Jidoka for Product Development of Electronic Modules for Automotive Applications

by

William Weller Phillips Jr.

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

at the

Massachusetts Institute of Technology

June 2003

Signature of Author: _____

William Weller Phillips Jr.
System Design and Management Program February 2003

Certified by _____

Prof. Nancy Leveson
Thesis Supervisor
Professor of Aeronautics and Astronautics and Engineering Systems

Accepted by: _____

Steven D. Eppinger
Co-Director, LFM/SDM
GM LFM Professor of Management Science and Engineering Systems

Accepted by: _____

Paul A. Lagace
Co-Director, LFM/SDM
Professor of Aeronautics & Astronautics and Engineering Systems

ABSTRACT

Software is playing an increasing role in the automotive industry. Software helps to control and create specific functions from mixing fuel and air to controlling brakes on slippery surfaces. Software is also involved in the creation of the vehicle attributes. Vehicle attributes are the properties of a vehicle that result from the interactions of the systems; these include such things as safety, ergonomics, fuel efficiency, emissions, ride, handling and lifecycle costs (maintenance and warranty).

Software control of systems can simplify some aspects of a design (and the design process) but make other aspects more complex. The systems that create vehicle attributes are increasingly well-organized systems characterized by increasing complexity in large part due to the increased use of software controls. In his book, Introduction to General Systems Thinking[1], the author, Gerald Weinberg describes three types of systems. The three types of systems are characterized by organization and complexity; they are organized simplicity, unorganized complexity and organized complexity. The increasing use of software controls in the auto industry creates a larger number of interfaces and more interaction between systems. This is moving automotive electronics toward organized complexity, which Weinberg describes as systems too organized to use statistical tools and too complex for complete analysis.

---

[1] Weinberg, Gerald; An Introduction to General Systems Thinking. New York: Wiley (1975).

The automotive industry already has tools and methods to deal with systems characterized by organized simplicity and unorganized complexity. Tools to manage the design of systems characterized by organized complexity must be adapted to the auto industry. The roles and scopes of systems and software engineering need to be broadened to include the full set of interdependent components and systems.

This paper addresses three main areas. Firstly, the paper describes the implications for product design due to the increased use of software controls in the vehicle. Secondly, the principle of Jidoka (error catching used to motivate continuous improvement) is extended from manufacturing into product design. Thirdly, the organizational issues in the implementation of Jidoka into the automotive design process are described and addressed.

TABLE OF CONTENTS

LIST OF FIGURES

1 INTRODUCTION

The automotive industry is undergoing a quiet revolution in the way that systems are controlled. The Auto companies (also known as Original Equipment Manufacturers (OEMs)) anticipate that they will be able to continue to do more with less by using computer-controlled systems. Automating components and providing computer control of specific functions has produced good results for the OEMs. Software has already moved beyond controlling components individually to controlling interacting systems and enabling new functions.

Software control has been utilized in systems much more complex than automobiles. This history of use provides a source of lessons learned. The history of software related accidents in power plants, chemical plants, medical equipment and aerospace has been documented in papers and books such as Safeware: System Safety and Computers[2]. Software and safety engineering have developed tools to address many of the issues raised by these accidents. These tools allow software architects and designers to link specifications, models and components and allow the design teams to catch, learn from, and correct errors earlier in the design process. Automotive Product Development (PD) can learn from the tools developed by these other industries. The application of these tools is similar to a principle already used in automotive manufacturing – the principle of Jidoka (automatic error catching). The early application of error catching practices and tools is presented later in this paper as Jidoka for Product Development.

---

[2] Leveson, Nancy; Safeware: System Safety and Computers; Addison-Wesley Publishing Co., 1995

Vehicle attributes (or emergent properties) such as safety, fuel economy, emissions, ride, handling, responsiveness and ergonomics are the result of the interactions of two or more systems. Assuring these attributes at a vehicle level has historically been accomplished by coordination between the relative few design teams that have responsibility for interacting systems. For instance, the steering and braking teams have developed methods to ensure that they work together to understand the mechanical interactions of their systems. These teams have learned over time what information they need to exchange, in what formats and when.

The problem the industry faces is that of managing complexity and assuring vehicle attributes such as safety when these attributes depend upon various software routines. Additional complexity arises when the software routines may be located in different computer modules (distributed control). Software control of systems increases the number of interactions between systems and distributed control makes the coupling between systems tighter. Increasing interactions and tighter coupling present new opportunities for errors both in the products and in the design process. This thesis discusses these new errors in the context of the automotive design process and how they may be anticipated and avoided.

