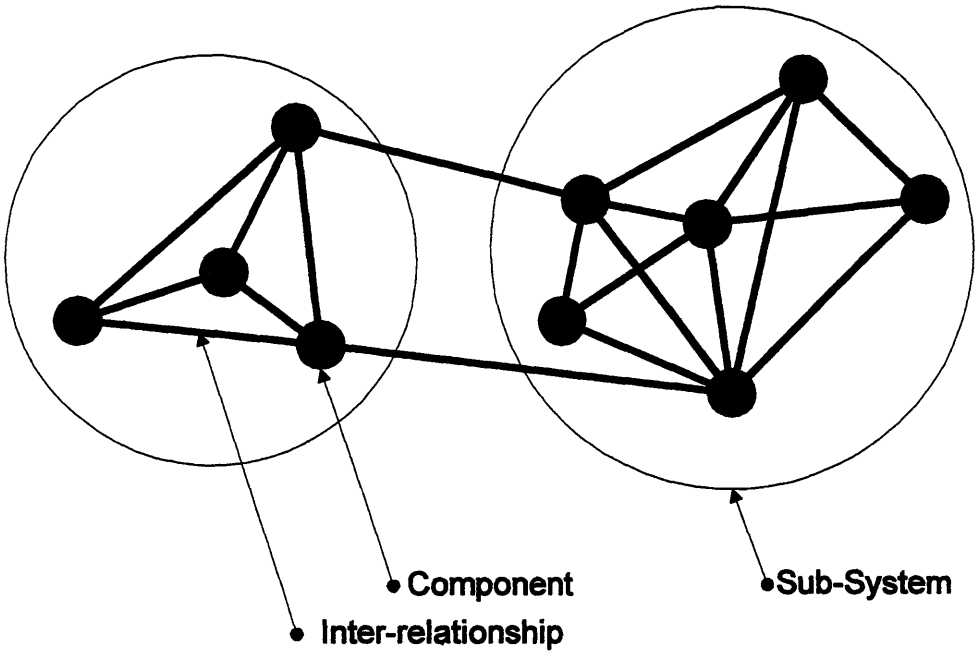

Figure 2-1 A System Illustrating Sub-systems [ref. Alexander (1964)]

This simple picture does not tell the full story, of course. Particularly absent is a representation of the functions of the assembly. Some if not all of the properties of an assembly are an outcome of the interaction of two or more components. These functions are assumed to reside within each sub-system even if not explicitly drawn.

Considering adapting Figure 2-1 for the Throttle Body. At what level of abstraction should the system be drawn? For illustration purposes, I will be begin by identifying the Throttle Body as a sub-system and presenting the surrounding sub-systems which interact with it. This is shown in Figure 2-2. The sub-systems conform to the circles in Figure 2-1; the components within them which will be addressed in later figures.

For Figure 2-2 to be strictly correct, some of the interconnecting links between sub-systems should pass through the system boundaries to show that most interactions occur between components themselves. Also, this figure is illustrative and does not present an exhaustive picture of all the sub-systems and their interactions.

Each sub-system represents a collection of components. Drawing these actual components for the Throttle Body results in Figure 2-3. This is a "close-up view" of the Throttle Body, and illustrates that surrounding sub-systems interact with components within the Throttle Body. Inside every other sub-system are similar "real components." If all of these components were also shown, the diagram would become quite complicated.

But this level of detail does not complete the picture. There is much more detail within the Throttle Body itself. Consider the Spring and all of its features. Figure 2-4 shows the Throttle Body sub-system with the spring details. Note that the spring interacts directly with components within the Throttle Body, but also with sub-systems that are outside it.

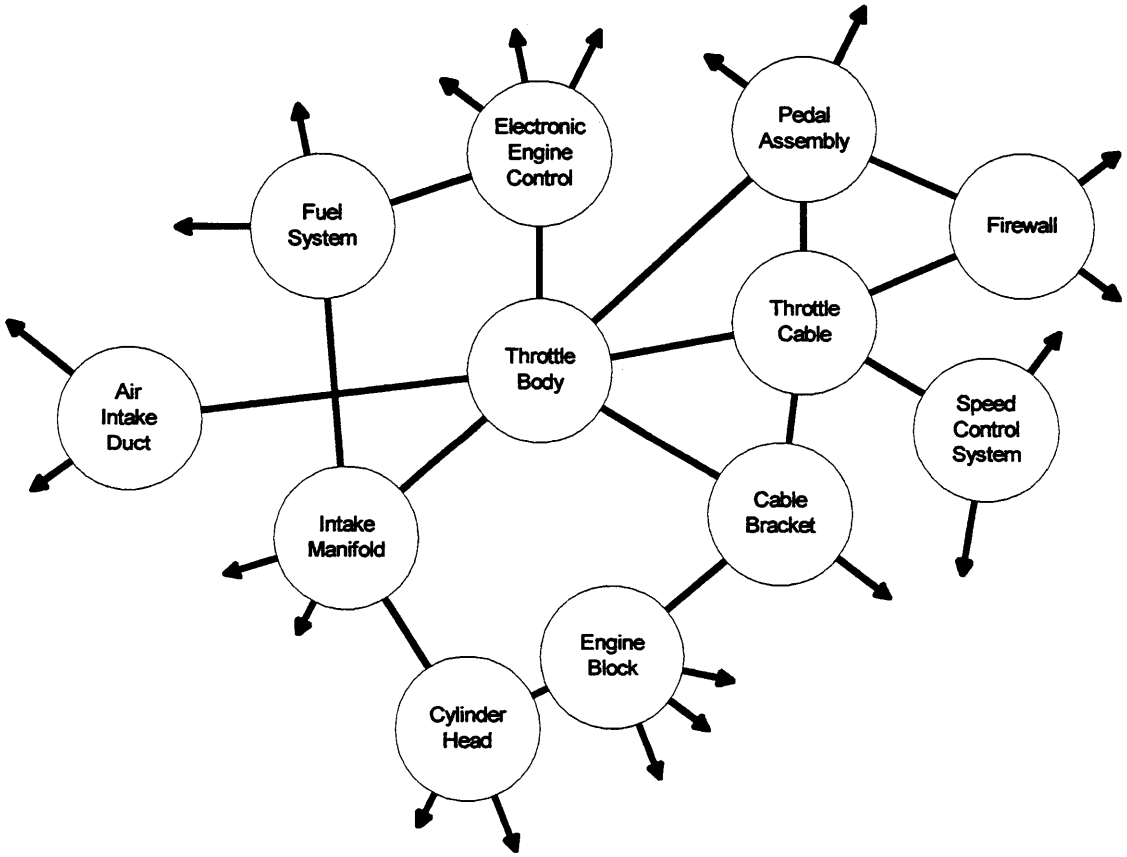

Figure 2-2 Throttle Body and surrounding sub-systems

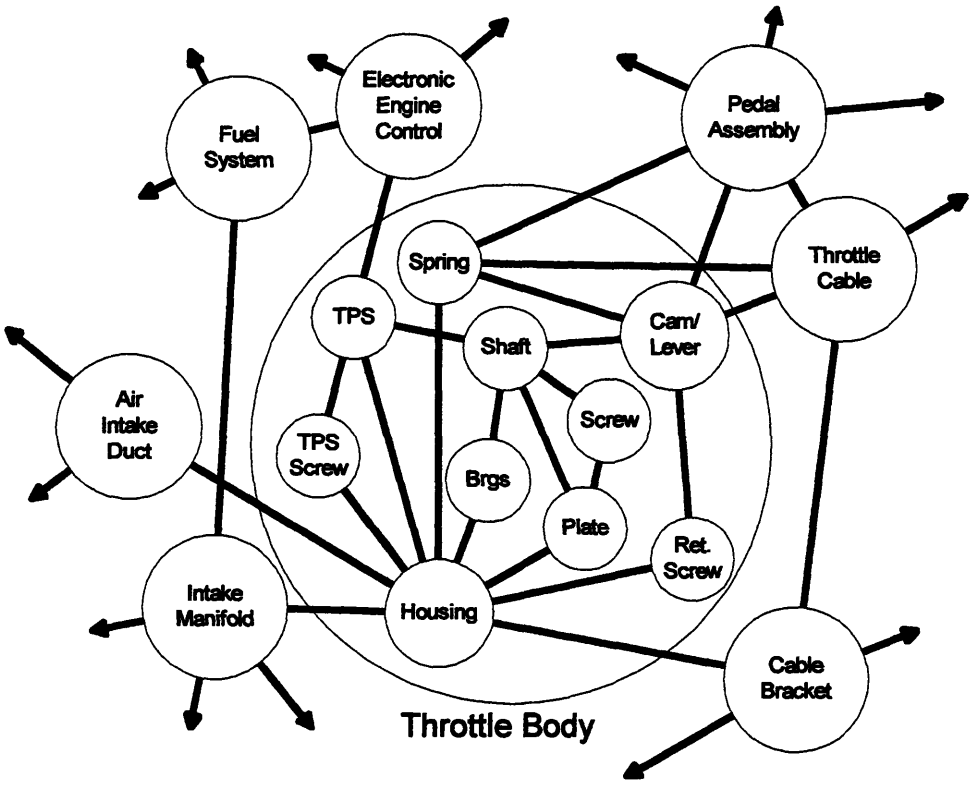

Figure 2-3 Close-up View of Throttle Body Components and surrounding sub-systems

The same level of detail shown in Figure 2-4 for the Spring can also be identified for the TPS (Throttle Position Sensor), for the Housing, or for the Lever/Cam. Now consider expanding Figure 2-2 for all such details for all components for all the sub-systems shown. This represents a problem of enormous size. Yet solving this type of problem occurs each time a car is designed; clearly vehicle engineers figured out many years ago how to manage this size of problem.

How? By using the process above, breaking a large, unwieldy vehicle design challenge into simpler, manageable problems. This is a key part of Systems Engineering and is discussed in Section 2.2. The above discussion and figures also help to illustrate Product Architecture and Architecture Evolution, important product development issues. These are discussed below.

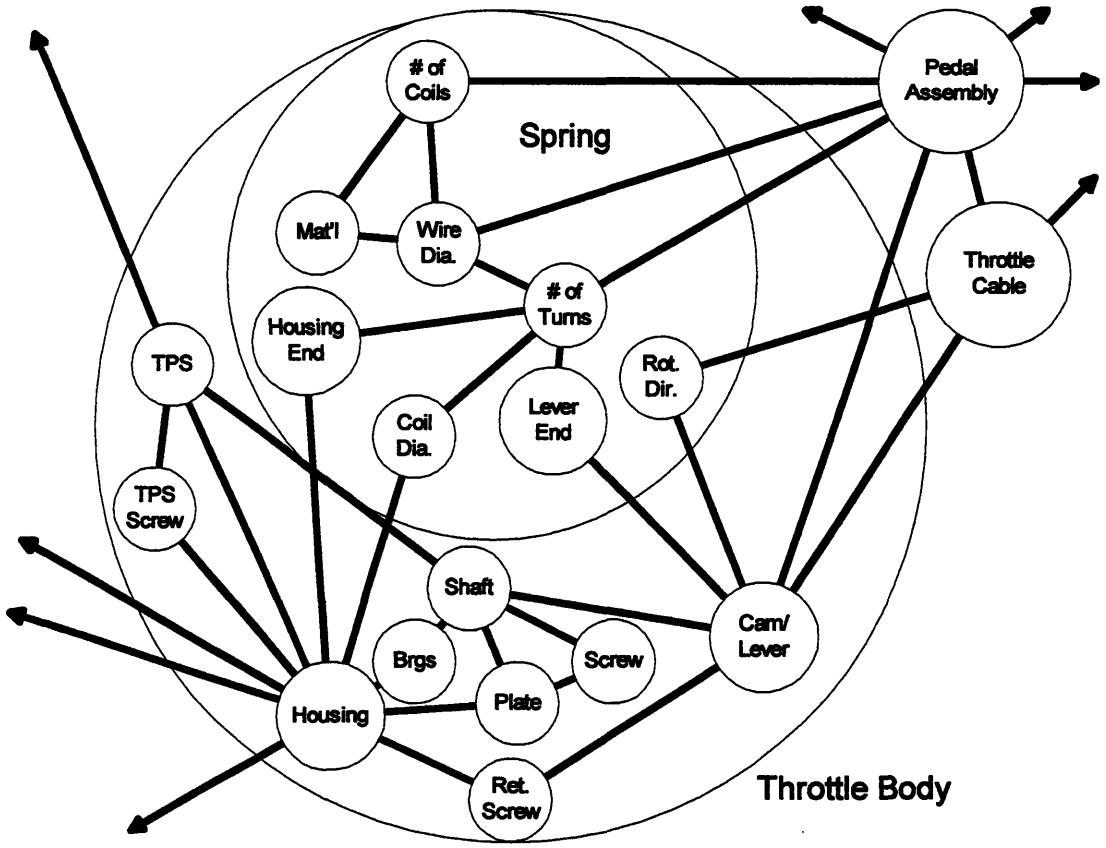

Figure 2-4 Throttle Body Sub-system with Spring details shown

2.1.1 Product Architectures

Ulrich and Eppinger (1995) define architecture as the scheme by which the function of the product is allocated to physical components, and the scheme by which the components interact. Architecture has also been defined as the laying out of how the various components will work together [Henderson and Clark (1990)]. The network of inter-relationships shown in Figures 2-1 through 2-4 above demonstrate a system architecture- one that is known and understood since the system exists. Issues of architecture design are only touched on here. Ulrich (1995) presents good arguments for various architecture choices, cast along an integral versus modular distinction. Suh (1990) presents a process for choosing architectures based on functional requirement independence. There are many worthwhile references for this topic;

Pimmler and Eppinger (1994) offer a useful summary.

Figures 2-1 through 2-4 illustrate an important architecture question: the assignment of components to a particular sub-system. How should one decide what component belongs where? The spring, for instance, is linked to components both within the Throttle Body and outside it (see Figure 2-3). Why is it located within the Throttle Body sub-system? An approach to answering these questions is Systems Engineering, discussed below.

2.1.2 Architecture Evolution

Assembly architectures are an important topic in product design, as the adoption, evolution and obsolescence of architectures are of central importance to design tasks and design organizations. The architecture illustrated in the figures above is applicable to a current generation of vehicles. Not many years ago, however, carburetors (which combined air and fuel metering) represented the dominant technology, and electronic engine control did not exist. Possible future developments include direct-drive throttle technology. Each of these significantly alters the sub-system map shown in Figure 2-2, and all maps "below" it (Figures 2-3 & 2-4).

Common themes of evolution seen are the separation, elimination or merging of both functions and components. These changes represent fundamental architectural changes, demanding that network of interactions be redrawn. A goal of product development management is that knowledge is retained and reused as these evolutions occur. And the method of saving knowledge must make change simple to implement, lest the organization become resistant to product evolution. These themes are picked up in Section 5. Examples of architectural evolution are:

- Separation of Functions: The internal-combustion engine carburetor has evolved into a butterfly valve and fuel injectors. Functions that were once in tight proximity are now physically dispersed, and the components can be developed by independent suppliers.
- Elimination of Components: The carburetor had jets, a float bowl and so forth, all of which are no longer necessary. Expertise in the design and production of these

products is no longer relevant; this knowledge became obsolete.

- Merging of Components: The throttle-body and intake manifold are expected to be integrated in the future. This change won't alter key sub-components such as the butterfly plate and cam-lever. Engineers will re-use know-how by integrating existing parts into the new architecture.

2.2 SYSTEMS ENGINEERING

Like most automotive assemblies and sub-assemblies, the figures in Section 2.1 illustrate a small part of a large and complex system with numerous interrelated parts. Overall performance depends on the collective performance of all these parts: vehicle functional requirements are established for the whole, not the pieces. Systems Engineering is the term given to a product realization process that addresses this challenge. It is used extensively in this thesis. The following description is taken from Fine and Whitney (1996).

The Systems Engineering process views the product as a series of levels; each lower level is defined in more detail than the level above, and contains subsidiary subsystems or components. The requirements defined for the lower levels support the levels above in precisely defined ways. Within a given level, requirements flow from the level above, are broken into supporting elements at this level, and are then expressed in terms of requirements for the level below. The "requirements" for an element means providing a function, or physical support, or power, etc.

Performing the systems engineering process means repeatedly determining the boundaries between elements in the level below. A basic principle of systems engineering is that the system at each level should be broken down into elements that have "clear and terse interfaces" with each other and the levels above. Interfaces are where elements and subsystems connect and across which the requirements are delivered. As much as possible, complex interactions are kept within subsystem (element) boundaries. Doing so simplifies subsystem requirements and minimizes the amount of interaction needed between subsystem developers during design. As Figure 2-1 suggests, components which share complex or strong interactions belong in one sub-system, and there should be few interactions between sub-

systems. Determining the relative strength of the interactions is an important question.

Often simple heuristics can be used to decide which sub-systems components belong to. Pimmler and Eppinger (1994) note that some types of interactions are more important than others, and that spatial adjacency requirements generally have a high priority in product architectural clustering. For this reason, engineering organizations have typically chosen to set architectural sub-system boundaries at clearly defined physical joints, instead of clumping all the components that contribute to a specific function into a sub-system.

A throttle body example is illustrative. The Throttle Return Spring's interaction with the Pedal Assembly can be characterized by a few numerical values (installed load and spring rate) which are easily communicated. It's harder to describe the spatial interaction of the Spring with the Throttle Body housing. Managing Spring-to-Throttle Body integration is more complex than managing Spring-to-Pedal Assembly integration. This heuristic approach satisfies the "judgment test": it is somewhat self-evident that the Spring belongs as a component of the Throttle Body.

In systems engineering, the process of setting the boundaries between elements at each successively more detailed product level is called decomposition. The decomposed, manageable pieces are developed individually and then reintegrated into the whole. Basic tasks required for systems engineering are determining good decompositions and then writing clear specifications (requirements) for each element or subsystem.

Figures 2-2 through 2-4 illustrate an example of decomposition, where the elements are assemblies or parts and the lines connecting them represent their interactions. The recursive nature of systems engineering is evident from these figures; each level of decomposition requires the same skills and presents the same challenges as the level above or the level below.

At each level of the systems engineering process, specification skills are the ability to determine the needs of the level above (essentially customers), break them down into supporting capabilities (decomposition) and then describe (specify) these capabilities to people or companies (essentially suppliers) who will have to figure out how to develop and deliver them [Fine and Whitney (1996)]. Defining a component's specification demands full knowledge of its interface with the system around it. On a purely technical level, it doesn't matter if

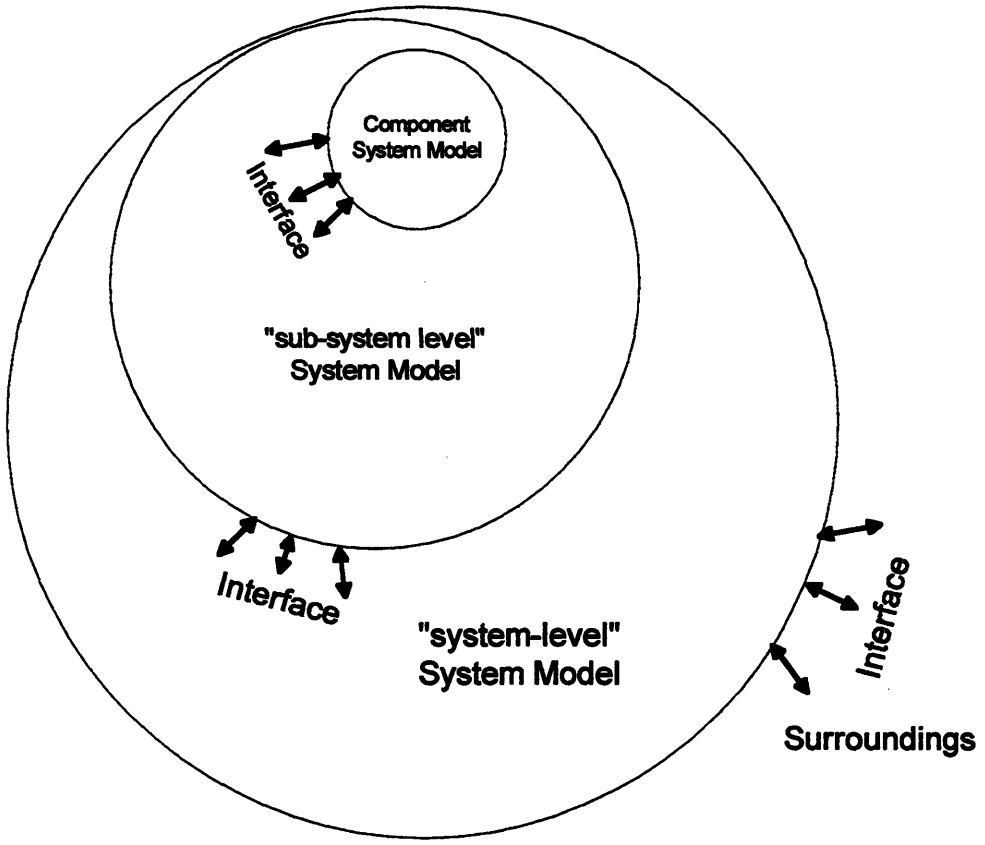

Figure 2-5 The Systems Model and Interface must be identified at each decomposition level

"suppliers" are members of the "customer" company or not: the engineering challenges are the same. As discussed in later sections, organizational issues do have important effects.