1.1 Motivation

The automotive industry has developed tools and methods for meeting the traditional needs of their engineering teams. These tools and methods are being augmented with tools from the Reliability, Safety and Software Engineering disciplines. The complexity of the interacting software has already made it more difficult to manage system complexity. We in product development in the auto industry need to understand our current situation and anticipate our

future needs. We need to understand the methods being used in other disciplines and learn from them, as we continue the evolution to the software control of our vehicle systems.

The automotive industry has traditionally been a mechanically based engineering community. The vast majority of systems in the vehicle have historically been of the organized simplicity type referred to earlier. As organized complexity increases the automotive community has been adopting some of the principles and disciplines of reliability engineering, system engineering and even system safety engineering. The adoption of these disciplines has been shaped by the organizational systems in which they are used. The success of adoption of these disciplines depends upon not only the knowledge of the tools but also the roles of the engineers in their organizations and the relationship of one organization to another.

Catching errors early in the design process provides an opportunity to reduce both design costs and risks. Errors in the design of software controls can result in safety problems, recalls and/or delays in the program schedule. Many system attributes or emergent properties are subjects of government regulations; these include safety, emissions and fuel economy. Other emergent properties are subject to intense market scrutiny and thus have an effect on the ability of a company to sell a new design. These market sensitive attributes include ride, responsiveness (steering, handling, and power), NVH (noise, harshness and vibration), and ergonomics. An additional complication is that market needs and regulations are changing, which increases the need for traceability and even direct linkage between vehicle specifications and lower level specifications.

1.2 Objectives

The objective of this thesis is to define system problems associated with software proliferation in the automotive industry, to examine the strategies and tools used in this and other industries, and to define opportunities for improvements. This paper examines the challenges facing automotive vehicle design using software controls now and into the near future. The principle of Jidoka is extended from its origin in manufacturing into the product development arena. Jidoka is presented as a method to create a learning organization able to continuously improve the product development process even in an environment of change. Finally, organizational issues regarding interaction and coordination are presented to illuminate the path to successful adoption of the ideas in this thesis.

1.3 Approach and Methodology

Current and future needs of automotive module design are examined using interviews, insights from professional experience as an electrical reliability engineer in the auto industry and through a literature review. It has become obvious to me that what is well known common practice in one industry may be unknown or under utilized in another. In this case, many common practices in software systems engineering and systems safety engineering are not employed or even well known in the same way in the auto industry. The auto industry has an opportunity to learn from them and move beyond just applying the same techniques in a different industry.

Interviews were conducted with engineers and managers in various positions in the automotive electronics industry. The interviews were conducted in various locations including work

locations, the SAE congress and the Convergence Conference. The participants had between 3 and 25 years in the design of automotive electronic systems or in software development. The interviews were open-ended but began with a set of questions designed to focus the discussion.

Initial Questions:

What is your experience in automotive electronic systems design and software development?

What do you see as the primary trends in electronic module design and use in auto industry?

From your personal experience what problems have you experienced in the product development of electronic modules and systems?

What issues created a need for changes in the software late in the design process?

Professional insights have been developed through a career that began in microelectronics manufacturing and testing. My career evolved into quality then reliability and finally into system engineering. During my tenure with the automotive industry, I have had the opportunity to participate in the design process of various electronic modules. I have also participated in the analysis of customer issues and field returns. Some of the non-proprietary information from this work is presented

The literature review focused on the areas of system engineering, automotive electronics and organizational change. Some of this is presented in Chapter 2 as background information while other research is integrated into the body of the paper. Chapter 3 extends the principle of Jidoka into the product development process. In it I discuss the types of systems errors that complex

software systems may experience as well as the types of errors that a design team may make that could lead to those errors. Chapter 4 presents specific improvements to the automotive design tools that enhance the identification of errors related to the design of electronic modules. Chapter 5 presents organizational issues with adoption of the systems approach in automotive product development. The conclusions are then summarized in the final chapter.