A look at Figures 2-3 & 2-5 illustrates an important concept that is central to the work done in this thesis. Consider the Throttle Body in Figure 2-3, or the "sub-system" in Figure 2-5. Applying the Systems Engineering process demands identification of three kinds of knowledge:

(1) The interface of the system with higher levels of the decomposition. In other words, the requirements imposed by the system's surroundings- how the Throttle Body interacts with the world around it.

(2) The system model at this decomposition level- what the Throttle Body elements are, and how they interact with one another. This will be called Assembly Knowledge here.

(3) The interface of the system with its elements, which are designed by "suppliers." Higher level requirements together with the system model determine element requirements or specifications.

Item (1) is the responsibility of the level above that being studied. Item (2), Assembly Knowledge- the system model- is discussed in Section 2.3. Item (3), the interface problem, is addressed in Section 3 along with other matters for supplied components.

Systems engineering problems are usually solved in an iterative process because they are too complex to solve in a single, deterministic step. Cognizant engineers at a given level perform repeated technical communication with suppliers, specifying requirements and collecting "latest revision" design information at each iteration. The goal is an optimum or acceptable design that satisfies the requirements and constraints of both the level above (the customer) and the numerous suppliers (the levels below, whether internal or not). The speed and accuracy of each iterative step have a significant impact on development time and the quality of the outcome.

The remainder of Section 2.2 discusses decomposition and re-integration in some depth, and identifies some problems inherent to this approach.

2.2.1 Intuitive Decomposition

Observations at Ford and Visteon suggest that cars have traditionally been decomposed into assemblies, subassemblies and components, manageable pieces which can be designed individually. The decomposition process occurs naturally to engineers and designers, who recognize their own limitations at managing simultaneous complex problems. The breakdown of automotive programs into engine systems, interior systems, and so forth recognizes similar coordination limitations at the engineering department level. Thus vehicle design organizations demonstrate an application of decomposition, a process which intuitively makes sense and enables rational management of the complexity inherent in an automobile.

There are limitations to this approach to decomposing vehicle systems. The decomposition may be along organizational or political boundaries that are not technically optimal. Without a formal process, clear requirements identification and specification probably does not occur.

Fortunately, the Ford Motor Company has adopted an explicit recognition of the decomposition /reintegration process: the FPDS V discussed below.

2.2.2 Ford's Framework for Decomposition: The FPDS Vee

Systems engineering is an integral part of the Ford Product Development System (FPDS) currently being rolled out across Ford. A primary tool for implementing systems engineering processes is the FPDS "V" which is shown in Figure 2-6. The figure's horizontal axis is time; the vertical axis represents levels of decomposition.

At the top of the V are broad vehicle requirements, which apply to major vehicle systems. Going down the V as a vehicle program proceeds, these requirements are decomposed into more and more specific sub-system, assembly, and component requirements. This is called "cascading" of design targets. Along the bottom of the V, design targets are converted into actual component designs, which are then integrated together into complete vehicle systems as the program proceeds up the right-hand side of the V. Thus the FPDS V is a visual representation of the decomposition and re-integration process.

Surprisingly, the need for iteration is not explicitly addressed with the V; the arrows shown on

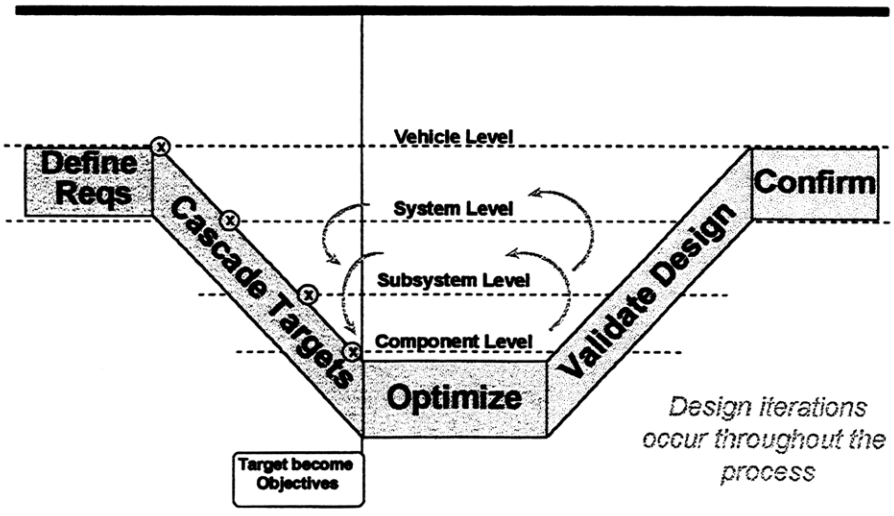

Figure 2-6 Ford Product Development System V. Used with permission.

the figure are my addition and are discussed below. However, iteration is addressed in the FPDS course taught across Ford. The FPDS V is a useful model for describing the complex vehicle development process.

2.2.3 Limitations of Decomposition/Reintegration

The systems engineering decomposition process is a valuable tool, but it captures complexity imperfectly. Problems within actual vehicle decompositions include components that influence multiple assemblies, performance parameters that aren't connected with a particular component, and designs which are not deterministic. Each of these are discussed below. The inability to capture complexity leads to inevitable conflicts between components and assemblies during the development process, which leads to repeated design iterations. While not guaranteed, the number of conflicts should decrease with each iteration of decomposition and re-integration.

Components influence multiple systems

It is the nature of complex integral assemblies such as automobiles that components will reach out beyond their place in the V by interacting with other parts, in "distant" assemblies. If these effects are strong, then determining system performance demands knowledge of components that aren't within the sub-system under consideration. For instance, the Throttle Return Spring interacts with other parts of the car: the Accelerator Pedal (pedal feel), the Speed Control System (operating loads), and the Accelerator Cable (location and direction).

This is related to the problem of choosing the system's decomposition. For instance, consider Figure 2-7, which is Figure 2-2 re-arranged with several candidate systems to which the Throttle Body could belong. Which one is the right one? Briefly, its hard to say. This is an area of study that has attracted much research attention. A noteworthy methodology for solving this problem is the Design Structure Matrix, or DSM [Steward (1981); McCord and Eppinger (1993)]. The DSM is developed by interviewing the development organization staff about interaction between parts. After the Design Structure Matrix is completed and optimized, it reveals the important component interactions or "clumps" and states that product development will proceed more effectively if these clumps are explicitly managed

together.

A simple heuristic discussed earlier is that components belong to the system with which they have the "strongest physical interaction." This appears to describe the chosen Throttle Body decomposition, which is considered part of the Air Intake System. The Throttle Body shares two mounting surfaces with this system: a flange on the Intake Manifold and a lip on the Intake Duct.

Performance parameters aren't connected with single components

Often overall system performance is an outcome of the interactions of many components and sub-systems and can't be cleanly assigned to a single component. This commonly occurs in highly integral architectures, where there are problems applying the systems engineering approach [Fine and Whitney (1996)]. Where many components interact in a small space and need to meet many integrative functional requirements, the system is not readily decomposed.

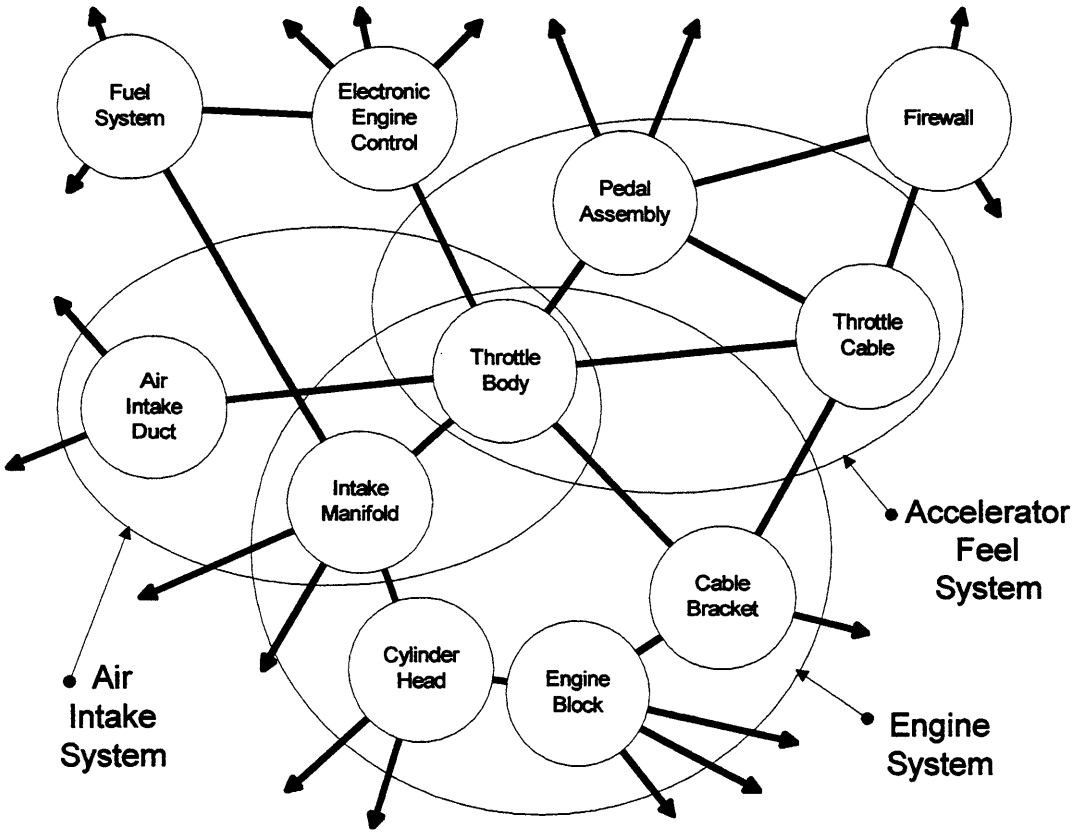

Figure 2-7 The Throttle Body is a candidate member of several systems

A good example of this is a car's B-pillar. Here, structure (roof, floor, door system), appearance (interior and exterior styling), and numerous functions (stiffness, NVH, safety, weight, wind noise) all intersect in a tight space. A systems engineering decomposition of this area is nearly impossible, as is recognized by many engineers at Ford.

Designs are not deterministic

As has been mentioned earlier, systems engineering decomposition may not produce a deterministic design. Experience shows that simply cascading targets doesn't insure that they can be met: going down the V once and back up again will not result in a good or even adequate design. This is due in part to the types of interactions discussed above, and explains why systems engineering requires iteration. Repeatedly, at intermediate stages in the V, individual components are assembled and the integrated whole is checked for performance. Problems as apparently simple as mechanical interference are identified, requirements are tweaked, and the design process is repeated. The need for such iterations suggest that the systems engineering process cannot be automated.

These observations suggest various strategies that may expedite the vehicle development process. One such strategy is "Set-Based Design" [Ward et al (1995)]; another is "Synch-and-Stabilize" [Cusumano and Selby (1996)]. Both are discussed as a directions of future research in Section 6.

Conflicts: the norm rather than the exception

The above paragraphs suggest that complex assembly design will lead to frequent conflicts which must be managed. For instance, a straightforward decomposition falls apart upon reintegration due to what I call the "reach across the V" problem. Yet traditional development processes seem to treat these as unlikely exceptions, unplanned problems that demand special attention and shift the overall process off the smooth development path. Since experience shows such conflicts will always occur, perhaps they should be treated as part of the normal process. For instance, the DE[SM] process should not be planned to provide one-time optimization, but instead to identify and manage the conflicts that will arise as the 10,000 parts in a car are collectively developed. The development process described in Section 4

suggests how conflicts between component requirements and component design can be managed for the Throttle Return Spring. Such a perspective is needed for the entire vehicle.

2.3 ASSEMBLY KNOWLEDGE

The term Assembly Knowledge is the name given here to the system model which captures how elements (components) interact with one another within a given decomposition level. For the Throttle Body, this is the network of interactions shown in Figure 2-3. Assembly knowledge also refers to a component's system model, because components can be viewed as assemblies of features. For instance, the spring in Figure 2-4 is an "assembly" of simpler elements. Managing assembly knowledge is central to the DE[SM] vision for expediting the product development process.

Each DIRECT ENGINEERING[SM] application, in essence a semi-automatic software model of an assembly, requires that assembly knowledge be collected and encoded. Depending on the decomposition level, the assembly knowledge might consist solely of integration of components designed at lower levels, or might include components designed at this level.

However, assembly knowledge is difficult to describe, difficult to capture and generally not explicitly defined. While certainly known to product-development personnel, observations of real organizations show that such knowledge is informally maintained. And due to assembly evolution, there is a real risk that assembly knowledge, if captured, will soon be obsolete.

This section discusses assembly knowledge: what it is, where it is found. It then identifies the need for managing assembly knowledge within the DE[SM] process: assembly knowledge is important, and the DE[SM] organization needs to figure out how to manage it.

2.3.1 Defining Assembly Knowledge

Assembly knowledge is the set of everything needed to be known about an assembly in order to design it at the assembly level. This includes how functional requirements, or specifications, for the assembly are related to individual part parameters. It includes the interactions between parts- the architecture- and the parametric equations that relate component dimensions to one another. It includes information related to each part which is not about the part, but about its

interaction with its surroundings. In short, assembly knowledge is the collection of information and relationships used when the assembly is developed. In the DIRECT ENGINEERING[SM] vision, assembly knowledge also includes the interactions of parts with their manufacturing processes, such as existing assembly lines and tooling.

2.3.2 Organizational location of assembly knowledge

In the organization studied at Visteon, assembly knowledge is managed by the people who do the assembling: designers (CAD drafting personnel). A new assembly is typically created by modifying existing designs and components to work together to new requirements. Parts are modified as needed to make them fit. An engineer will intervene to confirm tolerance stack-ups and resolve technical problems, but the assembly work is done by the designers.

When the engineers developing the Throttle Body DE[SM] Application realized they needed assembly knowledge, they went looking and found very little recorded information. They had to create their own, and in so doing developed a tool which was called the Associativity Map. The story of this map is a telling example of how assembly knowledge is maintained.

An early version of the Associativity Map is reproduced (albeit illegibly) in Figure 2-8. The map was normally printed on E-size (44" x 34") paper; the figure illustrates its general form.

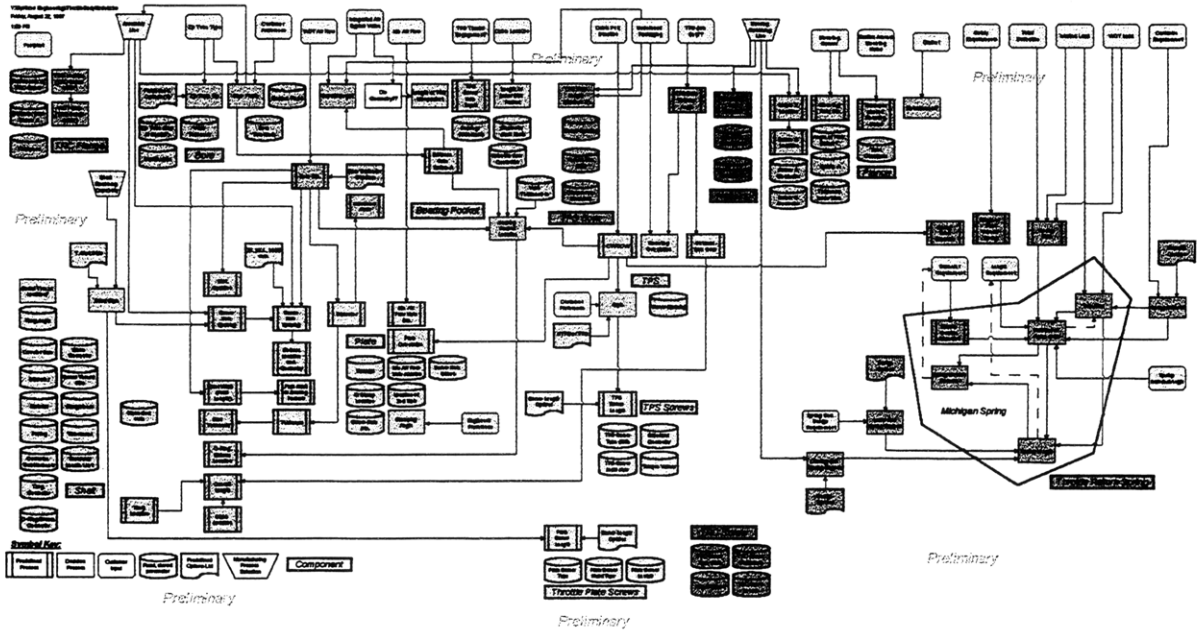

Figure 2-8 Preliminary Associativity Map

Each block represents a Throttle Body component feature which can be either fixed, dependent on other features, or user-edited. The blocks are grouped and color-coded to identify particular components. Relationships between the features are denoted by arrows running from one block to another.

The Associativity Map was first developed using Post-it Notes and a large roll of paper. Later, it was moved to a computer file. During the internship, this writer maintained the map, updating and extending it as the DE[SM] application grew to contain more components. The Associativity Map was difficult to get or keep up-to-date, was probably never completely correct, and has surely languished since. The engineers contributing to the Associativity map, which was the only record of the complex inter-part relationships outside of the Throttle Body DE[SM] Application code, saw it as an "extra" task and only infrequently used it to develop or understand the Throttle Body system model.

This anecdote supports the assertion that assembly knowledge is difficult to identify, collect and record, and because the benefits of doing so aren't immediately apparent, these tasks are neglected. As a result, assembly "knowledge management" occurs informally in the organization's unwritten memory, a topic discussed in Section 5.

2.3.3 Managing assembly knowledge for the long term

If an assembly's architecture- its part-to-part relationships- are recorded as assembly knowledge in a manner that supports ready modification, then architecture evolution is easily accommodated. If the components of an assembly are viewed as "building blocks" with well defined interfaces with their surroundings, then assembly knowledge is the map that reports these interactions: which components interface with one another, and how they interact. Because the assembly architecture will evolve, this map will have to change- an eventuality which must be considered at the outset. Good techniques for managing assembly knowledge for the long term weren't developed during the internship and aren't proposed here; this is an area of potential future research.

2.3.4 Assembly knowledge and the DIRECT ENGINEERING℠ application

The DIRECT ENGINEERING℠ semi-automatic development environment- e.g., the DE℠ application- embodies the product' system model and therefore all captured assembly knowledge. The DE℠ application also manages supplied component interfaces as described in Section 3. Despite its importance, the DE℠ process does not address assembly knowledge per se, and does not have a defined methodology for managing it or planning for its change.