2 CURRENT SITUATION and LITERATURE REVIEW

2.1 Introduction

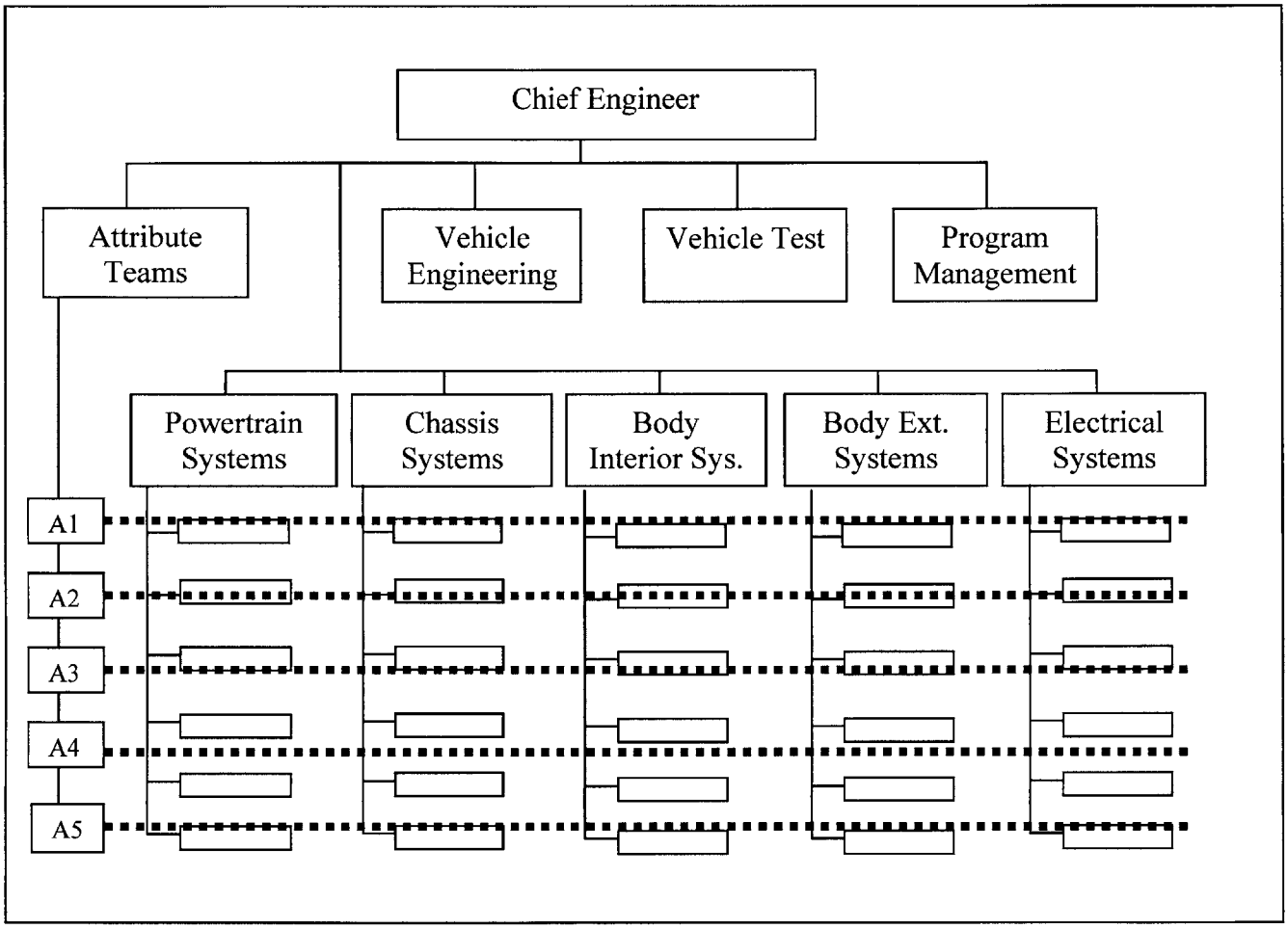
This chapter describes ideas in automotive product development and in the related disciplines of manufacturing, software product design and organizational behavior. The automotive context and ideas are presented for readers not familiar with the industry. The ideas in manufacturing and software will be integrated and extended in later chapters.

The automotive development organizations are divided into engineering teams that focus on the vehicle and those that focus on systems and components. The system and component teams do the detailed design work. A generic view of the organizational structure of the automotive design team is presented in the figure below. The organization focus of the design teams varies from company to company and over time within a company. In general the vehicle design teams are lead by the chief engineer who can be thought of as both the architect and primary manager of the vehicle development process. The design teams have primary responsibilities to their functional area and secondary responsibility to the attribute teams.

Each team determines the business case for the use of software in their system or component. The requirements flow down from the vehicle level to the system level and then to the component level. These requirements state what needs to be done and not how it is to be done. This allows each team to determine when and how to use hardware and software controls in their system.