As discussed above, recording assembly knowledge is a difficult but important step in developing a systems engineering model of a product, and this knowledge must be recorded in such a way as to make it readily modifiable. Therefore, as a DE℠ team plans a new application, their first step should be to identify the product architecture and corresponding system model, and then the components and their interfaces. After all this in-depth "homework" is complete, the coding can begin. A systems engineering perspective is useful here: each component or sub-system can be treated as a "subroutine" while a top-level set of code embodies the assembly knowledge and the system model. Architectural change then entails altering the top-level code and a small amount of lower-level component information.

This is not far from the process the writer observed with the Throttle Body team. New DE℠ applications are commonly written in the arcane LISP language of ICAD programs; some sample lines are shown in Figure 2-9. LISP is an object-oriented programming language, and the Throttle Body DE℠ Application organized many assembly relationships into top-level code with well defined parent-child relationships to the component objects. However, component-to-component interactions were often written into the low-level component code.

As a result, much of the assembly knowledge was buried rather deeply into 30,000 or more lines of LISP programming, which is not very easy to decipher. The code certainly maintains all the proper relationships, but is difficult to interpret and can rapidly grow beyond a new-comer's capacity to absorb (or for that matter, the original writer after a few months hiatus). When one of the developers changed jobs after some 10 months of active involvement, his undocumented work was left to the others to sort out, a difficult task.

Thus, my observation was that assembly knowledge in the Throttle Body DE℠ Application was embodied in the code but not well documented. Recording assembly knowledge was

```
(def-part shaft (subtracted-solid)
 :inputs
 ( :body-bore-diameter
  :plate-diameter
  :plate-thickness)
 :modifiable-attributes
 ( :shaft-style                    :slotted
...
 :attributes
 ( :overall-shaft-length (+    (:the :shaft-center-to-stake-end-length)
                               (:the :shaft-center-to-tang-length))
  :shaft-center (the  :slot  :center)
...
 :pseudo-parts
 ((shaft
  :type  cylinder-solid
  :radius (half (the :shaft-diameter))
  :length  (+    (:the :shaft-center-to-stake-end-length)
                 (:the :shaft-center-to-tang-length))
  :orientation (:numeric (alignment
                          :top (the :vertical-vector)
                          :rear (the :longitudinal-vector)))
  :position-about
  (:local-point (:face-center :rear)
   :model-point (translate (the :center)
                 :longitudinal (the :shaft-center-to-tang-length))))
...
```

Figure 2-9 Sample ICAD LISP Code

given a low priority because the immediate value of the Throttle Body DE[SM] Application could be provided without it. And time pressures for developing demonstrable code didn't favor careful documentation. Not recording assembly knowledge will make incorporating product changes difficult- but this problem will appear only slowly. Assembly knowledge management is a complex, long term matter- which the DE[SM] organization should learn to address.

2.4 ASSEMBLIES AND SYSTEMS ENGINEERING: CONCLUSIONS

This section reviewed product assemblies and architectures using the Throttle Body and Throttle Return Spring as examples. Product architecture was defined as the scheme by which product functions are allocated to components and how these components interact. Product architectures continually evolve through the separation, elimination, or merging of components and functions.

The systems engineering process was then introduced as an approach to managing the large, complex, and interrelated problems associated with designing automobiles. An important

lesson from systems engineering is that good product design begins with good specification and good decomposition. Fine and Whitney (1996) assert that this is a top level skill of major importance. At each level of decomposition, these steps identify individual components and how they relate to their surroundings. This information evolves as the system architecture evolves.

The systems engineering discussion identified limitations to decomposition which lead to the need for iteration in the design process. Iterations are necessary because decompositions don't result in deterministic solutions. The faster iterations are completed, the quicker a good solution can be reached.

Finally, assembly knowledge was argued to play a central role in product development yet be difficult to identify or manage. Assembly knowledge is at the core of DE[SM] applications, but has received little attention from application developers and was observed to be of secondary importance to the Throttle Body DE[SM] team.

This section "set the stage" for the remainder of the thesis by presenting the product context in which the research was done, and suggests some important observations for the DE[SM] effort. Systems engineering should be a focus of the DE[SM] process, since DE[SM] applications are tools that design systems. And inevitable product evolution demands that systems engineering information and assembly knowledge be recorded and maintained in a format amenable to extension and modification. Anticipating and accommodating product evolution is of central importance to the DE[SM] process as a product development methodology.

3. PRODUCT DEVELOPMENT WITH SUPPLIER-DESIGNED COMPONENTS

The previous section introduced the systems engineering process as a tool for managing the challenges inherent to the development of large, complex assemblies. At the core of systems engineering is the decomposition of assemblies into simpler elements, or subsystems. A subsystem can be decomposed and worked on separately if its performance requirements can be stated clearly and independently of those for other subsystems [Fine and Whitney (1996)]. Once the subsystem is completed, it can then be re-integrated with the rest of the product.

A primary motivation for systems engineering decomposition is that the subsystems can be developed by independent organizations. At automotive companies, "independent organizations" once meant component specialists within the parent company. In recent years, external suppliers have developed the capability to provide significant design content. In many respects the technical challenge is the same whether the component is outsourced or not: a different engineering group (the "supplier"), separated by distance if not organizational barriers from the "customer," develops part of the product. While the customer manages the assembly integration problem (Section 2.3 above), significant end-product design responsibility is in the hands of "supplier" engineers.

To create an effective process for product development in this environment, the nature of the relationship between the "customer" and the "supplier" along technical and organizational dimensions must be understood. Section 3.1 below presents a framework for classifying customer-supplier relationships along two axes: degree of product customization and design process interaction. A specific type of relationship, termed variant-iterative development, is identified as most relevant to the studied problem. It is applicable to a great many supplied components, whether developed internally or outsourced.

Since the thesis is focused on an outsourced component, the nature of the relationship between the customer and supplier (in this case, Visteon-Rawsonville and Michigan Spring, respectively) is important. Rather than focus on this single relationship, Section 3.2 discusses in broader terms the history and current norms of the automotive industry customer-supplier

relationship. The wide disparity in technology and skill levels observed have significant implications for any Systems Engineering process.

In the context of systems engineering, Fine and Whitney (1996) argue that good design equals good specification writing. But how is good specification writing done? And what if the specifications must be written repeatedly as the design is iterated? This work separates supplied component specifications into two complementary pieces: (1) a shared component framework or architecture (which will be called a variant component attribute structure), and (2) the set of unique information defined within this framework which must be communicated at each design iteration. The component attribute structure and the communicated information is described in Section 3.3.

3.1 TYPES OF CUSTOMER/SUPPLIER DEVELOPMENT INTERACTION

When exploring methods for improving the supplied-component development process, it became clear that there are many different approaches to developing supplied components. Supplied components can be classified by the degree to which a new design is custom, as well as the type and frequency of communication between customer and supplier organizations.

3.1.1 Characterizing by degree of customization

Supplied components can be characterized by the degree of customization of the design. I propose three general ranges, although more properly this should be considered a smooth continuum.

At one end of the scale of customization are Catalog parts, components which are selected from a predetermined set of alternatives- e.g., a list of fastener styles, sizes, and lengths. Developing new designs with catalog parts is pretty straightforward: the design engineer simply needs the list of alternatives and decision criteria. A part closest to the requirements is then selected from the list.

At the other end of the spectrum are fully Custom parts. These are parts which have a "never been done before" nature- or are at least perceived as such by the design group. Truly custom parts share no common features with any existing components. More realistically, custom

parts do have some common features, but are mostly new and different. Or a custom part could share a large number of common features with pre-existing designs, yet have them arranged in such an unusual architecture that they look nothing like stuff that's been made before.

Between the Catalog and Custom extremes are Variant parts, which Sferro et al (1993) describe as sharing many of the characteristics of items which have been designed before while being variations of these characteristics that have not yet appeared. They argue that "90% of all products fall in this category." Thus variant parts are unique variations drawn from an existing set of design options. For instance, a component may have several flange shapes, each conforming to a certain type of mating part. And it may have a few discrete diameter choices, as well as upper and lower length bounds. These 3 variant choices constrain design freedom, yet can result in "infinitely" many combinations, such that no two parts are alike.

Variant parts can have numerous design options, or very few. They can have quite flexible or rather rigid architectures: a boss feature might be restricted to a single location, or be freely located on any of several surfaces. Variant parts require well-defined and shared understanding of the component architecture and component-to-surrounding interfaces. Variant parts greatly increase the likelihood of reusing existing process tooling and component knowledge while giving the designer broad latitude in configuration. The variant approach is readily applied to automotive components.

An example of applying a variant approach (although not by this name) to automotive component assemblies is identified at Nippondenso Co. Ltd. (now Denso) by Whitney (1993). There, a "combinatoric method" delivers high assembled-product variation by having several versions of each component available, with well defined interfaces between the components. A noted advantage to Nippondenso's product development strategy is the ability to easily and rapidly design and manufacture very different assemblies. Variant parts apply the same type of thinking at the component level, allowing high component variation to be easily and rapidly realized.

Variant parts also are potential candidates for re-use. Since they can be catalogued in a well-defined manner, it is straightforward to review a product database and identify existing parts

that are close to a new component request. It is unlikely that a perfect match can be found, due to the high number of possible product permutations supported in a variant framework.

Variant parts lend themselves well to parametric, features-based design processes, as supported by recent CAD packages. They are also ideal for the DIRECT ENGINEERING℠ semi-automatic development environment. Developing a DE℠ application is comparatively straightforward when the components or subsystems being designed conform to well understood component architectures and design options.

3.1.2 Characterizing by type of interaction

Supplied components can also be characterized by the type of interaction observed between the customer and supplier during design development. While development interaction is properly considered a continuous variable, I propose three general ranges of interaction: static, iterative, and dynamic.

In many instances, and particularly in the older pattern of supplier relationships, there is a one-time communication with the supplier. The customer might supply the supplier with a completed drawing, a marked-up drawing or a complete specification. No supplier feedback into the design or specification is solicited or expected. I term these "one-time-through" component development processes Static Interactions.

Often, a customer will be uncertain of a supplied component's final design or final design specification. Because customer engineers lack the ability to translate their specifications into a supplied components' shape or size- they rely on the supplier to do that- they need to review the supplied part's fit with surrounding systems after it is designed. They then confirm or adjust its specification. Changing the specification repeats- iterates- the process. In this Iterative Interaction components are developed through repeated communication of product requirements and product design. At each step in this development process, a complete design is exchanged. Iterative Interaction is an outcome of the limitations to decomposition discussed in Section 2.2.3. It was observed for the Throttle Return Spring, where Throttle Body engineers don't know if a spring with the needed performance will fit in the space left for it until completing at least one design iteration.

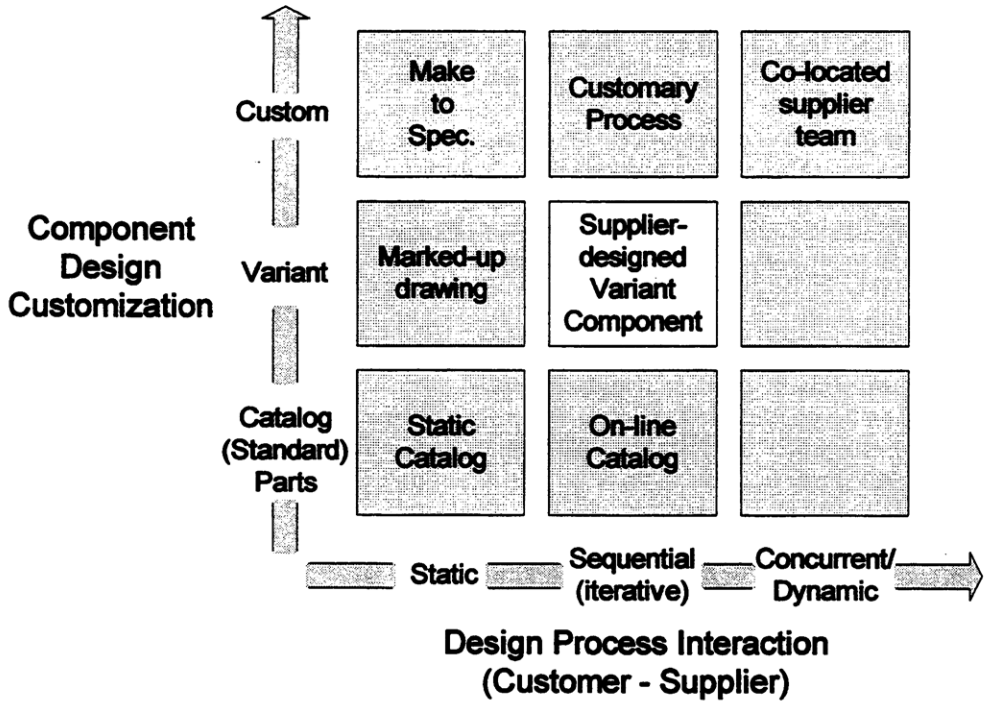

Figure 3-1 Types of Customer/Supplier Interaction

Finally, in some instances the supplier designs a new component in a Continuous Interaction process. Here, the customer and the supplier develop the requirements and configuration of the supplied part together. Frequent communications between the two parties convey partial design changes, say to only a small area on the part. These "mini iterations" occur so frequently that the product evolves in a more-or-less continual manner. Performing Continuous Interaction demands strong communication process, perhaps through collocated supplier/ customer development teams.

3.1.3 Focus of this work: Variant, Iterative Design

The two axes of characterization described above can be plotted against one another, which makes a three-by-three chart of potential types of customer/ supplier interaction (see Figure 3-1). Any supplied component can be characterized and then placed on the chart. In general terms, the development process is different in each cell.

The process developed in this thesis is applicable to Variant, Iterative types of parts. To

reiterate from above, these parts are unique variations drawn from an existing set of design options, where the design process is iterative in nature: both functional requirements and component design evolve over time.

It is important to note that a great many types of products, particularly those in rapidly-evolving industries, cannot claim to have a well-defined architecture or clearly defined component-to-surrounding interfaces. For these industries, the Variant- Iterative process described in this thesis is not particularly relevant. But many industries and products are mature, with significant product changes occurring every decade or longer. For these industries, the product development challenge is the rapid adaptation of a known component to a new application. This problem is addressed by the Variant-Iterative process defined in this thesis.

3.2 ORGANIZATIONAL AND BUSINESS RELATIONSHIPS

Supplied components are developed outside an organizational boundary, an important distinction which raises pricing and intellectual property concerns that are tightly intertwined with the history of the automotive industry. The "big three" talk about long term supplier partnerships, but organizations moving in this direction are still pioneers. Old behavior patterns and expectations remain. This section is intended to demonstrate that establishing the trust necessary for the process described in this thesis is challenging given today's typical automotive supplier relationship.

This section discusses the relationships inherent to an automotive supply chain from an historical and organization viewpoint. It begins with a brief general history and then discusses the relationships observed at Visteon Rawsonville and the importance of developing trust. A discussion of the heterogeneity observed in the supply chain product development environment concludes the section.

3.2.1 Automotive Supplier Relationships- a short history

The following short history draws from Womack, Jones and Roos (1990), as well as personal reading and experience. Early in the automotive industry's history, Henry Ford decided the

best approach was complete vertical integration: every single part of a car was designed and manufactured within the Ford Motor Company. In the 1920s, Alfred Sloan of General Motors modified this approach slightly, by making internal "suppliers" independent profit centers. Then in the 1950s, Ford began to bid fully designed parts out to independent suppliers, a practice that become commonplace. By and large throughout the automotive industry, internal engineers designed parts for each new car and provided suppliers with complete drawings. This was a world of "arm's-length, market-based, short-term interactions with independent businesses," and it lasted for decades. Parts were detail-designed internally by the vehicle manufacturer, and some 1,000 to 2,500 suppliers would be called in to review the drawings and supply bids. Contracts were for short periods- several years at the most.

In recent years suppliers have taken on greater and greater amounts of design responsibility. In this process, competing bids submitted in response to preliminary specifications are used to select suppliers and set pricing; prices are then incremented if the design changes. One common supplier practice is to start with a low-ball price and expect to raise it to profitable levels during redesign [Ward et al (1995)]. The usual contract duration is still only a few years, and the customer typically buys the suppliers' tooling partly to control the profitable aftermarket. However, this makes the threat of switching suppliers quite realistic.

This is often not a friendly business, and suppliers can be treated quite aggressively. Of course, suppliers are known to attempt to reap excess profits when possible [Walton (1997)]. As a result relationships between the "big three" and their supply base can be pretty strained, with no lack of mutual responsibility and plenty of mutual blame. Unfortunately for many suppliers, the potential for good relationships with the automotive companies is often spoiled by the behavior of less level-headed suppliers. Automotive management really does not trust the suppliers they rely upon. Suppliers expect periodic non-negotiable price cuts. The supplied-component design process occurs in a context where it is difficult to be open about many issues.

3.2.2 Organizational norms at Visteon-Rawsonville

While broad historical generalizations are helpful in setting the stage, every specific customer

and supplier has a unique relationship. My research was conducted at the Visteon Rawsonville plant, and with a spring supplier named Michigan Spring. Visteon is perhaps leading the way at Ford as the automotive giants move in the direction of long-term mutually beneficial relationships with their suppliers. Michigan Spring expressed a great deal of commitment to an open and trusting relationship with Visteon. While memories are durable and there is a long history to overcome, this relationship suggests that improvement can happen.

Despite this positive experience, evidence of earlier behavior patterns remain. For instance, Visteon personnel generally expect suppliers to bid below cost and make money back on design changes. And all parties expect Procurement to beat suppliers up on price. Some engineers express doubt as to whether Procurement is part of the same organization, or shares the same goals. These same engineers, however, can recount tales of why suppliers can't be trusted. One such story told of a supplier asking Ford for specification relief on a component, suggesting that it could then use a lower cost material. When the requested specification change came through, the supplier increased the price- knowing that the cognizant parties at Ford weren't likely to discuss the matter with each other. A Visteon engineer caught the supplier and was furious. Needless to say, the supplier's action hurt the future prospects of itself and other suppliers. To counter such behavior, Procurement maintains an approved suppliers list that engineers are supposed to use.

Visteon Procurement personnel, not unlike any at the "big three," expect suppliers to be compliant due to the large volume of business they are offering. For example, suppliers are asked to adopt Visteon's specific CAD system. To this end, Visteon may provide CAD systems to suppliers. Accepting this "gift" means that (1) the supplier will train and maintain the skills needed to support it, and (2) that only Visteon/Ford business will be conducted on this piece of hardware/software. But even small suppliers are likely sell to all of the "big three" (each making a similar demand) plus other non- automotive businesses. For many suppliers the cost of supporting a stand-alone CAD seat for Visteon alone will be untenable despite the size and importance of this customer.

3.2.3 The importance of trust

Designing a product with an supplier is not the same as doing it with a separate in-house organization. While internal and external organizational barriers both impede communication and interaction during design, new concerns arise when company boundaries are crossed. To perform the process presented in this thesis, the customer and supplier must create a shared technical component strategy. Because each must "reveal their hand" to the other as to potential design options and features, they have to be ready to work together. This demands an expectation that the customer-supplier relationship will be longer than just the next part. It demands trust- that the supplier's contributions won't be taken to another bidder, and that the customer won't capriciously change its system architecture, obsolescing the supplier's investment in the current approach.

In addition, exchanging product development information leads to intellectual property exposure for both parties. Suppliers are very concerned about protecting their knowledge even when rational judgment suggests that the customer has no use for it. Partly they are concerned that proprietary information could be accidentally (or intentionally) revealed to a competitor. Customers have the same concerns, since suppliers conduct business with the customer's competitors and information might similarly be revealed.