Boxes A1 to A5 and the horizontal dotted lines in the figure below represent the attribute teams. The attribute experts may either lead attribute teams composed of members of the various design teams or act as internal consultants to the design teams. The system and component design teams have the "design and release" (D&R) responsibilities for the parts. They are also responsible to provide designs that will result in the vehicle meeting the attribute requirements. This relationship can best be understood as a matrix organization with the primary responsibilities being with the functional (vertical) design teams. The design teams usually also manage the supplier relationships

Figure 1: Organizational Structure



. The horizontal attribute teams have less direct authority. The membership in the attribute teams depends upon the coupling and interactions of the systems being designed or redesigned in the specific vehicle development program.

In automotive design, there are three important considerations: features, functions and attributes. The term "attribute" is the most abstract and general of these three terms. Safety is one of the most important vehicle attributes. In this sense, the attribute of safety has two meanings. The first meaning is the set of features that enhance a customer's perception of vehicle safety. Examples of

safety features are Antilock Braking System (ABS) and Airbags. The second sense of the vehicle attribute of safety is the sense used in system safety engineering; it is the freedom from hazardous conditions. The functions used to create the ABS feature include sensing wheel speed, sensing brake pedal force (driver's input), activating brakes and selective deactivation of the brakes.

2.2 Software Controlled Systems Design in the Automotive PD

Software controls are finding increased use in each of the traditional systems of the vehicle. Published papers and interviews show that the trend is continuing and the scope of the controls is increasing. An example of the complexity and future trend can be seen in the area of "cruise control". This feature is generally implemented in an electronic module that facilitates the driver's interactions. The electrical systems team is most likely to have the responsibility to design this module. The module will communicate with the engine and transmission controllers. These controllers have the responsibility to arbitrate between the various inputs for power and set the throttle at the appropriate position.

"Active cruise control" is a next generation feature. This feature involves changing the speed of the vehicle using cruise control to match the speed of a car ahead of the vehicle using the feature. When the system is in this mode, the transmission and engine controls need to respond differently based upon inputs from another module that decodes the radar signals. In the slightly more distant future, information about the road and traffic conditions may also play a role in this process. The intelligent highway concepts are expected to further increase the number of modes and interactions possible in this real time system. This example shows that 3 or more design

teams, each with their own embedded software, produce parts of a system that has an impact on safety, fuel efficiency, ergonomics / drivers interface and a convenience feature.

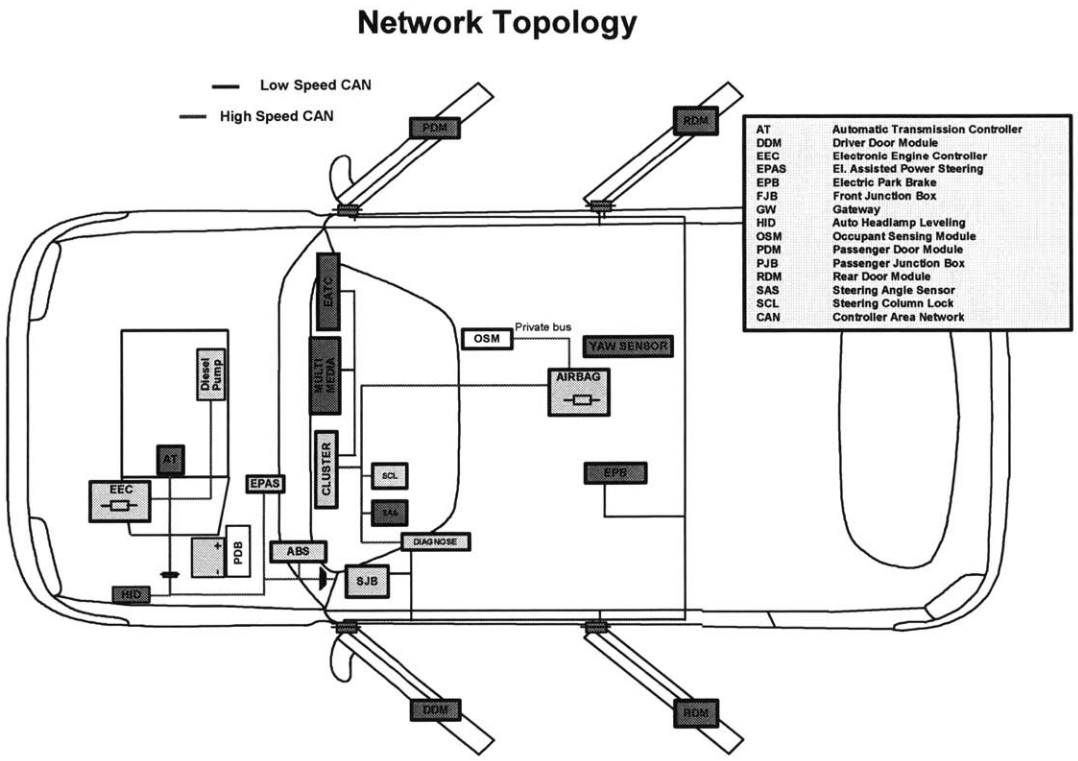
The automobile is quickly becoming an embedded network of computers. The various computers or electronic modules are connected through communication protocols. The multiple communications protocols have been developed in various ways for various purposes. The electronic modules and their interfaces are also used for diagnosis of the problems that the customers experience. Some of the communications protocols are proprietary, some are industry standards, and some are being adopted from other industries such as the newer multimedia standards.