An anecdote helps illuminate these concerns. During my internship, Visteon was "spun-off" from Ford as a wholly owned automotive component supplier which would now compete in the marketplace for Ford business. At a meeting the same week as this announcement, Visteon engineers (who had been Ford engineers the previous week) began raising concerns about intellectual property: they feared Ford would use them to help design a new component and then bid the business out to competing suppliers. The irony was not lost on a gray-haired supplier engineer nearby who chuckled and muttered, "Welcome to the real world."

While the need for trust is universally recognized, establishing it despite the influence of historical automotive supplier relationships will be difficult. Yet trusting supplier relationships are crucial for meeting the stated goal of speeding the development process.

3.2.4 Heterogeneous environments

Across any automotive supply chain, the observed product development environments differ along many dimensions. There are broad disparities both vertically (Ford compared to suppliers) and horizontally (suppliers compared to one another). Observed dimensions of variation are:

- Software used. Components may be modelled in complex 3-D solid modelling packages (especially at Ford) or simpler 2D wireframe packages (smaller suppliers). Ford is a heavy user of CAE, some supplier use paper charts as design standards.
- Operating Systems: Most high-end CAD/CAE software runs in a hardware-specific version of Unix. Mid-to-low end software runs on PC/ Windows.
- Hardware used. Many suppliers use a PC environment; Ford uses workstations. This impedes either organization adapting the other's standards because there is a wide difference in software availability and cost.
- Range of expertise. Ford is more likely than suppliers to have personnel devoted to developing expertise in software customization, as well as complex programming. During this research, Ford was developing 30,000 line LISP-based ICAD programs, while the was writer coaching the supplier in writing Microsoft Excel macros.
- Differing rates of technology adoption, particularly with respect to Internet/Intranet technologies. Ford is an early and strong adapter of Intranet technology [Cronin (1998)] whereas Michigan Spring came "on-line" during the internship.
- Differing valuation of technology as a business tool. Suppliers see adopting technology as a means of keeping up with Ford's requirements; whereas engineers and managers at Ford (like the DE[SM] team) see technology as a competitive investment.

In summary, Ford/Visteon differs along many technical and organizational dimensions with most of its supply base. These differences somewhat impede the dissemination of new processes and technologies. For a new product development process to succeed, it must work across the broad range of skills, tools, and technologies that exists within the supply chain.

3.3 SUPPLIER-DESIGNED COMPONENT DEVELOPMENT FRAMEWORK

The following steps occur during supplied variant component development:

- The customer determines the supplied product's requirements (specification).
- The customer communicates the requirements to the supplier.
- The supplier designs a component that may or may not conform to the requirements.
- The supplier communicates the resulting component design back to the customer.
- The customer integrates and evaluates the component in its system, determining if the design is satisfactory. If not, the supplier will be asked to try again, possibly with revised requirements.

This process (specify-communicate-design-communicate-integrate-evaluate) repeats until the design is accepted as complete. This section describes the framework developed during the thesis research for performing these steps; Section 4 then shows how the framework was demonstrated.

This development process might proceed at a leisurely pace if it weren't for competitive time-to-market pressures in the automotive industry. The DIRECT ENGINEERING[SM] semi-automatic product development environment is meant to meet this need. In the hands of a knowledgeable engineer, a DE[SM] application can develop new assemblies remarkably quickly. The problem this thesis addresses is how to realize this speed advantage when a significant component is supplier-designed, while supporting the systems engineering process.

Systems engineering demands comprehensive specifications; competitive pressures and the DE[SM] process demand rapid communication. These needs are accommodated by separating component specifications into two complementary pieces: (1) a shared component framework or architecture (which will be called a variant component attribute structure here), and (2) the set of unique information defined within this framework which must be communicated at each design iteration. The shared component framework applies to all instances of the item being designed; the set of unique information applies to a specific instance.

The first piece of the component specification- the shared component framework- is readily

established for variant parts. Such a framework isn't required for catalog parts as their design is fixed, and is too restrictive to be useful for truly custom parts. As described above, variant parts are unique combinations and variations drawn from an existing set of design options. A shared variant part framework simplifies writing component specifications as described by example in Section 3.3.1. An approach for identifying the second piece of the component specification- the unique information communicated at each design iteration- is presented in Section 3.3.4.

3.3.1 Defining a variant component attribute structure

Using a variant part requires a well-defined and shared understanding of its architecture, available design options, and its component-to-surroundings interfaces. The architecture and available design options can be codified; this component design strategy is here given the name variant component attribute structure. The component attribute structure conforms to the part's location within the customer's system model (Section 2), as well as the supplier's design and manufacturing processes. Therefore, the variant part's attribute structure is developed jointly with input from both parties.

Fortunately, the attribute structure concept fits nicely into parametric, features-based design processes. And variant components are central to the DE[SM] process. Therefore, extending this concept to supplied parts is straightforward.

Figure 3-2 illustrates (in abbreviated form) an example of a attribute structure for the Throttle Return Spring. It is shown as a "tree" of design options; all terms have shared customer and supplier definitions. All allowable options or ranges are predetermined. This is not as restrictive as it sounds: the attribute structure shown can result an infinite variety of unique, dissimilar-appearing springs.

Why develop a component attribute structure? For these reasons:

- It identifies and records the Variant component design framework- the architecture, all design options, and how the options interact. Not all options are compatible, as seen for the End Types in Figure 3-2. The component attribute structure clarifies component discussions.

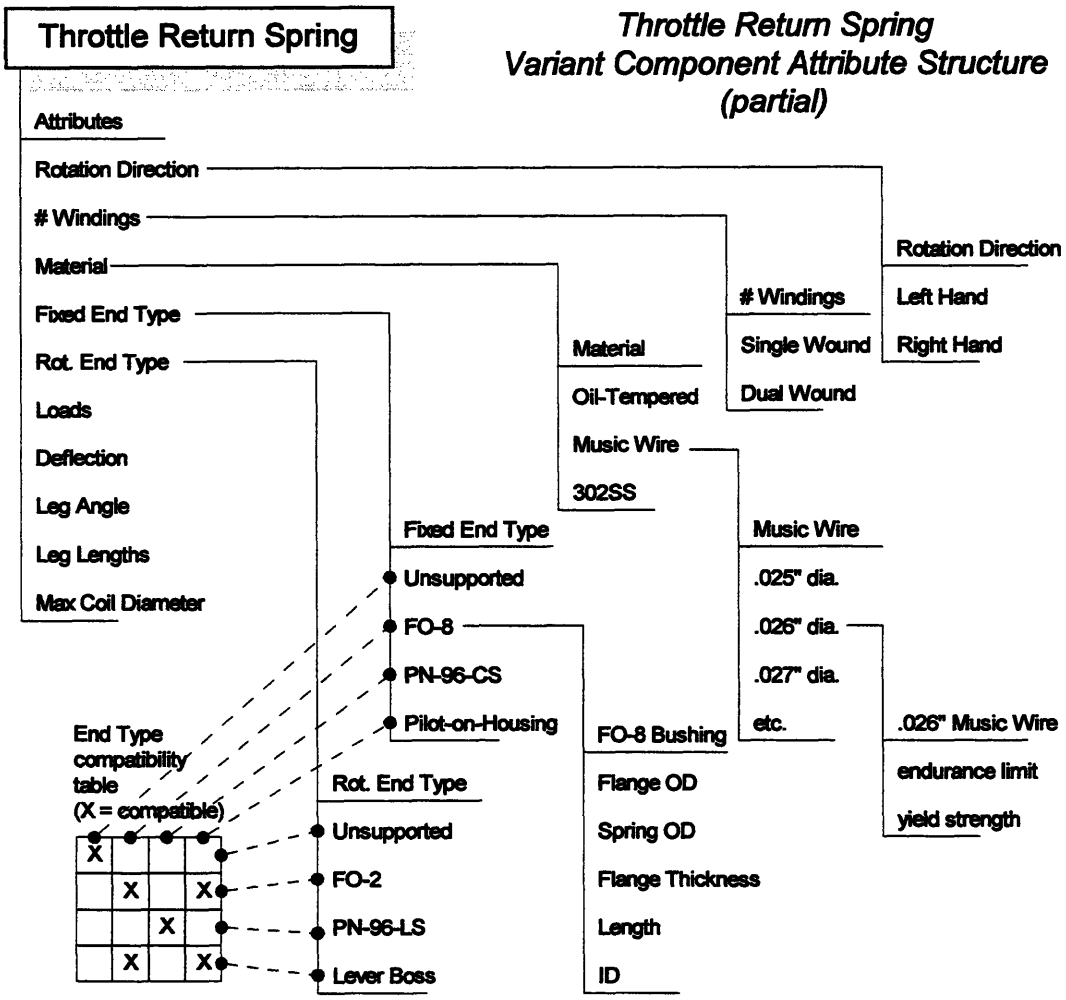

Figure 3-2 Component attribute structure example (for Throttle Return Spring)

- The component attribute structure determines manufacturing process capability needs. And it insures that requested parts conform to the established manufacturing process.
- The component attribute structure makes communication much easier, which is shown in Section 4. Since both the customer and supplier share the same understanding of the component's general configuration and the meaning of selected options, they can simply exchange option selections instead of a drawing or solid model.

Developing the variant component attribute structure is a necessary first step when introducing a supplier designed component into the DE[SM] process.

3.3.2 Customer: interface and simplified representation

The customer is uniquely interested in and capable of understanding the component's fit into its surroundings. However, the customer is neither aware (nor particularly interested) in the intricacies of the inner workings of the component- this is the supplier's concern. What the customer needs in order to work with the component is:

- A model of the component's functional interactions and interfaces with its surroundings: the system model.
- A model of the component, with sufficient detail to insure that all interfaces and performance characteristics are properly accounted for.
- A capability for writing the specification for the component: turning the interface and surrounding parts into component requirements.

These are illustrated in the left-hand side of Figure 3-3. The customer manages the integration

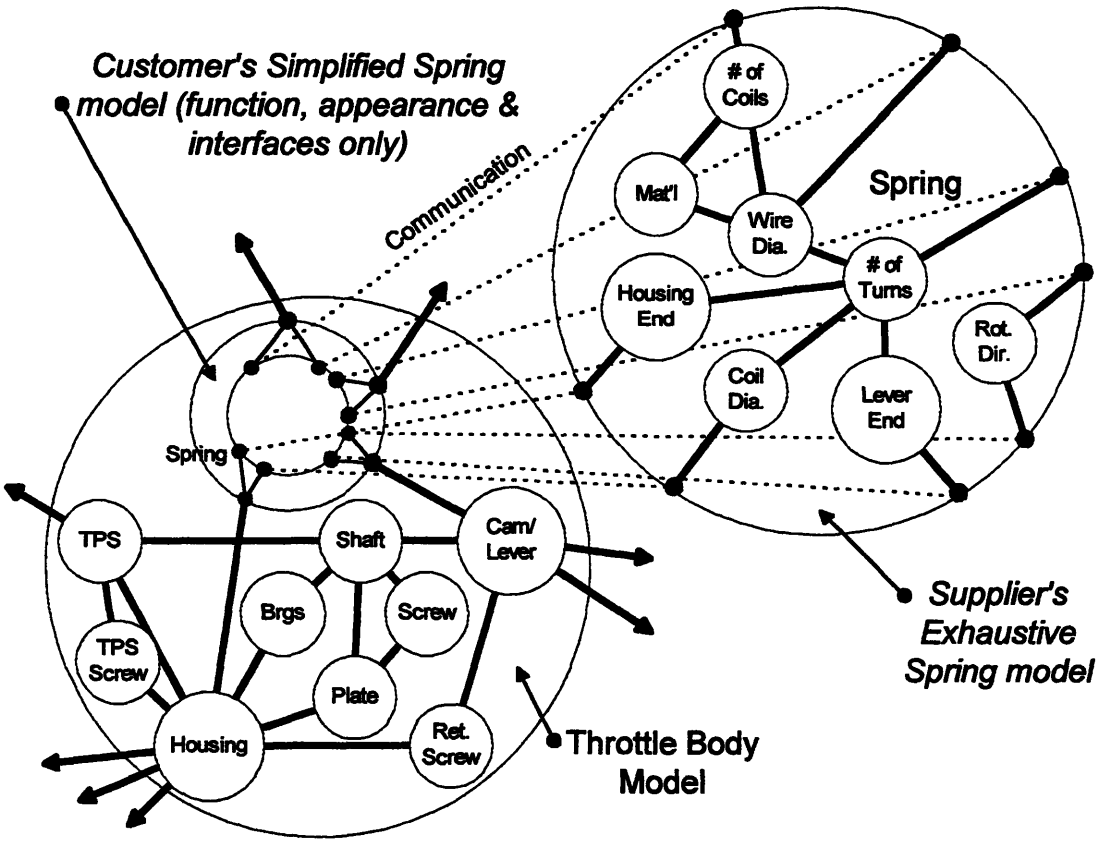

Figure 3-3 Customer and Supplier contributions to the Supplied-Component Framework

and the interfaces, but not the component design.

The customer's simplified Spring Model looks like the spring in all relevant ways, but is not the spring. It lacks unnecessary details. For instance, the coils are modeled as a cylinder, not a specially created helix. It is really just a shell, a simplified representation of the spring which contains only needed information and which converts the Spring's surroundings into Spring requirements.

Thus, the customer manages everything about the spring that isn't the spring. The supplier manages the spring. The customer manages the system model and integrates the spring into this model, while relying on the supplier to understand its stated spring requirements and perform spring design. In this process, the customer's initial determination of spring requirements is unwittingly limited by a lack of knowledge of supplier capabilities or springs in general. This contributes to the observed need for iteration of both requirements and design.

3.3.3 Supplier: exhaustive component model

The supplier is uniquely able to model the intricacies of the component and to perform component design, but is unable to determine the component's requirements and expects them to be stated by the customer. The supplier contributes to component architecture definition, and (in my observations) often learns enough about the customer's engineering concerns to offer valuable advice about the surrounding assembly. This is reinforced by differences in department tenure between supplier and customer engineers- a topic discussed in Section 5.2 below.

The supplier designs the spring using an exhaustive component model, which is also the system model at its level of decomposition. This system model may be found in engineers' memories, in design guides, or in complex software tools. Requirements come from above (the customer); if there are supplied components within its model, requirements are specified to levels below. For the Throttle Return Spring, the only supplied components were off-the-shelf plastic bushings, and no lower-level system models were used.

The supplier's knowledge is local to its component. But the supplier "knows" things that can help the customer optimize the overall system design for weight, size or cost. The supplier's

component model probably involves expertise which is not transferable to the customer due to its experience-based or proprietary nature. Then optimizing the assembly requires supplier involvement through repeated iterations. How much does a customer such as Ford need to know about the supplier's processes to reach a satisfactory outcome? Not very much. Ford is relying on the supplier to provide skills which Ford doesn't have or has chosen to not develop. During my internship, it was clear that Throttle Body engineers understood spring design in only general terms and must rely on the supplier. This is reasonable when the two organizations have developed a mutually beneficial relationship and some degree of trust, as was discussed in Section 3.2.3 above.

3.3.4 Identifying what is communicated: attributes describe the component

The pieces of the supplied component development framework described thus far are the variant component attribute structure, the customer's simplified component model, and the supplier's exhaustive component model. The next step is to identify what needs to be communicated between the supplier and customer. This communicated information was classified by introducing the concept of an "attribute language." Attributes are names assigned to every component feature- those that interact with the surrounding assembly, and those that are internal to the component. The variant component attribute structure described above lists the attributes which the customer and supplier both care about- and their allowable values. Types of attributes are:

- Descriptive. A word that conveys a feature of the part that is more than a single value. For example, Spring-Fixed-End-Type = "FO-8" defines a set of interface dimensions for both the spring and the mating surfaces of the Throttle Body housing.
- Numeric. An attribute assigned a value. Examples are Length, Installed-Load, or Wire-Size. Numeric attributes can be bounded (Length > 0.0mm) or discrete-valued (wire is available in .001" diameter increments).

The attribute listing for a component is a method of communicating the requirements and the part's configuration, performance, and interface. Therefore, shared definitions are absolutely necessary- the supplier and the customer must agree on the meaning of every attribute.

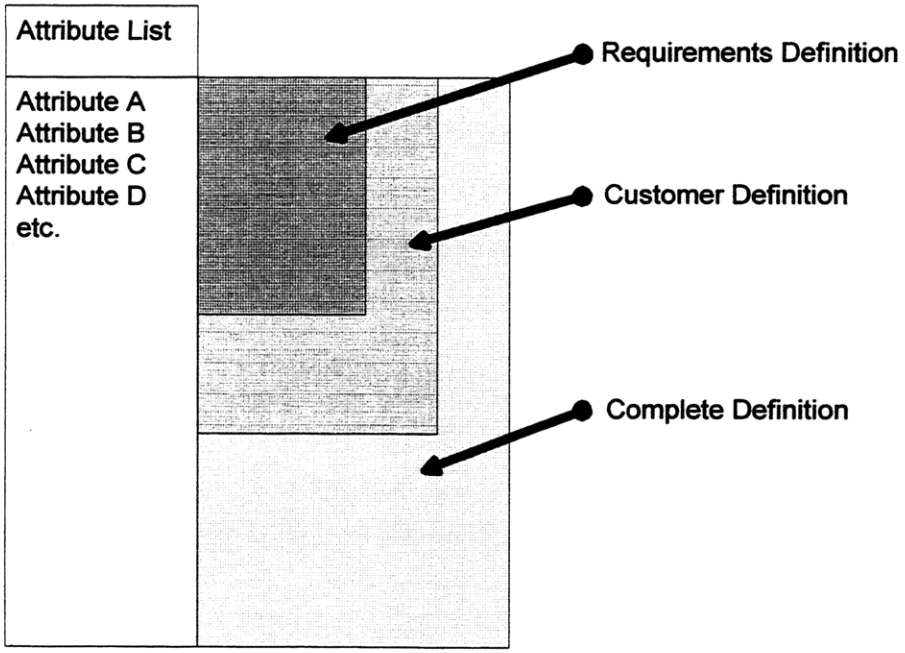

Figure 3-4 Attribute Classes

3.3.5 Classifying attributes

Some attributes are specified by the customer. Others are determined by the supplier when the part is designed. For instance, the customer specifies a Throttle Return Spring's required rate, deflection, and overall dimension limits, but has no idea what the spring's wire size will be given these requirements. Yet the customer needs the wire size to create a CAD model of the spring. The supplier receives the requirements and identifies the wire size while designing the spring, and communicates this value along with the rest of the spring design.

This example suggests attributes can be classified in terms of who controls, and cares, about them. I developed three classifications: complete definition, customer definition, and requirements definition. Each of these attribute classes is a subset of the former class; see Figure 3-4.

- Complete definition. All the information needed to fully define and manufacture the part. This is the information managed by the supplier (the development organization)- normally seen on the comprehensive collection of drawings that define the part.

- Customer definition. A subset of the complete definition. All the information that the customer needs to incorporate the part into its assembly. Therefore all interface, dimensional, performance information. This information is traditionally on an "installation" drawing.

- Requirements definition. A subset of the customer definition which conveys the complete component specification; sometimes called "functional requirements." Note this is not all the information that the customer must have about the part. Because the supplier defines the part, the customer cannot know some aspects of it before receiving a customer definition created in response to a requirements definition.

Table 3-1 illustrates these attribute classifications using the spring. Each successive definition contains greater amounts of information. The customer specifies the requirements, but many

Spring Attribute List (Abbreviated)			
Attribute List	Requirements Definition	Customer Definition	Complete Definition
Installed Load	■	■	■
Max Work Load	■	■	■
Working Deflection	■	■	■
Installed Deflection			■
Direction of Rotation	■	■	■
Maximum OD	■	■	■
Actual OD		■	■
Mean Free OD			■
Maximum Confined Length	■	■	■
Minimum Confined Length		■	■
Free Length			■
Leg Length	■	■	■
Fixed End Bushing Type	■	■	■
Number of End Coils			■
End Coil Diameter			■
End Coil Orientation			■
Design Life	■	■	■
Stress Margin at Max Work		■	■
Spring Wire Diameter		■	■
Material	■	■	■

Key: ■ Included in List | Not Included

Table 3-1: Abbreviated Spring Attribute List

of the design characteristics that are needed to evaluate the design are determined by the supplier. In addition, the supplier manages a lot of information that the customer has no use for- things that are internal to the design, such as the diameter of the end coils that wrap around the end bushing. These internal features are of no interest to the customer. From a practical standpoint only the Requirements Definition and Customer Definition are important to the communication process; these are defined by the variant component attribute structure. How these attribute lists were communicated is described in Section 4.