Until recently, each electronic module had its own role that was largely independent of the other modules. Each module individually monitored and reported system states and chose the operational modes that were appropriate based upon the various inputs from the driver, sensors and the status (outputs) of other modules. Such a system falls into the category of organized simplicity. The following figure illustrates that there may be 10 to 20 "computers" or "electronic modules" in a modern vehicle. If each module can be designed independently then even this complicated looking view can be simplified by decomposition; this has been the design approach generally used in the auto industry.

There are two competing trends in automotive electronics that need to be managed: systems interactions and "smart" components. Both of these trends could move the system into the realm of organized complexity. The toolboxes of the automotive architect and systems engineer need

to include tools that allow decomposition and coordination from the vehicle level down to the component level and synthesis and coordinated action from the component level up to the vehicle level.

Figure 2: Network of Computers



2.2.1 System Engineering Design V and Systems Architecture

The "System Design V" model is common for describing the automotive development process. The following figure illustrates the "System Design V" model. The automotive PDP starts at the vehicle level. Customer wants and needs, regulations, standards, and corporate goals are defined and translated into requirements at the vehicle level. An initial architecture is developed which

includes the decomposition of the vehicle into systems and subsystems and then components. The requirements are specified for a particular vehicle at the vehicle level and then flowed down to the system, subsystem and component level. The decomposition process is iterative and allows for give-and-take in the reconciliation of specifications between systems and subsystems. The verification and validation phase is intended to be a linear process of synthesis and confirmation testing (components into subsystems and subsystems into systems and finally vehicle level testing).

The software engineering in today's automotive development process is largely viewed as a sub-set of the component engineering, but the need for a vehicle-level control architecture is growing as the control systems are becoming more computer-dependent and more interactive with the driver and other systems.

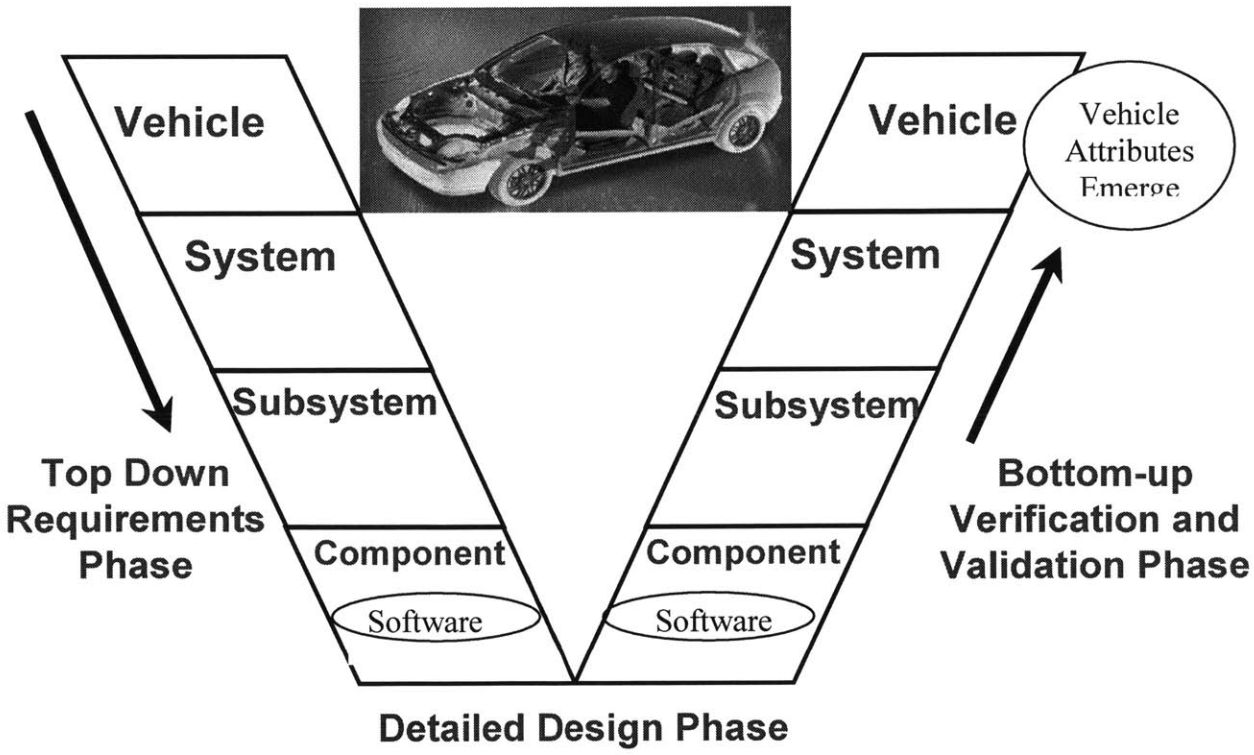


Figure 3: Current View of Software in the System Engineering Design V Model

Software control architecture at the vehicle level is increasingly important as vehicle attributes become more dependent upon the software. The emergent nature of vehicle attributes requires the architect to understand the interdependencies of the control strategies. These strategies are increasingly software control strategies. The designers and coders of the software components need to understand when these components have an impact on the vehicle level (and system level) attributes. This extended view would include a Vehicle Control Architecture role at the vehicle level and a systems software role at the system and subsystem levels.

What is meant by System Architecture? One way of looking at the role of the system architect was presented by Professor Ed Crawley. He describes the deliverables of the architect[3] in his System Architecture course at MIT. These deliverables are summarized in the following figure.

Prof. Ed Crawley's Deliverables of the Architect

- **A clear, complete, consistent and attainable set of goals (with emphasis on functional goals)**
- **•A functional description of the system, with at least two layers of decomposition Function is defined as intents plus process.**
- **•A concept for the system that includes elements of both form and function.**
- **•A design for the form of the system, with at least two layers of decomposition**
- **•A notion of the timing, operator attributes, and the implementation and operation plans**
- **•A document or process that ensures functional decomposition is followed, and the form at interfaces is controlled**

Figure 4: Prof. Crawley's Deliverables of the Architect

The functional decomposition describes which systems interact to create the attributes at the system and vehicle levels. He also recommends that the form of the interfaces be controlled at the architectural level. Architectural decisions (the form of the interface) that are made at the component level are by definition late changes or more properly late decisions.

[3] Prof. Ed Crawley Lecture on Systems Architecture ESD.34 at MIT

2.2.2 P-Diagram

Madev Phadke developed the P-Diagram or Parameter Diagram[4] in the late 1980's. The P-diagram is a method for describing systems in the design phase. The major parameters that Phadke was concerned with are: Inputs, Outputs, Primary Functions, Error States, Control Factors and Noise Factors. This System Engineering Tool is being used in the automotive industry. The tool works well for mechanical systems. The design team uses the P-diagram to focus on ensuring that the input energy is transformed efficiently into the output function. The theory of use suggests that if wasted energy is minimized then it cannot be used to create error states. This approach is very powerful for mechanical systems; a later section will expand its use to make it more relevant to software control systems.

The P-diagram performs three functions for the design teams. The first is the use of the P-diagram to help produce designs that are more efficient. The second use is to identify and prioritize the factors that are important for robust performance. A third use is to assess the completeness of the validation plan.

2.2.2.1 Background: Description of Use

The P-diagram puts the system under study in the center of the diagram. The inputs are listed to the left of the central system. The outputs or ideal functions are placed to the right of the central box and below the outputs are the error states. Above the system noise factors are described and below the control factors. The P-diagram analysis method involves listing the most critical

---

[4] Phadke, Madhev; Quality Engineering Using Robust Design, Prentice Hall PTR, 1989.

control factors and noise factors. Control factors are those design parameters that most strongly affect the primary functions of the system. Noise factors are those parameters that are not controlled by design but which affect the functionality of the system. Noise factors are generally divided into five classes or types as in the figure below.