3.3.6 Organizational barriers to developing a component framework

Creating a variant component attribute structure such as presented above demands the cooperation of both customer and supplier engineers: all the "stakeholders" must understand the reasons for and be convinced of the merits of doing so. In the writer's experience with the Throttle Return Spring, customer engineers in particular did not see this framework for design as particularly important, and did not contribute to its development. Why this might be so is discussed in some detail in Section 5. Creating component frameworks- the variant component attribute structure- may require a change in attitude towards component design.

3.4 SUPPLIED COMPONENTS- CONCLUSIONS

This section outlined a process for integrating supplier-designed components into a systems engineering process such as the DIRECT ENGINEERING^{SM} process. A key part of systems engineering is writing good specifications, and a process was developed to do just that while supporting the DE^{SM} process's need for rapid communication and design turn-around. The identified process- which was developed and demonstrated for the Throttle Return Spring- is applicable to variant components designed in an iterative process. Iterative design of a supplied component consists of the following steps:

- The customer determines the supplied product's requirements (specification).
- The customer communicates the requirements to the supplier.
- The supplier designs a component that conforms to the requirements.

- The supplier communicates the component design back to the customer.
- The customer integrates and evaluates the component in its system, determining if the design is satisfactory. If not, the supplier will be asked to try again, possibly with revised requirements.

These steps are repeated until the design is accepted as complete; each loop is one design iteration.

The identified process for supporting supplier-designed component development in the DE[SM] environment consists of the following "building blocks:"

- The variant component attribute structure (Section 3.3.1; Figure 3-2)
- The customer's simplified component model (Section 3.3.2; left side of Figure 3-3)
- The supplier's exhaustive component model (Section 3.3.3; right side of Figure 3-3)
- Communication process using attribute language (Sections 3.3.4 & .5; Figure 3-4)

These building blocks were demonstrated for Throttle Body Return Spring design, as described in the following section.

Throughout this section, the organizational environment within which the product development process will be implemented was discussed. Central to successfully introducing this process is establishing commitment and an attitude of trust at both the customer and the supplier. Given the history of automotive supplier relationships, this alone is a challenging goal.

4. DEMONSTRATED PROCESS FOR RAPID SUPPLIED-COMPONENT DEVELOPMENT

As described in Section 1, the goal of this work was to develop a method for speeding the development of a supplied component- the Throttle Return Spring- in the context of a semi-automated product development environment called the DE℠ process which significantly enhances the speed at which Throttle Bodies can be designed.

Sections 2 and 3 described the context of this work in a general way. The Throttle Body is an assembly of numerous components arranged in an fairly fixed architecture. Each of these components is adapted to a new vehicle application using systems engineering. In systems engineering complex assemblies are recursively decomposed into simpler systems, sub-systems, and components with clearly defined interfaces and requirements. The components are then designed individually and re-integrated to produce a complete product.

Effective systems engineering demands effective specification writing: a component's interfaces and requirements must be clearly communicated to the organization designing the component. This is a key step for a supplier-designed component, whether the supplier is an internal engineering group or a separate organization. It is simplified when the supplied component can be described using a variant component attribute structure. In this case, a component's specification can be split into two pieces: an attribute structure shared by the customer and supplier, and a simple list of attribute choices and values.

This section describes the work that was completed to implement these ideas in a working development process for Throttle Return Springs and Throttle Bodies. The process works effectively between two very different companies: one very large, one very small; one an aggressive adapter of technology, the other sticking to PCs. The "high-level" attribute approach to component specification readily bridges the technology and skills gap between these organizations. And the process significantly improved design turn-around speed, by enabling very rapid communication and product design. It is presented as a model for other similar relationships. For lack of a better name I have called it the Rapid Development Process.

Section 4.1 presents a Rapid Development Process scenario, describing the activities at both the customer and supplier. It then presents the elements of the design process in detail. Section 4.2 presents measured results of applying the process. Section 4.3 assesses the application of the Rapid Development Process in other areas of an automotive supply chain.

4.1 THE RAPID DEVELOPMENT PROCESS

The Rapid Development Process is introduced through an example below. Figure 4.1 illustrates the communication steps in the Rapid Development Process.

The Rapid Development Process begins at Visteon in Rawsonville, MI when a customer requests a new Throttle Body design. A Throttle Body engineer receives the new Throttle Body requirements, and turning to a nearby workstation, starts up the Throttle Body DE[SM] Application. A "baseline" Throttle Body solid model with all components is immediately visible. Selecting each component using the mouse opens its "graphical user interface" (GUI) which allows the component to be modified to conform with the customer's requirements.

The engineer looks at the Throttle Return Spring by selecting it and opening the Return Spring GUI. This sheet displays all of the relevant parameters for the spring. Some are not editable: they are linked to other components in the Throttle Body. Some can be selected from a pull-down options list. Others have open fields where a numerical value can be typed in. The engineer notices there are two columns: one for requirements (which are being edited), and one for an actual design. The design parameters are set to default values, and a note states that "No spring has been designed yet."

The engineer inputs all the spring requirements- "fills in the blanks"- and exits from the Spring

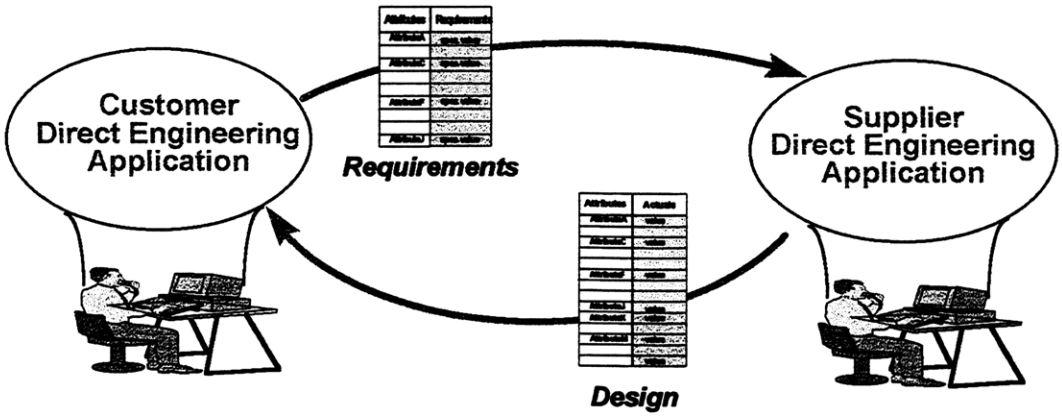

Figure 4-1 One iteration of the Rapid Development Process

GUI. Selecting a Reports button then writes all of the spring requirements just specified into a simple text file. This file is attached to an email and sent to the spring vendor. In Figure 4-1, the left and top half of the loop has been completed.

The Rapid Development Process now shifts to the spring vendor, Michigan Spring of Muskegon, MI. A vendor engineer opens the email, retrieves the attachment, and then opens the Torsion Spring Designer. This Microsoft Excel workbook incorporates worksheets for each spring design activity. By selecting a macro button, the email attachment from Rawsonville is automatically opened and the requirements are loaded into the spring design workbook.

Next, the vendor engineer uses an automatic macro to search a stored spring database for existing springs that might be used in this new application. In this instance, none are found, and the "Design New Spring" button is selected, starting a macro that iterates through the spring design process using the customer-specified requirements. The engineer evaluates the resulting spring, and once satisfied, selects a "write output file" button which creates a new text file containing all pertinent spring design attributes. This file is attached to an email which is returned to the Throttle Body engineer at Rawsonville. The right-hand and lower half of Figure 4-1 has been completed.

When the email arrives at Rawsonville, the attachment is opened from within the Spring GUI of the Throttle Body DE[SM] Application. All of the spring attributes are automatically loaded into the design column on this sheet, and the solid model of the spring is updated to conform to the actual design. The Visteon engineer reviews the spring, confirming its match with the requirements and its fit with surrounding components.

A complete design iteration (one loop around Figure 4-1) has been completed in less than a morning. The process may repeat if the Visteon engineer is dissatisfied with the spring design or receives requirements changes from its customer. Then one or more of the spring requirements would be edited, and another iteration performed.

4.1.1 The Customer's role in the Rapid Development Process

The Throttle Body DE[SM] Application at Visteon-Rawsonville contains the customer's part of

the Rapid Development Process. It is important to note that the Rawsonville engineers have chosen to not be spring experts and rely on the supplier to design the spring in response to their requirements. Using the Rapid Development Process enhances the supplier's capabilities sufficiently to support the speed of the DE[SM] application.

Throttle Body DE[SM] Application: a semi-automatic CAE development tool

Many details of the DIRECT ENGINEERING[SM] process are proprietary to the Ford Motor Company, and this discussion is restricted to areas pertinent to describing the Rapid Development Process. The Throttle Body DE[SM] Application contains a complete solid model of the Throttle Body and all its components including a simplified model of the Throttle Return Spring. These are far more than simple CAD solids. Instead, the DE[SM] application contains the entire systems model of the Throttle Body: the left side of Figure 4-1 embodies the left side of Figure 3-3. This systems model, including part-to-part parametric relationships, performance interactions, and so forth, is "in the background" but is tied to the solid model. Linked to the systems model is a great deal of product and process knowledge, which can be accessed by a product engineer very rapidly. In this way the DE[SM] vision of bringing Rawsonville's collective Throttle Body expertise to bear on a new design is realized.

Each component in the DE[SM] application can be edited by selecting it directly, which opens the component's GUI (graphical user interface). The GUIs display different types of information depending on the component chosen, but typically display all the relevant component attributes- some of which are editable, and some of which are not. The Throttle Return Spring is the only component designed by a supplier for each new Throttle Body, and it is managed differently from the others. The design sequence below describes how the Throttle Return Spring is incorporated into the Throttle Body DE[SM] Application.

Development process sequence

The Throttle Return Spring development sequence is illustrated in the flowchart on the left side of Figure 4-2. The right side of the figure shows the status of the Requirements and Design Attributes, and the Geometry, at each step in the process. Some Requirements Attributes (such as length) are established by assembly relationships in the Throttle Body

systems model; other attributes (such as installed load and spring rate) are input by the engineer based on performance criteria. These user-edited Requirements Attributes are input using numeric fields or drop-down option lists as determined by the spring's attribute structure.

The spring's Design Attributes aren't edited by the engineer: they can only be assigned by the input of an actual spring design provided by the supplier. This results from Rawsonville's inability to design the spring, or even be certain about a given spring's dimensions. After the supplier responds to a design request the Design Attributes conform to an actual spring's dimensions (such as outside diameter, wire size, and installed length).

The Throttle Body design process begins when Rawsonville's customer (typically Ford's Power Train Operations, or PTO) requests a new Throttle Body. At the outset, the DIRECT ENGINEERING[SM] application displays a default Throttle Body. This is rapidly altered as the engineer inputs new application-specific requirements, including those for the spring. Since the spring is a visible part of the Throttle Body solid model, but no spring has been designed, the spring's geometry is driven by the Requirements Attributes and default values. Doing this lets Throttle Body design proceed before the spring supplier is involved.

Once the spring requirements are complete, the next step is communicating them to the supplier. The communication process is described in Section 4.1.3 below. After designing the spring, the supplier communicates its design back to Rawsonville. An automatic process loads the supplier's response into the spring Design Attributes, and now both Design and Requirements Attributes are defined. Also, the spring geometry can be toggled between the Design and Requirements Attributes, so the exact appearance of the spring can be visually inspected, and compared to the design that was requested.

At this point, the Throttle Body engineer evaluates the spring design against the spring requirements: do the spring's Design Attributes equal its Requirements Attributes? Possible outcomes are:

- The supplier designs an exact match; Design Attributes = Requirements Attributes.

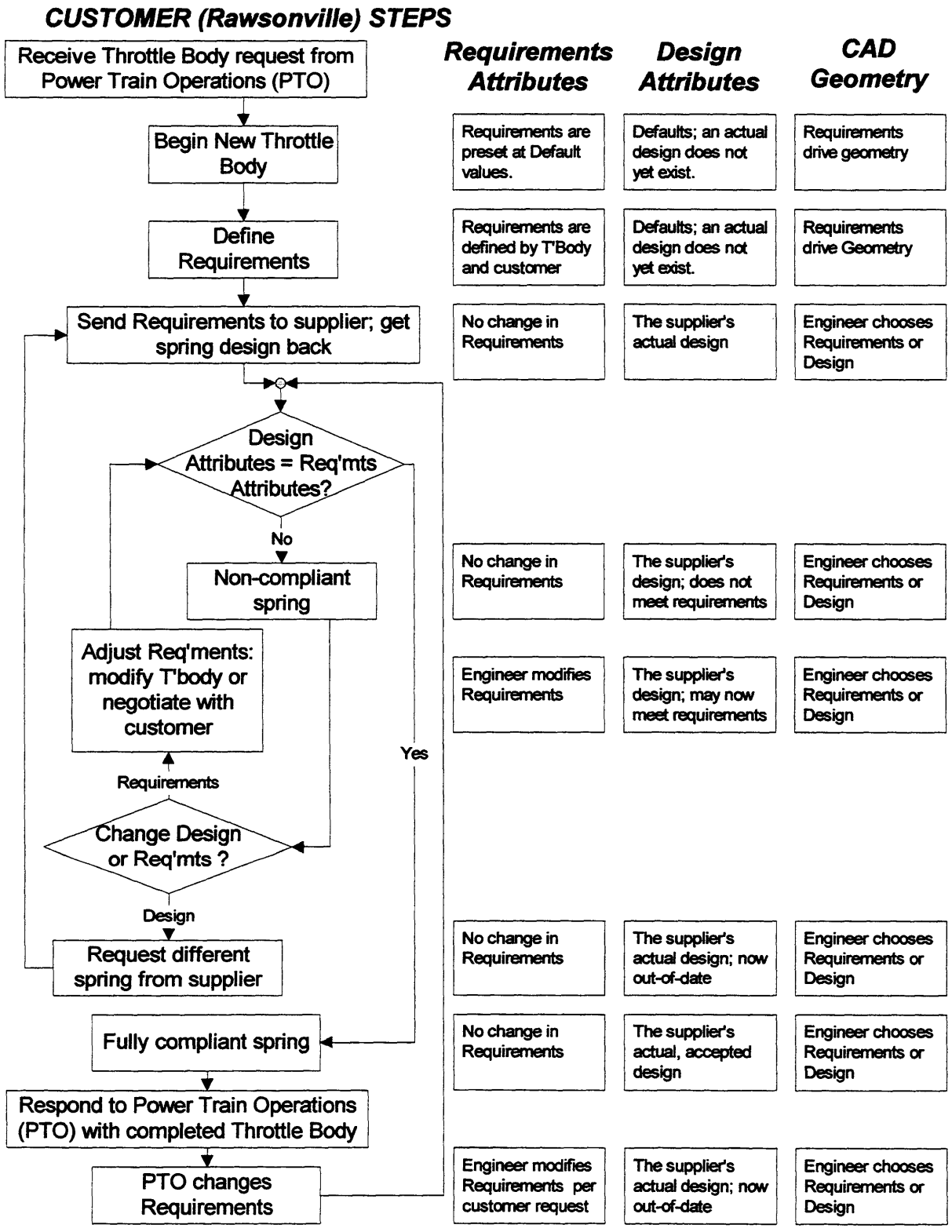

Figure 4-2 Customer Spring Design Process Flowchart

- No spring will meet the requirements within the space allotted. The supplier provided a spring that is as close to the requirements as possible. For instance, all requirements are met except spring length, which is exceeded. The result: Design Attributes ≠ Requirements Attributes
- A close match is found in the existing spring database (it is highly improbable that an exact match would be found). For instance, all dimensional requirements are met, but the spring rate is a few percent too low. The result: Design Attributes ≠ Requirements Attributes

The first outcome is easy- the spring meets all requirements and the design process is done. As shown in Figure 4-2, this tidy conclusion may well be upset when Rawsonville's customer imposes a requirements change. As a result, through no fault of the Rawsonville or Michigan Spring engineers, Design Attributes ≠ Requirements Attributes, and the spring is non-compliant. This occurrence imposes the same decisions on the Rawsonville engineer as the second and third outcomes above.

When the spring is non-compliant (when Design Attributes ≠ Requirements Attributes) the Throttle Body engineer needs to change *something* to reach compliance. The decision is not simply, "make the supplier do a better job," because the spring as originally requested may well be physically impossible. Instead, the engineer must decide whether to change the Requirements or the Design. The engineer's experience, judgment, and the various help tools embedded in the DE[SM] application are all tapped to aid this decision.

Several factors may suggest the best choice is changing the Requirements. For instance, an existing spring that is a close match could offer sufficient cost savings to justify asking Rawsonville's customer to tweak one of their load requirements. Or the shaft could be extended a few millimeters to accommodate an otherwise acceptable spring. For each of these decisions, the Throttle Body engineer has to weigh the difficulty of changing the requirements against the comparative cost of a fully compliant spring.

Often changing the spring will be the most practical approach- and the supplier will be asked to provide a new, compliant design rather than an existing design that is not quite close

enough. It merits repeating that this supplied component cannot be modified directly by the customer engineer- there is no in-house capacity for spring design. The spring can be modified only by going around the design loop once again.

The Throttle Return Spring design process concludes at Rawsonville when the product engineer accepts a spring design as final. Subsequent steps (engineering sign-off, procurement, etc.) then proceed.

The above discussion demonstrates that maintaining separate Design and Requirements Attributes lists supports differences between spring requirements and actual spring design at every step in the development process, permitting (for instance) evaluation of a "close-but-not-exact" spring from the existing spring database. In short, using two attributes lists supports the fundamental difference between a specification and an actual design that is inherent in systems engineering. At the same time, the two lists insure that the really challenging "requirements versus design" decisions are clearly spelled out and left to the customer engineer's discretion.

4.1.2 The Supplier's role in the Rapid Development Process

In a broad sense, the supplier has to identify with the goals of the process. The supplier has to develop a disciplined approach to product design and help develop the supplied product's variant attribute structure. These steps are sufficient if the speed of existing design processes is satisfactory. But in the DE[SM] world, the customer's product development speed is 10 times faster than former practice. To actually participate in the Rapid Development Process, the supplier must develop and maintain a process which expedites its design process as well.

For the demonstrated example, the supplier's speed was enhanced using a component-development software tool which automated the Throttle Return Spring design process. Created as a Microsoft Excel workbook, the "Torsion Spring Designer" embodied the component systems model and its variant attribute structure. Features of the workbook included automated input of Requirements Attributes, output of Design Attributes, searching of an existing springs database, and design of new torsion springs.

Structured input (requirements) and output (design definition)

The supplier gets the Requirements Attributes from Rawsonville in the attribute terms defined by the shared variant component attribute structure (discussed in Section 3.3 above). Included are all attributes in the "Requirements Definition" column of Table 3-1. The requirements are automatically read into the Torsion Spring Designer using a Visual Basic macro. After the design process is complete, the resulting spring is communicated back to Rawsonville as Design Attributes, using the same variant component attribute structure. Included are all attributes in the "Customer Definition" column of Table 3-1. A Visual Basic macro automatically creates the output file. More details of the communication process are presented after a discussion of the Excel workbook below.