Categories of Noise Factors in P-Diagrams

1. **Part-to-Part Variation** – List the part parameters that most affect system performance.
2. **Aging Effects**– List the changes over cycles of use or time that affect the system performance.
3. **Customer Usage Variation** – Can/will the system be used in other ways (e.g. a steering wheel used to pull a driver into the truck).
4. **Environmental Effects** – e.g. Temperatures of use, humidity, splash, cleaning fluid exposure, etc.
5. **System to System Interactions** - What interfacing systems parameters affect performance of this system?

Figure 5: P-Diagram Noise Factor Categories

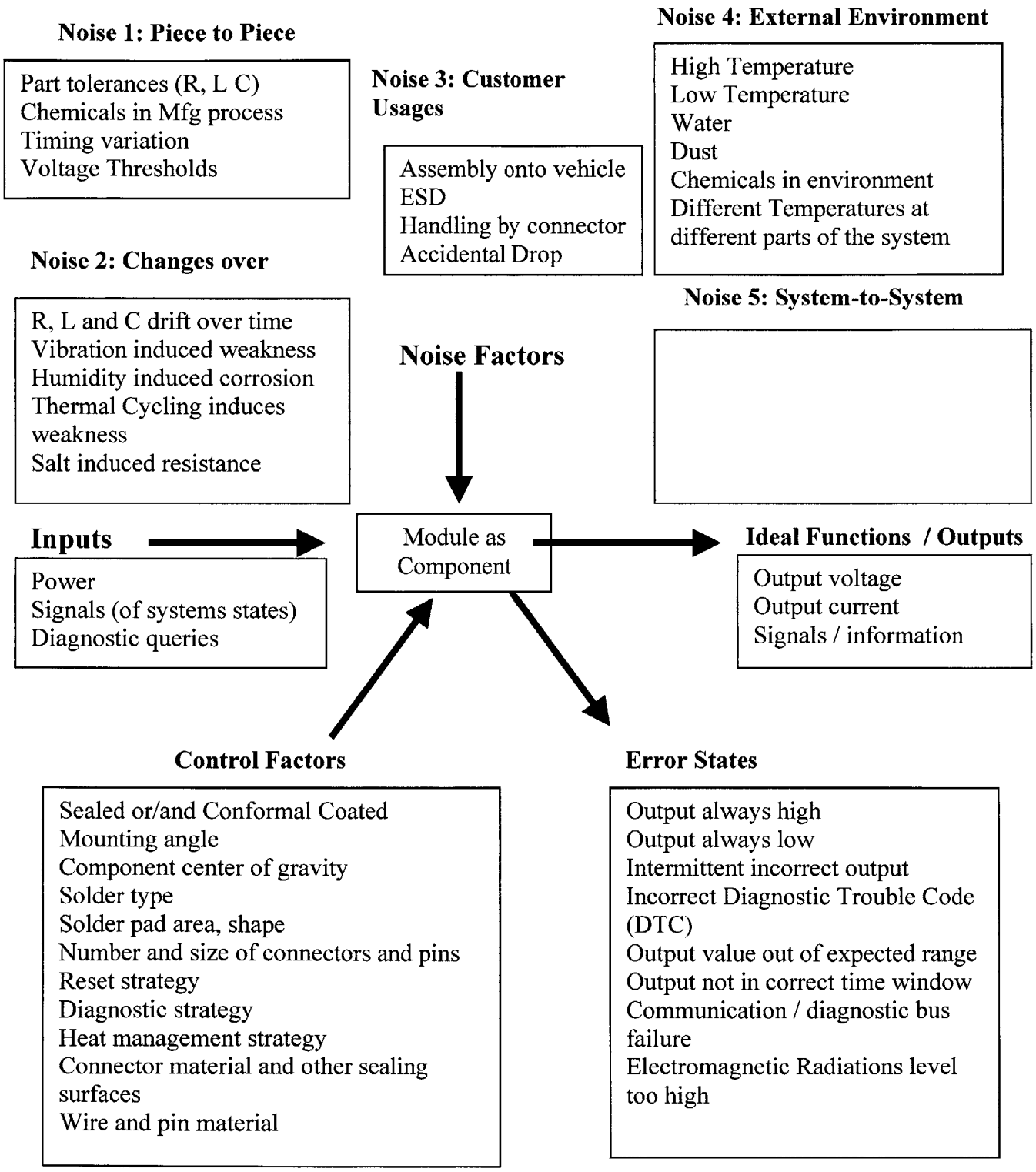
2.2.2.2 P-Diagram Applied to the Module as a Generic Component

The electronic module P-diagram describes the functions of the module generically and in simple terms. Even large modules with many scores of inputs and outputs (I/O) can be described in generic terms (from the component point of view). Inputs either supply power or information (signals). Similarly outputs are either power to an actuator or signals to another module or to the driver of the vehicle. The system functions for electronic modules are generally control and timing rather than the efficient transformation of input energy to output energy,

The following figure is a generic example of a P-Diagram applied to an electronic module. An electronic module in the automobile industry may have many dozen inputs and outputs. The diagram defines the inputs as power, signals of system states, and diagnostic queries. The outputs or ideal functions are generically listed as voltages, currents and signals.