Contents of supplier's semi-automatic spring development tool

The supplier's Torsion Spring Designer is in essence a simplified DE[SM] application developed

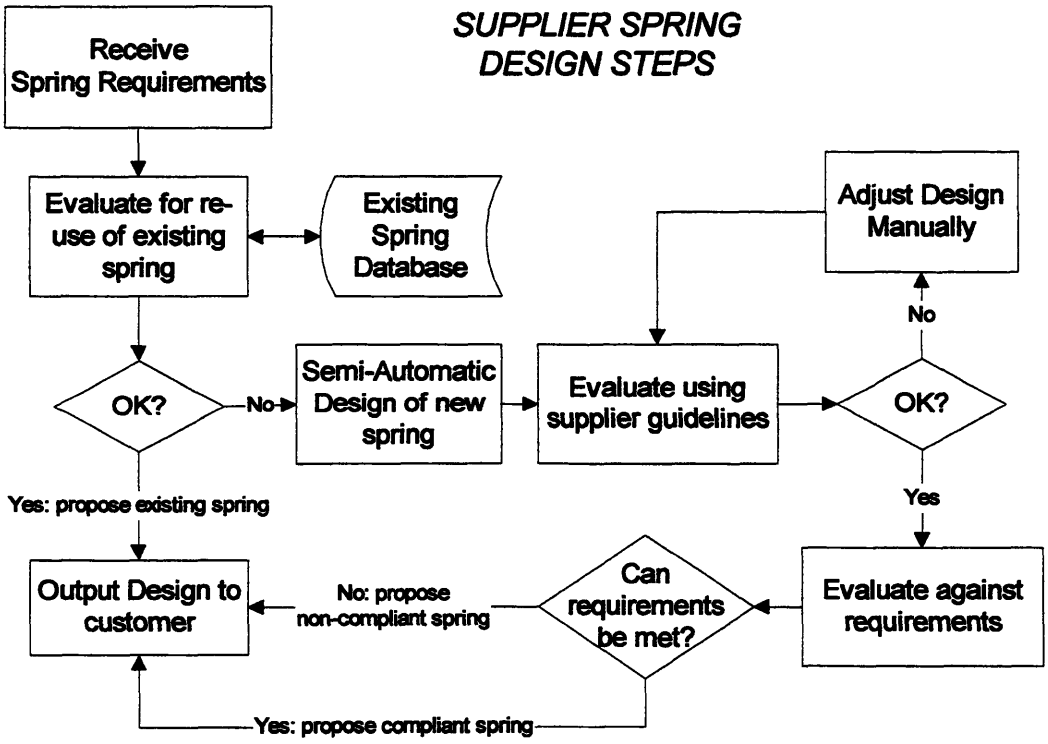

Figure 4-3 Supplier Spring Design Flowchart

in Microsoft Excel. It embodies the complete systems model of the spring; the right side of Figure 4-1 contains the right side of Figure 3-3. This systems model includes the engineering relationships that define the spring and supports all of the options defined by the variant component attribute structure. It incorporates a software process for determining optimal spring design within manufacturing capabilities by performing the laborious, iterative calculations inherent to spring design. However, it is not completely automatic, as the supplier engineer's expertise is needed to evaluate and approve the final spring design.

Because the re-use of existing components in new applications can be very cost-effective, the Torsion Spring Designer also supports intelligent searching of an existing / carryover spring database. This feature alone attracted significant interest from Ford Procurement. Searching through existing spring drawings was formerly a difficult, laborious process; it now takes 10 seconds of computer searching and then 5-10 minutes of engineering evaluation to compare all the existing springs against the new requirements.

Describing the contents of the supplier's Torsion Spring Designer would take a great deal of space and reveal some of the supplier's proprietary design processes. Suffice it to say that this Excel Workbook contains approximately 10 worksheets and perhaps 20 Visual Basic macros and subroutines. Some of the worksheets are formatted repositories of design alternative information as determined by the variant component attribute structure (for instance, a list of Fixed-end Bushing options). One of the worksheets manages the spring design equations, another performs searching and evaluation of existing springs against the customer requirements. Figure 4-3 illustrates a flow chart of steps the supplier takes in using the Torsion Spring Designer to develop a new spring. Not surprisingly these are the same steps observed in a manual process.

4.1.3 Communication in the Rapid Development Process

The Rapid Development Process demands communication of customer requirements to the supplier and component design definitions back to the customer. What is actually transmitted between the supplier and the customer, as discussed in Section 3.3, are Requirements Attributes (the Requirements Definition) and Design Attributes (the Customer Definition)

Customer,Ford
Program-name,Test
Customer-contact,Jared Judson
Customer-part-number,Test-0001
Date,13 November 97
Load-at-installed-required,.23
Load-at-max-work-required,.33
Working-deflection-required,78
Direction-of-rotation-required,Right-Hand
Winding-type-required,Single-Wound
Max-outside-diameter-required,30
Max-confined-length-required,20
Leg-angle-at-installed-required,305
Fixed-end-leg-length-required,25
Rotating-end-leg-length-required,25
Fixed-end-bushing-type-required,FO-8
Rotating-end-bushing-type-required, Reduced-Coil
Material-required,Music-Wire
Design-life-required,Infinite-life
Additional Comments,Trial run only.

Figure 4-4 Email file of Customer Requirements

using a shared attribute language. This information is easily exchanged due to its simple format.

For the spring, the list of attributes is pretty short, and a telephone call or fax might suffice to communicate them. In interest of testing as "hands-off" a process as possible, the demonstrated process chose to exchange information with email, using very simple comma-delimited ASCII text files. All the information- the attributes and their values- could be characterized using text strings, either words or numbers. To simplify programming the automated input and output procedures, an agreed upon attribute sequence was specified. Then both the customer and the supplier developed code to read and write compatible files. Figure 4-4 illustrates a typical Requirements email file written at Rawsonville and ready to be automatically read by Michigan Spring. The return email containing all Design Attributes is slightly longer because the Customer Definition contains more attributes.

By using systems engineering to develop and apply a variant component attribute structure, very small, "universal format" files contain all required component specification and definition information. There is no need to impose hardware and software demands on suppliers or deal with cumbersome conversion between incompatible CAD solid models. These are very

attractive characteristics of the Rapid Development Process.

4.2 OUTCOME: 10X REDUCTION IN DESIGN CYCLE TURN-AROUND TIME

The outcome of a series of trials conducted at Visteon and Michigan Spring in November and December of 1997 are shown in Table 4-1. These were "realistic" trials in that no effort was made to interrupt engineer's normal activities to wait for an email. The process described above supports continual iteration, but in actual practice only one or two iterations were required to reach a satisfactory spring. And demonstrated iteration times were very quick. The former process needed 2-3 weeks to complete a new spring design. The new Rapid Development Process showed an average Throttle Return Spring design turn-around time of 2-3 hours. If the customer and supplier engineers were coordinated, a 20 minutes turn-around time could be achieved.

Given this process, Throttle Return Spring development is able to keep up with the speed demands of the DIRECT ENGINEERING[SM] semi-automatic Throttle Body development process. In addition, the quality of the communication is better: more information with clearer definition is communicated with less effort. Thus, rapid, accurate, high-content design iteration is created and supported through the Rapid Development Process.

4.3 INTEGRATING RAPID DEVELOPMENT ACROSS A SUPPLY CHAIN

The proposed Rapid Development Process incorporates systems engineering while enabling much improved product development speed. It allows the DIRECT ENGINEERING[SM] vision to be extended outside of Ford's boundaries. It delivers clear, organized product information at all stages of the development process. However, it is not applicable to every product development situation. This section notes where it can be applied and lists some of the organizational benefits of using the process.

Trial #	Number of Iterations	Elapsed Time
2	2	2:41 (hours:minutes)
3	2	6:27
5	1	0:32
7	2	2:35

Table 4-1 Spring Design Iteration results using the Rapid Development Process

4.3.1 Where the Rapid Development Process can be applied

The Rapid Development Process is most applicable to variant, iterative parts. Rapidly evolving products, or products that are custom designed are not good candidates. It can be applied arguably at any level of a systems engineering decomposition.

Variant components, Iterated Design

The Rapid Development Process applies to variant products that are designed in an iterative process. A variant component can be assigned an attribute structure which becomes the "background specification." The communication process then simply transmits attribute values. However, the Rapid Development Process is vulnerable to product evolution which causes the attribute structure to change, or when the knowledge underlying the product and process is changing rapidly. When this occurs, both the customer and supplier must revise their shared understanding of the product and probably update their software programs that embody a rapidly obsolescing attribute structure.

Applicable at any level of design decomposition

The Rapid Development Process was demonstrated at the Component level of a systems engineering decomposition. It is equally applicable at higher levels, such as the assembly or sub-system level. This is because system engineering problems look the same at each level of decomposition. At Ford this is referred to as the "fractal" nature of systems engineering.

The challenge at higher levels in the decomposition is determining a well defined architecture and variant attribute structure. Since higher decomposition levels often show more architectural variety, this may be difficult. And more complex products are likely to have lengthier attribute structures which increases the effort needed to identify and codify them. Lastly, higher decomposition levels contain many sub-components and are more likely to be vulnerable to the product evolution discussed above. Nevertheless, assemblies such as the Throttle Body should fit into an "Air Intake System DE[SM] Application" in the same manner as the Throttle Return Spring fits into the Throttle Body DE[SM] application today.

4.3.2 Business relationships in the Rapid Development Process

The Rapid Development Process demonstrated above allows product development to proceed much more quickly than the process it replaces. It benefits both customer and supplier while addressing intellectual property concerns, and is sensitive to the local capabilities and skills of the existing organizations. In short, as the paragraphs below discuss, the Rapid Development Process was designed to closely fit the existing organizations rather than requiring an entirely new culture.

Customer and supplier interests are maintained

Using the Rapid Development Process to develop supplier designed components allows Rawsonville's expectations for response time to be exceeded. At the same time, more and better information is exchanged with the supplier. The speed and accuracy of Rawsonville's overall product development process is enhanced.

At the supplier, Michigan Spring, implementing the Rapid Development process greatly improves responsiveness toward many customers, not Ford alone. The "Torsion Spring Designer" was developed to support both the specific Rawsonville attribute structure and generic torsion springs as well. This leverages the supplier's investment in development and maintenance across multiple customers- all of whom can expect greatly improved design responsiveness. Also, the number of engineering hours required to complete a new product design are very small, so more requests can be supported within the limited hours available to the engineering group.

Proprietary information and trust

As demonstrated, the Rapid Development Process leaves proprietary knowledge where it belongs. Ownership and maintenance of both the supplier's and customer's knowledge stays within their respective organizations. As this was gradually realized during the internship, the level of trust between the two organizations was observed to increase. Initial meetings with the supplier were candidly tense. The supplier was uncertain about Ford's intentions and not quite ready to believe that this activity wasn't an effort to "steal" knowledge. As the supplier

saw how the new process would work, its concerns subsided.

At the same time, Ford's early expectations that the supplier had to reveal all its knowledge in order to be trusted gradually relaxed. Despite verbally supporting trust and long-term relationships, some Ford engineers and managers initially felt that the only way to get an "optimal" design was for the supplier to give its product development processes and deeply held knowledge to Ford. These beliefs subsided as they realized that the supplier's expertise was both valuable and unlikely to be matched within Ford, and that the supplier could turn designs around very quickly, obviating the need to have Ford develop this ability in-house.

Process complexity is sensitive to local capabilities

As discussed in Section 3.2.4, there are wide differences between the product development environments observed in an automotive supply chain. By structuring and communicating supplied component information using a "high level" shared attribute structure and language, the Rapid Development Process readily conforms to local capabilities. The ICAD/LISP-based Throttle Body DE[SM] Application readily communicates with the Microsoft Excel-based Torsion Spring Designer. As a result, the programming expertise needed at each organization matches the skills available there. And there is no loss of information through conversion between incompatible CAD software.

4.4 SUMMARY: THE RAPID DEVELOPMENT PROCESS

The Rapid Development Process is an application of the supplier-designed component development process outlined in Section 3. The demonstrated and proven process consisted of:

- The variant component attribute structure for the Throttle Return Spring.
- The customer's simplified component model developed with the Throttle Body DE[SM] Application. This component model incorporates separate Requirements Attributes and Design Attributes lists, allowing the Throttle Body engineer to choose between changing the requirements or requesting a new spring in order to reach a compliant design.

- The supplier's exhaustive component model, developed as a Microsoft Excel Workbook called the Torsion Spring Designer.
- Communication of Requirements and Design Attributes using ASCII comma-delimited email. Both the customer and supplier developed automated procedures to read and write these email files, making communication proceed very quickly.

Implementing the Rapid Development Process easily reduced product development turn-around time by an order of magnitude. Applying this approach can produce similar gains for other variant, iterative components at any level of a systems engineering decomposition. However, rapidly evolving products, products that are custom-designed, or product with complex architectures that are vulnerable to evolution are not good candidates.

The Rapid Development Process is sensitive to the local skills and capabilities of existing product development organizations. It addresses intellectual property concerns for both the suppler and customer. And it is applicable without demanding compatible software tools at the two organizations. These all make the process an excellent model for other automotive supply-chain relationships, both inside and external to Ford.

5. ORGANIZATIONAL FIT AND THE LONGEVITY OF PROCESS CHANGE

Prior sections have described a demonstrated process for integrating suppliers into Ford's DIRECT ENGINEERING℠ semi-automated product development process. The focus has been on the technical setting and illustrating a method for applying systems engineering to supplier-designed components. But this work was done in the context of a proposal to dramatically rethink the product development process: the DE℠ vision (Section 1.3). While a carefully thought out technical solution is necessary, issues of organizational fit and long-term adoption will ultimately hold more significance to the successful outcome of this project.

Introducing the DE℠ semi-automated product development process and the coordinated supplier process produced an order-of-magnitude or better reduction in design iteration turn-around time. However, the software tools that achieve this end are inherently quite complex. And both creating and successfully operating the new process demands a high degree of discipline in the software developers and the target engineering community, which are substantial shifts from current practice within these organizations. Naturally one should be concerned with the fit of these changes to their audience, as much as the fit of the technical solution with the engineering problem.

The primary organizational issues are adoption and longevity. The technical work is of little worth to Ford if its application is not accepted and sustained. Will the engineering community embrace this new approach? Over time, will it evolve with the product, or will it atrophy when enthusiasm fades and initial proponents move on?

The DIRECT ENGINEERING℠ group has been anticipating these problems and developing means to address them as they arise. There is a strong emphasis within the DE℠ management team toward creating and managing change, and every engineer on the DE℠ teams goes through training to build change management skills. The DE℠ group has developed tools such as the Knowledge Management Process expressly to support long-term adaptation in target engineering groups where applications are being developed. Thus the DE℠ group has been working toward bridging the gap between today's engineering

organization and the one that will support the DE[SM] process.

This writer would be presumptuous to assert a new set of answers to the complex problems of adoption and longevity. However, it is useful to identify characteristics of the organizational setting that bear directly on these problems. To this end, the technical and cultural setting of the Visteon-Rawsonville product development organization were studied through interviews and observations over the course of the internship. It is important to note that mid-level management at Rawsonville stepped forward and volunteered their organization to participate in the DE[SM] process, and have continued to voice their support. An assessment of the product development organization at the engineer level is presented in this section, aimed at helping the DE[SM] group understand the sources of resistance it will encounter.

This section looks at the Visteon-Rawsonville organization alone. While it is equally important that suppliers embrace this process, I have not analyzed them in this section for several reasons. I argued in Section 4.3 that the proposed process meets suppliers' business goals. Whether it fits a supplier's organization is a problem specific to that organization. And suppliers to some degree follow Visteon's lead: if Visteon embraces this process, suppliers will gradually have to as well. Finally, I worked solely with Michigan Spring during the internship, and it would be inappropriate to extend observations drawn from this single, and decidedly positive, relationship across Visteon's broad supply chain.

Section 5 is organized as follows. Section 5.1 presents a short summary of the organizational setting. Section 5.2 uses a Capabilities/ Rigidities framework to explore the organizational changes that adopting the DE[SM] process entails. Section 5.3 then looks more closely at architecture as a source of rigidity. Section 5.4 focuses on the questions of acceptance and longevity.

5.1 ORGANIZATIONAL REVIEW

Visteon's Rawsonville engineering organization is divided along major product lines. One such division is Air/Fuel Handling, comprised of roughly 100 engineers. About 30% of these engineers work exclusively on Throttle Bodies. There are two Throttle Body development groups organized under engineering supervisors. One group is focused on smaller (I-4)

engines and the European market. The other handles Throttle Bodies for larger, generally American engines (V-6s and V-8s); this is the group that the DE[SM] team has been working with. A typical Throttle Body engineer has an undergraduate engineering degree.

The history of this group deserves mention. Throttle Body development used to be based within Ford's Dearborn campus as part of Power Train Operations (PTO), while all manufacturing occurred in Rawsonville. Only a few years ago the design function was moved the 20 miles or so west to the plant, in part to enable closer collaboration between manufacturing and product engineers. All the Throttle Body engineers that had worked in the PTO organization refused to move with their jobs, so the Rawsonville department was somewhat built from scratch. The manufacturing engineers did not immediately accept their new neighbors; it has taken the intervening years for the two groups to begin to work together despite their proximity on either sides of an aisle. Some exclusivity is attributed to Product Engineers whose attitude, in the words of one engineer, is: "I design parts, my shoes are clean."

5.1.1 Functional Roles in Product Development

Following are short job descriptions for the Rawsonville product development organization.

The Product Design Engineer is responsible for three Throttle Body programs on average, and in this role works with the customer, does component design, and manages the balance of vehicle system needs against internal targets such as cost, manufacturability and production needs. Product Design Engineers coordinate all aspects of the product development organization including supplier activities. They have a stronger program management role than a technical development role. Product Design Engineers report to a Product Engineer Supervisor.

The Product Engineering Designer performs the solid-model drafting of the Throttle Body, and makes the drawings of all piece parts. In this role, designers actually do a great deal of what they call the "real design work," per engineers' direction. This demands interpreting engineers' concepts or sketches. Designers don't worry about cost issues directly. Their role brings them in contact with the product engineers, manufacturing engineers, the prototype

shop, and even the manufacturing floor. Product Engineering Designers report to a functional manager.

A typical Product Engineering Supervisor in the Throttle Body has responsibilities including supporting the individual engineers, reducing cost (both production and warranty) and improving customer satisfaction. The supervisor works with suppliers on quality, financial, and timing issues, and with customers only when higher level questions are posed.

5.2 CAPABILITIES AND RIGIDITIES AT RAWSONVILLE

Introducing the DIRECT ENGINEERING[SM] semi-automated product development process into an existing organization requires (and causes) many changes in the overall organizational setting. To identify these complex changes and assess their effects, I have chosen to use a framework proposed in "Core Capabilities and Core Rigidities: a Paradox in Managing New Product Development." [Leonard-Barton (1992)].

Product development organizations draw on sets of complementary skills, assets, and routines to create new products. This skill set, which contributes to the firm's competitive advantage, is called a Core Capability. One goal of the firm is to continually enhance this capability, thereby maintaining its strategic position. An often conflicting goal is to develop new capabilities or skills as the competitive environment changes. Leonard-Barton (1992) argues that Core Capabilities are often self-reinforcing: they enhance their own development. But core capabilities are also a source of resistance to changing or redirecting the firm, and have an equally important role as Core Rigidities.

For instance, a firm's success with a particular product line focuses attention and careers around that product line, a "virtuous circle" that continually improves this line. Yet this attention occurs at the expense of other, less successful products. To put it simply, many factors conspire to keep a firm on the path it is currently taking. Leonard-Barton (1992) calls this the "paradoxical struggle to maintain, yet renew or replace core capabilities." She identifies four inter-related dimensions to a core capability/ rigidity: Technical Systems, Managerial Systems, Skills and Knowledge Base, and Norms and Values.

I have introduced Core Capabilities/ Rigidities because I find the concept and its

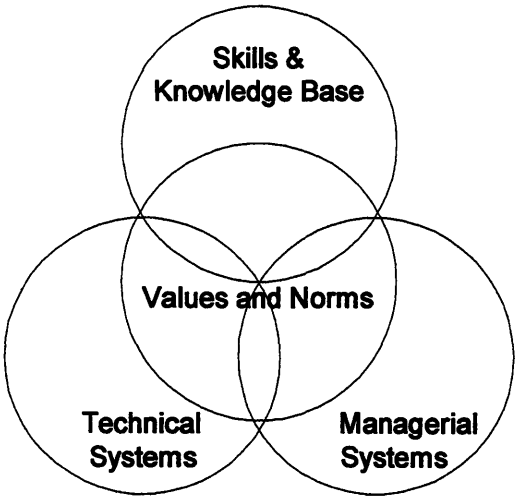

Figure 5-1 The four interrelated dimensions of a Core Capability/ Rigidity [Leonard-Barton (1992)]

corresponding dimensions to be a useful framework for assessing the DE[SM] process's influence in the Visteon- Rawsonville Throttle Body product development organization. For the DE[SM] process to attain its vision for Rawsonville, it must successfully change the way product development is performed. In the language of Leonard-Barton, the DE[SM] process must replace some of Rawsonville's Core Capabilities and doing this will inevitably conflict with some of the organization's Core Rigidities.

The four inter-related dimensions of Core Capabilities/ Rigidities are shown in Figure 5-1. Each of these dimensions both influences and is influenced by the others. They are discussed in detail in the following sections. The four dimensions vary in terms of ease of change. In the sequence presented below, Leonard-Barton (1992) asserts that each is a little less tangible, less visible and less explicitly codified than the one before.

5.2.1 Technical Systems

The Technical Systems dimension of a Core Capability/ Rigidity is comprised of information, procedures, and tools developed in the organization over time- "artifacts" left behind by talented individuals that embody their skills in a readily accessible form. Such Technical Systems contribute to core capabilities by providing advantages over competitors in timing, accuracy or amount of available detail [Leonard-Barton (1992)]. This description no doubt sounds familiar to DE[SM] process proponents: developing such Technical Systems is central to

the DIRECT ENGINEERING[SM] vision.

Technical Systems: Status and Capabilities

Prior to the DE[SM] process, only a limited amount of Technical Systems had been created by the Throttle Body product development group. They had apparently inherited some Design Handbooks from the earlier PTO organization. For example, the Throttle Body Design Guide summarized component requirements and standard engineering and design parameters. It did not provide clear design procedures or calculations. A sample component's fatigue requirement (say, 1.5 million cycles) would be noted, but material properties, how to determine loads or how to calculate stress, were not.

My impression was that Design Handbooks tended to be referred to only rarely. Existing parts were (and are) frequently used as guidelines for new parts. A Complexity Reduction effort identified a set of parts that were targets for re-use, with the expectation that designers and engineers would turn to this list first in developing new parts. And in general, as is discussed under "Employee Skills and Knowledge Base" below, there was an observed reliance on the collective memory of the organization.

Rigidities associated with Technical Systems

Technical Systems become a Core Rigidity when they are difficult or expensive to change. The skills and processes captured in software or hardware become easily outdated [Leonard-Barton (1992)]. And there is a reluctance to leave familiar tools or abandon their "sunk" cost even when faced with obsolescence. While arguably the current state of the Technical Systems observed in the Throttle Body organization makes this a dimension that is open to change, I observed a fair bit of attachment to today's development process. Throttle Body engineers are comfortable doing their jobs without the support of firm design systems and procedures.

DIRECT ENGINEERING[SM] and the Technical Systems dimension

Given the absence of strong Technical Systems today, the DE[SM] process has the attractive task of filling a relative vacuum. However, the DE[SM] process's necessary emphasis on structure and form will introduce its own kind of Technical System rigidity, because the DE[SM] tool

represents a significant investment in a particular product architecture. This won't be felt until there is a need for architectural change, as discussed in Section 5.3 below.

Interestingly, one outcome observed during the internship was the development of a new, more thorough design guide- almost a "competitive" effort. As the DE[SM] team was collecting and organizing product information into the Throttle Body DE[SM] application, product engineers in the other (European market) Throttle Body group began to extend their own design guide.

5.2.2 Managerial Systems

The Managerial Systems dimension of a Core Capability/ Rigidity represents the formal and informal ways of creating and disseminating knowledge (through job tracks or career progression) and controlling knowledge (through incentive systems or reporting structures). Managerial Systems that incorporate unusual blends of skills or foster beneficial behaviors contribute to core capabilities [Leonard-Barton (1992)]. This dimension is illustrated through the examples below.

Managerial Systems: Status and Capabilities

One prominent characteristic of the managerial systems in the Throttle Body organization is the rotational career model. In this model, apparently developed for the Ford College Graduate (FCG) program, engineers spend 1 ½ to 2 years in each of a succession of departments; for example, product design, then operations, then manufacturing engineering, and finally procurement. The stated goal of this process is to develop a well-rounded understanding of the company. But, as one engineer told me, "it is understood that people that don't have their boxes checked will not advance." The more promising engineers- those who've been successful at each rotational step- are tapped to advance to a supervisory role, where the rotation process begins anew. The most avid rotators are the recently recruited FCGs who are anxious to begin the first steps toward a management career; many other engineers follow this career pattern as well.

The result is high turnover and limited average stays in each functional area. I observed about

30% turnover/year on average in the Air/Fuel Handling departments. Clearly this has an adverse effect on the average level of in-depth product knowledge- in the engineers who rotate out and the organization they leave behind. It also interacts with the Employee Skills and Knowledge dimension below by shaping the type of learning that occurs. It must be said that not all engineers are on the rotation program- some settle into a department and become technical specialists. These are the department "old guys," a key information resource with a real impact on learning styles.

The rotational career model does not always please management. Engineers rotate away just as they become experienced and successful within one department. But managers "learned the ropes" and reached their positions through a similar rotational program. Engineering supervisors are circulating through a similar process at their level. Interestingly, no one I spoke to is particularly proud of this system, and many think it hurts competitiveness. One feeling expressed was that the organization promotes people willing to do what it takes to advance over those with core product knowledge. The group's "mile wide, inch deep" model is compared negatively to the perception of in-depth engineering expertise at competitors like Denso. Despite disparaging the system, few choose to buck it and give up the chance for advancement.

Within a given department, an employee's performance is measured against stated yearly objectives. Engineers recognize that anything not on this objective list is not of particular importance- even topics that get substantial "lip service." For product development engineers, key objectives are meeting program milestones and financial performance. Financial performance is measured against figures from the Finance Department, which are hard to interpret at the level of a specific program. There is a sense that the figures "flow down" from the level of the plant manager, who is measured against the same criteria.

Rigidities associated with Managerial Systems

Managerial systems can become intractable. Career path models take on a life of their own: managers evaluate subordinates against the yardstick of their own careers. Subordinates are aware of the career paths that led to advancement and expect to follow in this path. As a result it is difficult to create a new career path when the need for a new role is recognized.

Skilled people will be understandably reluctant to apply their abilities to tasks that had formerly been undervalued or nonexistent [Leonard-Barton (1992)]. Rigidity in managerial systems perpetuates existing career patterns.

The managerial systems observed at Visteon also contribute to rigidity of employee skills and knowledge (below) and technical systems (above), in several ways. Because the rotational model demands engineers make a good impression at every turn, and engineers expect to complete an assignment in a limited time, there is a reluctance to try something new. Innovation carries the risk of slowing down a career plan. At the same time, the short stays mean that little in-depth knowledge is added by the average engineer to a given department.

DIRECT ENGINEERING℠ and the Managerial Systems dimension

The DE℠ vision addresses the transfer of skills and knowledge both to and from engineers as they rotate through the product engineering department. The DE℠ vision proposes that accumulated product knowledge can be embedded in a Technical System where it is readily learned, rapidly applied to projects, and continually maintained and extended. But in doing this, the DIRECT ENGINEERING℠ process requires a shift in functional roles, which is understood and promoted by the DE℠ management team. In the DE℠ vision, product engineers no longer spend most of their time repetitively collecting and applying the organization's knowledge to each new Throttle Body. Using the DE℠ application, they complete these tasks in days or hours instead of weeks. The engineers' freed-up time is used to develop new product technology and then codify this knowledge for the DE℠ application. A software developer who translates this knowledge into the DE℠ code is also added to the organization.

Arguably the outcome of these changing roles will be better product knowledge and better products developed much more rapidly than before. But while these new functional roles are recognized and embraced within the DE℠ organization itself, there are no precedents or existing career paths for these roles in the target departments where most engineers work and hope to advance. This is a clear conflict with Managerial Systems rigidity.

The DE℠ process, like any change of such broad scope, is also challenged by the existing

measurement system. Engineers report that supervisors are more concerned with meeting targets than supporting a program like the DE[SM] process because they know at the end of the day they are measured on timing and financial performance. The message from management is, "the (Throttle Body DE[SM] Application) may be a good tool, but don't make me look bad." As a result, time spent toward the DE[SM] project is lumped into the "10% miscellaneous" time slot. In this culture, the DE[SM] process will make the department's objective list only when senior management demonstrates genuine support through written, measured objectives.

5.2.3 Skills and Knowledge Base

The Skills and Knowledge Base dimension of a Core Capability/ Rigidity is simply what the employees know. Included is specific product understanding-which may not be officially recognized or documented- and the organization's collective memory. This dimension of capabilities/ rigidities interacts strongly with the managerial systems dimension because the latter shapes experiences that contribute to employee knowledge, and suggest the types of knowledge most valued in the organization (the "mile wide, inch deep" model).

The Skills and Knowledge Base dimension has a significant tacit component. By tacit knowledge is meant what MIT's Eric von Hippel calls "the things that people know but don't know they know." Some knowledge is encoded in explicit terms, while some is not- it is tacit. Often the skills and expertise extensively employed in problem solving are of the latter sort, as individuals obey sets of rules which are not known as such to the person following them. Even in modern industries indefinable knowledge is still an essential part of technology [von Hippel (1994)].

Skills and Knowledge Base: Status and Capabilities

The skills and knowledge base of the engineers and designers reflect the technical systems and managerial systems environments discussed above. Many engineers have a breadth of skills and functional understanding, to the neglect of depth in any one area. Their knowledge is built over many short department tenures and they are continually learning new roles and responsibilities. The environment demands and expects good on-the-job learning skills.

During my discussions with this engineering group, questions about learning styles revealed a strong preference for face-to-face learning via conversations with colleagues, often with the department "old guy" (the resident technical specialist). A typical response was, "I'd rather talk with (him) than study a manual or design guide." And there was even less appreciation- and some genuine derision- for "learning from a computer."

This could be seen as more a critique of today's Technical Systems than a rigid learning preference. The Throttle Body DE[SM] Application may well change these attitudes if it is user-friendly and readily supports learning. Nevertheless, I felt the department's views demonstrate a reliance on tacit knowledge. Rather than turn to a dry collection of recorded information, engineers would rather tap into the comparatively rich set of experiences of the "old guys." These experienced engineers could recognize problems and identify solutions intuitively and much more quickly than consulting a design guide- and provide an interesting anecdote or two to boot. When department engineers were asked to compare Design Handbooks to the department's technical specialists, the Handbooks were regarded as a possibly useful reference, whereas the technical specialists were seen as reliable source of practical advice for making the best decision.

A second repository of tacit knowledge was recognized in the designers (CAD drafting personnel). Designers aren't likely to rotate because their product knowledge is narrow and focused, and they remain in the same role for long periods. Due to this durability, they are important contributors to organizational memory. As one designer put it, "I spend a lot of time educating the engineers, especially the new ones." Again, interviews and observations confirmed that a great deal of this knowledge was experiential- in the designers' heads. And there was value placed on the senior designers who "have been around for a while and seem to know a lot about a lot." Designers, like their counterparts in engineering, relied on colleagues for much of their learning.

Rigidities associated with Skills and Knowledge Base

The skills and knowledge dimension is not very amenable to change because skills are built over time and many remain tacit- uncodified and in employee's heads [Leonard-Barton (1992)]. If you can't see it or measure it, how can it be changed? Also, Skills and Knowledge

Base rigidities arise because focusing on a particular knowledge area becomes self-perpetuating. The organization talks and listens in the language of mechanical engineering, impairing the development of skills in less recognized areas such as computer programming.

DIRECT ENGINEERING℠ and the Skills and Knowledge Base dimension

The DE℠ vision aims to enhance engineers' personal skills and knowledge by making organizational knowledge easier to access. By establishing the department's collective Throttle Body knowledge base, new efforts should add to that knowledge rather than reinvent it. These are both valuable goals.

A fundamental step of the DE℠ process is collecting and encoding the knowledge that employees have developed over time. But the tacit information relied on in this organization can be difficult to collect and transfer for use in a new location- it's "sticky." Information stickiness involves not only the nature of the information itself, but the amount of information that must be transferred, and attributes of the seekers and providers of information [von Hippel (1994)].

Such stickiness manifests itself in several ways for the DE℠ process's goal of collecting everything the department knows. For instance, the Throttle Body DE℠ Application team lacks a deep technical understanding of Throttle Bodies, and must rely on those who hold knowledge to help them. But there is a great deal of this sort of information, much of it conveyed through anecdotes or the nonverbal, integrative skills of the technical specialists and designers. Since the DE℠ team can't know in advance which subset of this vast array will be most relevant, it is very difficult to know what to record. Over time this disadvantage should decline as the Throttle Body DE℠ Application matures and knowledge gaps are filled in.

5.2.4 Values and Norms

The Values and Norms dimension of a Core Capability/ Rigidity identifies what knowledge is valuable and who controls it [Leonard-Barton (1992)]. The values assigned to knowledge creation and content, constantly reinforced by corporate leaders and embedded in management practices, affect all the development projects in a line of business. The very same

values, norms and attitudes that support a core capability and thus enable its development can also constrain it.

Values and Norms: Status and Capabilities

My observations of the Throttle Body development organization noted low possessive ownership of knowledge, a certain unwillingness to codify knowledge, and a high value placed on creative freedom leading to broad product variety.

Possessive ownership of knowledge or projects was not observed, even among the technical specialists. Given the rotational nature of careers, this is hardly surprising. No one I spoke to had conducted a Throttle Body project from its beginning; engineers pick up work from someone who is rotating away. In this environment the culture must be open for projects to survive. And by being open with their knowledge, the technical specialists enjoy the respect of their more junior peers- who, given their short tenures, are unlikely to threaten the job security of these "old guys." Everyone interviewed, engineers and designers alike, felt that generally people throughout the organization are very open with their knowledge.

Related to such knowledge openness was a certain unwillingness on the part of engineers to codify knowledge- or to establish norms for their peers. In the course of my internship I did not see the department settle on a Throttle Return Spring "variant component attribute structure," required for the technical part of this thesis. The engineers agreed it was a good idea, admired how the preliminary attribute structure (which I created) worked in the Rapid Development Process, expected the supplier to contribute to this attribute structure, but never recorded an "official" structure. Always cited were concerns that not all the issues had been addressed. My impression was that under this reluctance to catalog the Return Spring was an unspoken reluctance to accept the responsibility of creating the "rules" by which all Throttle Bodies might be designed.

I also surmised that this reluctance to codify knowledge was related to the absence of a top-level strategy for the content and evolution of the Throttle Body organization's product line. Confusion was expressed about this at even the supervisor level. The lack of strategy might be attributed to an "emergent" philosophy- but that hadn't been identified either, and it wasn't

apparent from where new Throttle Body technologies would emerge. Instead, the lack of a defined strategy became an excuse: if you don't know where the product is headed, what's the point of codifying what you know? You might record things that would soon be obsolete.

Another value observed throughout the organization, not uncommon to engineering cultures, was the high importance placed on having sufficient creative freedom to precisely meet customer demands. At Rawsonville this value can partly be attributed to past relationships. Not many years ago, the customer- which was always Ford- did all the design work and called all the shots, and the Rawsonville plant made whatever was requested of it. While the design function has moved to Rawsonville, and components plants throughout Ford have been joined under the Visteon banner, the earlier mindset remains (and is probably reinforced by customer attitudes). As the new structure shifts focus towards cost and profitability, Rawsonville may start to "push back" at the customer.

But until these new values have filtered through the organization, engineers and designers alike look at the existing product line, see vast complexity, and feel that such will be necessary for all future Throttle Bodies. As one engineer saw it, "There is always something on a Throttle Body that demands a unique solution. Maybe there is need for a special bracket, or something interfering on the engine. For whatever reasons, Throttle Bodies are always different, due to issues we don't control."

Rigidities associated with Values and Norms

The value embedded in a core culture is the dimension least able to change; values are closely bound to culture, and culture is hard to alter in the short term. Values and Norms tend to reinforce the status quo- the people in the organization place low value on what outsiders know or do [Leonard-Barton (1992)]. One interviewee observed that traditions are strong at Rawsonville, noting that "It's always been done that way" is a common response to change.

DIRECT ENGINEERING℠ and the Values and Norms dimension

The high value Throttle Body engineers place on product flexibility impacts the DE℠ effort in several ways. Engineers see the remarkable variety in the current product line and decide that such variety is an outcome of customer needs. There is a greater focus on what is different,

not what is the same, from one part to the next, particularly for Lever/Cams and Throttle Body Housings. The widely held belief is that such variety results from genuine technical needs rather than historical product development practice. Almost universally, engineers felt that the DE[SM] process would be unable to support this needed creative freedom and product flexibility.

For example, the team assigned to collect design rules for adding the Lever/Cam to the Throttle Body DE[SM] Application decided this was an impossible task. This isn't unreasonable, if one wanted to replicate the variety observed in existing Lever/Cams using one set of design rules. But a functional decomposition of the Lever/Cam suggests that there are only a handful of functional requirements, and the design rules could be quite straightforward. This would result in fully compliant Lever/Cams with a "family" appearance. The important question, which the team didn't ask, is: how much creative freedom is really needed? Today, the answer is "quite a lot."

To be accepted in this cultural setting, the DE[SM] process must gradually overcome the total-flexibility mindset- a challenging task. One method for reaching this goal is to provide tools that increase the incentive and authority to influence the customer, most notably by supporting cost calculations. Better costing information for custom design requests would provide constructive counters to customer demands. The DE[SM] process does propose to develop more and better costing information- but this goal has not been reached for the Throttle Body DE[SM] Application. In the meantime, many of the criticisms Throttle Body engineers level at the DE[SM] process are associated with its inability to support "complete creative freedom," a Values and Norms rigidity whose merit is not known, just assumed.

5.2.5 Summary of Capabilities/ Rigidities and DIRECT ENGINEERING[SM]

The severity of the challenge facing those who would alter or develop a core capability depends on the number and types of the four dimensions that are mis-aligned with the new effort. For instance, new Technical Systems (such as the DE[SM] process) must be accompanied by new skills- to both create and apply these systems- and new values- so that these systems are adopted and used. But the new skills will atrophy or leave if the managerial systems are

incompatible. New values will not take root if associated behaviors are not rewarded. [Leonard-Barton (1992)]. Successful change demands an appreciation of the multi-dimensional nature of core Capabilities/ Rigidities.

The DIRECT ENGINEERING^{SM} process is most strongly identified with the Technical Systems dimension of the organization's core Capabilities/ Rigidities. This is readily explained: it was the functionality of burgeoning Computer-Aided Engineering technologies that both lit and continue to fuel the DE^{SM} vision. But of the four dimensions in the Capabilities/ Rigidities framework, Technical Systems are most readily modified. Unfortunately, the DE^{SM} process faces some misalignment on all the dimensions; the challenges facing it are significant. The DE^{SM} organization recognizes and has been addressing these challenges; it is this writer's intention that the preceding analysis helps to clarify some of the conflicts the DE^{SM} process will encounter.

5.3 ARCHITECTURAL EVOLUTION VS. ARCHITECTURAL RIGIDITY

As presented in Sections 2 and 3 of this thesis, the Throttle Body and Throttle Return Spring are elements of a large, complex system. The systems engineering process was proposed in earlier sections as an excellent tool for designing this system. In this process, the overall system is decomposed into many levels, with each level being separated into individual elements which are then decomposed again. The architecture of this system is the manner in which the many elements are related to one another and to the overall functions of the system.

Each of these elements is designed somewhere in a large and complex organization which encompasses the entire vehicle supply chain. While the organization for a single element at one level of decomposition- the Throttle Body- has been discussed above, each level and every element of the overall vehicle system decomposition is designed by an engineering group within Ford, Visteon or suppliers. In theory, systems engineering leads to product decomposition based on purely technical rationale. But in existing organizations the decomposition is established by organizational boundaries and existing relationships. Architectural knowledge becomes embedded in the structure and procedures of established organizations [Henderson and Clark (1990)]. Therefore, the decomposition derived from

current organizational relationships for practical purposes defines the current state of the architecture.

Earlier I argued that technology, components and component functions all evolve, so the system architecture will evolve as well. This demands that the organization which designs the system evolve: some engineer groups (or suppliers) disappear, new groups develop, and the relationships between the many groups are rebuilt around a new architecture. An example used earlier was the advent of fuel injectors and obsolescence of the carburetor: fuel and air metering once occurred in one location but now are physically quite distinct. The speed with which existing organizations react to technological and architectural evolution is an outcome of what I call the organization's "architectural rigidity."

Architectural rigidity has implications both for the entire product development organization (the supply chain) and within an individual engineering group. With respect to the former, the process described in this thesis appears to more tightly link the supply chain- and therefore increase its architectural rigidity. Because the Rapid Development Process demands close coordination with a supplier to define a variant component attribute structure, it limits competition unless several suppliers agree to use the same variant structure - or the customer decides to give the structure to competitive suppliers. While this may hurt relations with the first supplier, it is easily done given the simplicity of the communication process. More importantly, the Rapid Development Process inhibits evolution of the product, as changes to the variant attribute structure must be coordinated between all the organizations who use it. The DE[SM] process and the Rapid Development Process deliver fast supplied-component design and reinforce long-term relationships, but increase the product's architectural rigidity.

Architectural rigidity is also a factor within the Throttle Body engineering group. As Section 5.2 above showed, this group currently develops Throttle Bodies by relying on organizational memory, tacit knowledge, and to a small degree, Design Handbooks. In this environment, evolution of product or architecture occurs in an unstructured, exploratory process where who sits next to who is as important as raw technical knowledge. Inefficiencies such as reinvention no doubt result. One could posit that this organization absorbs technological evolution somewhat unevenly. After a period where competing ideas would struggle for preeminence,

an improved architecture embodying the new technology would emerge. Disseminating this new architecture would take some time.

The DIRECT ENGINEERING℠ vision proposes to change all that by introducing a complex semi-automatic product development environment that embodies the product's architecture (and more) to develop new Throttle Bodies. With the DE℠ process, architectural evolution would occur much differently. Engineers assigned to this as-yet undeveloped role would identify new technologies and develop the corresponding "best" architecture. Their solution would be embedded in the Throttle Body DE℠ Application through a large-scale re-writing of the underlying code. After testing, the new application embodying the new architecture would be published to the Throttle Body group.

Because such large scale rewriting will take a long time and not occur very often, the DE℠ process adds a new sort of rigidity to the product development environment. This rigidity in the face of architectural change is an important concern. Relevant questions include: is the proposed process more or less rigid than what is used today? How much current knowledge is lost when the architecture changes? And, is the design speed payoff so great that the cost of increased architectural rigidity is a small price to pay?

Certainly the DE℠ process will be more rigid than the existing process- any tool is surely less flexible than the absence of one. How much more so is really unknown, as no DE℠ systems have to-date absorbed an architectural change. Within the application itself, it is vital that the methodology chosen for capturing and recording component information simplifies reuse as the surrounding assemblies evolve. The process described throughout this thesis is no doubt satisfactory in this regard for Throttle Return Springs. Can the same be said for more complex components or assemblies? It is hard to say. This section offers more questions than answers. But architectural evolution is a real and inevitable phenomenon that the DE℠ process will have to accommodate.

5.4 ACCEPTANCE AND LONGEVITY

This section presents short analyses of two fundamental problems facing all DIRECT ENGINEERING℠ applications including the Throttle Body group. These are acceptance of

innovation and software complexity. The question of longevity is then reconsidered.

5.4.1 Resistance to Innovation

To be ultimately successful, the DE[SM] semi-automatic development process has to be embraced by the organization and displace traditional Throttle Body development routines. At the same time it will change traditional development roles. To do these things the DE[SM] process must overcome an expected natural resistance to innovation- a "blind reaction to technological change"- that will attempt to fight it off [Morison (1966)]. Thus the coming months at Rawsonville will be a study in innovation and acceptance.

Innovations like the DE[SM] process are often opposed by three considerations: honest disbelief in dramatic but unsubstantiated claims of the new process, protection of the existing devices and instruments with which engineers identify themselves, and maintenance of the existing society with which they are identified [Morison (1966)]. The first consideration, disbelief, was observed repeatedly among peers and supervisors in Rawsonville, all of whom listened and accepted the DE[SM] teams' assertions but never seemed to really believe. The second consideration is simply the Technical Systems rigidity discussed above.

The third consideration to resisting innovation is maintenance of the existing society, which stems from a normal human instinct to protect oneself, and more especially, one's way of life. Just like the military man who intuitively feels that "a change in weapon portends a change in the arrangements of his society" [Morison (1966)], the engineers at Rawsonville will see the DE[SM] process as a threat to their accustomed activities. Such fears, if unreasonable, are not unfounded. The engineers and managers in the DE[SM] organization expect (and indeed intend) the DE[SM] process to bring about a fundamental change in development roles. In some ways the DE[SM] process is to product development engineering as automation is to manufacturing. As automation enters a factory, less workers' time is spent in production and more is spent tending the new robots. Similarly, the DE[SM] process intends to so expedite the product development process that engineers' focus will shift to extending product knowledge and updating the DE[SM] application.

An alternative response to the increased speed of product development using the DE[SM]

process might be a reduction in the number of engineers- a strategy certain to sharpen resistance to this innovation. However, this is neither a goal nor a perceived threat. The DIRECT ENGINEERING[SM] organization emphatically states that newly freed-up time must be used to improve product technology. And designers and engineers in Rawsonville unanimously expressed doubt that the DE[SM] process would free up any department time at all. They felt that every earlier product development enhancement introduced to purportedly save time had always left department person-hours unchanged or increased, and the DE[SM] process's claims were interpreted from this perspective.

The developers of DIRECT ENGINEERING[SM] tools see their product in a fundamentally different light than their audience. Like any creator identifying with its creation, they obtain a satisfaction from the successes of DE[SM] applications which prevents them from thinking too closely about either their use or defects [Morison (1966)]. And many discussions with the DE[SM] organization suggests their technological choice represents a vehicle for the expression and enactment of the worldviews of DE[SM] advocates and designers [Thomas (1994)]. Many proponents fo the DE[SM] process view it as applying exciting technology to an obvious problem (in their eyes), the static, low efficiency of Ford's product development organizations. But the targets for all this work and innovation- such as the Rawsonville Throttle Body engineers- probably identify themselves with the "way of life" they have inherited or accepted with little modification and aren't eager to change [Morison (1966)]. The DE[SM] team is developing a solution for a group with a fundamentally different worldview. They hope for enthusiastic acceptance- and expect the tool to sell itself- while the audience sees the problem differently (if at all) and certainly doesn't grasp the DE[SM] team's solution. The writer also found much of the DE[SM] vision to be quite compelling, but believes that the DE[SM] team faces an uphill battle in any product development environment.

5.4.2 A complex process vs. a complex software tool

Today, Throttle Body design is a complex process that is not well documented- it might be called an "unrecoverable organization" because no one has ever written down how it works. A proposed solution to this uncertainty is the DIRECT ENGINEERING[SM] process which leads to a complex program that is not well documented- what might be called an

"unrecoverable program." From observation, a DE[SM] application rapidly grows in size (developers spoke proudly of their code's line count exceeding 30,000). At the same time, the language used is pretty arcane and documentation is light or nonexistent. Thus one outcome I observed is hardly surprising: the application team members gradually began to lose the ability to understand either the program's "big picture" or each other's work without help. When the developers have difficulty navigating their program only months since its writing, there are clear concerns for long term software evolution. And given the high turnover observed throughout the organization, this is doubly worrisome.

This is a real problem that needs to be addressed. During development, the DE[SM] application teams are under pressure (both actual and perceived) to show results. Rather than spending a long time documenting product architectures and knowledge while developing an overall software strategy, teams are tempted to plunge onward. They create remarkable reams of code, demonstrate early results, but ultimately hurt the long-term effectiveness of their DE[SM] effort.

The DIRECT ENGINEERING[SM] process shouldn't simply replace a complex human system with a complex software system. If not carefully planned and documented, the solution is as obtuse as the process it is trying to capture. As a result, editing, extending or replicating the software will not be easy: And (for instance) changing the throttle body architecture will be very difficult. A vital first step for each DE[SM] team is to identify and record the systems engineering structure for each application, be it Throttle Bodies or any other vehicle sub-system. Within this structure (or architecture), each component, interaction, and interface can be identified and described. Finally, only after these important steps, should the software be written.

5.4.3 Fit and Longevity

The DIRECT ENGINEERING[SM] organization recognizes the far-reaching change that their proposed process entails. They are aware that this will be met with resistance. In this section, I have tried to identify and catalog much of that resistance. The DE[SM] process raises conflicts with many types of rigidity, both organizationally and technically. These rigidities will resist its

introduction, even if not openly, but the DE[SM] process's most important opponent is perhaps the human resistance to innovation. Given its ultimate acceptance, the DE[SM] process's worst enemy will be its own complexity and the resulting reinforcement of architectural rigidity. If not anticipated and managed, this complexity could so impair the ability to adapt to change that it could lead to the eventual abandonment of DE[SM] applications. It is this writer's belief that a concerted effort to address these concerns can successfully make the DE[SM] process a long-term solution.

6. CONCLUSIONS AND FUTURE RESEARCH

This thesis presented a successfully demonstrated method for implementing rapid, iterative product development between an automotive customer, Visteon-Rawsonville, and a component supplier, Michigan Spring. The demonstrated process integrated supplied component design into assembly design in Ford's DIRECT ENGINEERING^{SM} semi-automatic product development process. A direct outcome of implementing the process was an order-of-magnitude improvement in supplied-component design iteration speed. As a result, the DE^{SM} vision for rapid product development was realized with a significant component being designed outside the immediate organization and its DE^{SM} application.

Throughout the thesis, it was argued that systems engineering is at the core of the DE^{SM} process as well as the enhanced supplier process. Systems engineering demands a careful decomposition of a product system into sub-systems and components, each with well-defined interfaces and requirements. When decomposition is done properly, a supplied component's specification can be clearly defined. But systems engineering skills aren't simply needed to develop the process described in this thesis- Fine and Whitney (1996) assert that the skills of decomposition and preparation of specifications comprise basic strategic skills for any product development organization.

The process developed in the thesis was shown to apply to a specific class of supplied components: variant components designed through iterative processes. Variant components are unique variations drawn from an existing, defined set of design options. Iterative design processes involve the repeated modification and communication of product requirements and product design. A large fraction of the supplier-designed parts in an automobile fit the variant, iterative classification.

Systems engineering and the use of variant parts were then combined by introducing the concept of a variant component attribute structure. This attribute structure codifies the supplied component's architecture, available design options, and component-to-surroundings interfaces. It is developed jointly by the customer and supplier and is shared by both parties. The attribute structure becomes a "background specification," and communicating

requirements and design during product development entails the simple exchange of attribute values. For the demonstrated process, these attributes were exchanged using text emails.

A theme revisited numerous times throughout the thesis was the certainty of eventual product evolution and the need to anticipate and manage it. Thus it was argued that the DIRECT ENGINEERING^{SM} process, which embodies the product's architecture at the time an application is written, must be designed to accommodate architectural evolution. Currently, Assembly Knowledge- the system model which captures how components interact with one another- is embodied in the DE^{SM} application software. But the software tools available to developers lead to large, cumbersome programs which become difficult to navigate and edit just as they reach a useful level of functionality. Therefore, it was proposed that Assembly Knowledge must be deliberately captured, maintained and updated by the DE^{SM} process. This is a complex problem, and one not solved in this thesis, but ignoring it will lead to architectural rigidity in the future as incorporating product changes becomes increasingly difficult.

Finally, an assessment of the organizational challenges facing the adoption and longevity of the demonstrated process was presented. In short, the DE^{SM} process, and the supplied-component process, requires an environment that differs on many dimensions from the product development environment observed today. As a result, organizational resistance to change and innovation presents sizable barriers to the DE^{SM} process. The DE^{SM} management team is aware of these challenges and is taking steps to address them- and realizes they face an uphill battle. The DE^{SM} management team believes strongly in their idea- it represents "a step out in front for Ford- a company who traditionally follows others' leads."

This writer agrees with DIRECT ENGINEERING^{SM} process proponents that today's product development environment, as observed, lacks discipline and is unnecessarily inefficient- and needs to adopt a coherent design strategy such as systems engineering. While taking this step doesn't in principle require a software-based approach like the DE^{SM} process, it is unlikely that any such change will endure without it: it is probably harder to get these engineers to become disciplined users of design standards than reluctant embracers of technology. Thus do design strategies (systems engineering) and design tools (the DIRECT ENGINEERING^{SM} process) become inextricably entwined. If this is the case, the key

challenges for the DE℠ organization are two: first to devise a plan for future applications to incorporate systems engineering and support architectural evolution, and second to learn how to overcome the multifaceted organization resistance to this dramatic change in Ford's product development process.

6.1 DIRECTIONS OF FUTURE RESEARCH

The demonstrated process for integrating supplier-designed components into vehicle design greatly enhanced design iteration speed. But it even more dramatically reduced the amount of time required to produce and evaluate a new design. This is disguised by the measurements in Table 4-1, which shows total elapsed time, a number that includes a fair degree of latent time. The supplier engineer actually spent perhaps 10 minutes at most for each iteration- a remarkable improvement in process speed.

At the same time, a comprehensive system model of the spring (in the Torsion Spring Designer at Michigan Spring) and its location in and interaction with the Throttle Body system model (in the Throttle Body DE℠ Application) were both created. And the transfer of information between these systems models was automated as well, reducing to nearly zero the time required to introduce and evaluate a new spring design.

Taken together, these two developments could challenge traditional views of the product development process by making new approaches to both optimization and total vehicle integration possible.

Car-sized systems engineering tasks always have optimization challenges: after all the decomposition has occurred, is it difficult to optimize the design as a whole before re-integrating. At the same time, it has always been difficult to manage all the information for a single design, let alone multiple designs. As a result, optimizing vehicles is commonly done through a point-based approach in which the design process starts at a single point in the "design space," defined by an initial estimate of top-level requirements. As these requirements change and bottom-up designs are evaluated, this point in the "design space" is repeatedly iterated until it is concluded to be optimal [Ward et al (1995)]. The number of iterations possible within the time available has always limited this process. And design spaces tend to

be discontinuous and non-linear, so its hard to know when an optimum is reached.

By embodying the product's system model in code, automating the transfer of information up and down the system decomposition, and significantly reducing the time needed to create a new design, the process developed in this thesis makes it relatively easy for the supplier and customer to support a collection of design alternatives simultaneously. This can support a new product development approach: set-based design. In this process, the requirements begin as a range rather than a single point. A design space- a set of designs- are evaluated in parallel and gradually narrowed as the requirements converge and the product is optimized. To maximize product flexibility, final design decisions are delayed and the design space is kept open as long as possible. The end design benefits from rational exploration of the design space and is more likely to approach a globally optimal design [Ward et al (1995)].

The complexity and information needed to conduct this process renders it impractical for standard development methodologies. But it appears quite reasonable that DE[SM] applications at both the supplier and customer could be designed to accommodate requirements and design ranges, enabling the set-based approach to be utilized. The unique setting provided by the DE[SM] process enables an exploration of the benefits of set-based design.

A second challenge to traditional product development that is based on the DIRECT ENGINEERING[SM] process and the work demonstrated in this thesis is a new approach to total vehicle integration, one that borrows from processes developed in the software industry. This is the "synch-and-stabilize" process for software development. Core to this process are frequent synchronization steps where all contributors submit the current state of their components, and periodic stabilization points where the overall architecture of the product is incrementally defined [Cusumano and Selby (1996)]. This process has heretofore only been possible in software development, where a complete prototype can be assembled and tested daily. The introduction of a complete systems engineering process, full vehicle solid modelling, and development processes such as those demonstrated in this thesis could make a similar process feasible for cars. The result would be a dramatic shortening of development cycle time and better total product integration. Applying "synch-and-stabilize" to cars is an intriguing vision, but one that may take many years to achieve.

End Note:

DIRECT ENGINEERING[SM] and DE[SM] are Service Marks of Ford Motor Company.

7. REFERENCES

Alexander, Christopher, *Notes on the Synthesis of Form*, Harvard University Press, Cambridge, 1964.

Cronin, Mary J., "Ford's Intranet Success," *Fortune*, March 30 1998.

Cusumano, Michael A. and Richard W. Selby, "Synchronize-and-Stabilize: An Approach for Balancing Flexibility and Structure in Software Product Development," *Communications of the ACM* (forthcoming), 1996.

Cusumano, Michael A. and Takeishi, Akira, "Supplier Relations and Management: A Survey of Japanese, Japanese-Transplant, and U.S. Auto Plants," *Strategic Management Journal*, 12:563-568, 1991.

Fine, Charles H. and Whitney, Daniel E., "Is the Make-Buy Decision a Core Competence?," MIT Center for Technology, Policy, and Industrial Development, Working Paper Rev 3, 1996.

Fujimoto, Takehiro, "The Origin and Evolution of the "Black Box Parts" Practice in the Japanese Auto Industry," The University of Tokyo, Working Paper 94-F-1, 1994.

Leonard-Barton, Dorothy, "Core Capabilities and Core Rigidities: a Paradox in Managing New Product Development," *Strategic Management Journal*, 13:111-125, 1992.

McCord, Kent R. and Steven D. Eppinger, "Managing the Integration Problem in Concurrent Engineering," MIT Working Paper #3594-93-MSA, August, 1993.

Morison, Elting, "Gunfire at Sea: A Case Study of Innovation," *Men, Machines and Modern Times*, MIT Press, Cambridge, 1966.

Pimmler, Thomas U. and Eppinger, Steven D., "Integration Analysis of Product Decompositions," ASME Design Theory and Methodology Conference, Minneapolis, 1994.

Sferro, Peter R., G. Frederick Bolling, and Richard H. Crawford, "It's Time for the Omni-Engineer," *Manufacturing Engineering*, Vol. 110, No. 6, 1993.

Steward, Donald V., "The Design Structure System: A Method for Managing the Design of Complex Systems," *IEEE Transactions on Engineering Management*, Vol. EM-28, No. 3, 1981.

Suh, Nam P., *The Principles of Design*, Oxford University Press, New York, 1990.

Thomas, Robert J., *What Machines Can't Do*, University of California Press,Berkeley, 1994.

Ulrich, Karl T., "The Role of Product Architecture in the Manufacturing Firm," *Research Policy*, Vol. 24, 1995.

Ulrich, K.T., and Eppinger, S.D., *Product Design and Development*, McGraw Hill, Inc., New York, 1995.

von Hippel, Eric, " "Sticky Information" and the Locus of Problem Solving: Implications for Innovation," *Management Science*, Vol. 40, No. 4, 1994.

Walton, Mary, *Car: a Drama of the American Workplace*, W.W. Norton and Co., New York, 1997.

Ward, Allen, Jeffery K. Liker, John J. Cristiano, Durward K. Sobek II, "The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster," *Sloan Management Review*, Spring, 1995.

Womack, James P., Daniel T. Jones, and Daniel Roos, *The Machine That Changed the World*, HarperCollins, New York, 1990.

2478-21