The use of the P-Diagram to create a more efficient system is not applicable to electronic modules since their function is rarely to transform energy but rather is to provide energy or control timing. The P-diagram can be applied constructively to electronic modules since it will allow the teams to evaluate the control factors and noise factors, which can then be compared to the test plan details to improve the test coverage. System level testing can also be improved since the system-to-system interaction are described and prioritized.

Figure 6: Generic P-Diagram



2.2.3 Specifications

The parameters that impact a design and other requirements must be translated into requirements statements or specifications. The specifications of the requirements by engineering terms must be passed down the hierarchy to the suppliers of the sub-systems (and components) and passed horizontally to interacting systems at the same level of the hierarchy. Characteristics of good requirements statements are well described in the literature[5] but less well used in practice. The proliferation of software control increases the need to link the low level specifications to the attributes that they are intended to create.

2.2.3.1 Intent Specification Tool

Understanding the reasons that lay behind various design decisions is critical to successful evolution of the design. Capturing the intent behind design choices and requirements is often viewed as a lessons learned activity. Documentation of code in software has long been considered a best (and essential) practice. New tools are being developed which allow the specification process to flow into the architecture and into the code writing and documentation. One such tool is called Intent Specification.

---

[5] Ulrich and Eppinger; Product Design and Development; McGraw Hill (2000). (Ch 4 and 5)

The tool called Intent Specification was developed by Safeware Engineering Inc[6] to provide a development environment / tool for systems software engineering. The tool begins with the specification process but extends beyond it. The tool allows the linking of models and even executable code to the specifications. The Intent Specification Tool provides a framework (which can be shared between teams) that improves the definition of the specifications, provides the ability to trace specifications and provides a hierarchy for specifications.

The Intent Specification Tool developed by Safeware Engineering Inc. has seven levels of the specification hierarchy. The tool allows linkages to be established between specifications at various levels. It also allows the specifications to be linked to models of the desired behaviors, to the code-level components and even to the test methods. The logical levels used in this tool are defined in the figure below.

Specification Levels in Intent Specifications
Level 0: Program Management Information
Level 1: System-Level Goals, Requirements, and Constraints
Level 2: System Design Principles
Level 3: Blackbox Behavior
Level 4: Physical and Logical Design Representations
Level 5: Physical Implementation
Level 6: System Operations

Figure 7: Intent Specification Level Defined

The use of this tool is relatively straightforward. Existing specifications can easily be cut and pasted into the tool, which allows easy transition to the tool.

---

[6] Saferware: Systems, Safety and Computers; Nancy Leveson; Addison Wesley Publishing Co. 1995.

2.3 Jidoka in Quality

Jidoka and Just-in-Time (JIT) are commonly described as the two pillars of the Toyota Production System (TPS). They are important methods used in manufacturing to implement the principles of effectiveness and efficiency respectively. The literal definitions of Jidoka are "automation" and "automation with a human touch". These definitions come from a translation of the characters used to write the word in Japanese[7]. Jidoka can trace its origins back to the automation of weaving. The looms were redesigned to shut themselves off when they had detected that a thread had broken. The principle of detecting and correcting errors was adapted to the automotive manufacturers starting with Toyota and spreading to the other manufacturers. The techniques of Jidoka are those that make the equipment or operation stop the assembly line whenever a defect is detected[8]. Either the worker or the automated machinery may identify a defect and stop the manufacturing operation. In this way, value is not added to defective parts and the root causes of the defects can be identified earlier. In a later chapter, I shall extend this idea to the product development process.

2.4 Organizational Assessments

The effectiveness of an organization depends upon both the skills of the people in the organization as well as the structure of the organizations of which they are a part. Success in

---

[7] Harrington, Fred; Web glossary; http://www.fredharriman.com/services/glossary/jidoka.html ; 2002.

[8] Hamilton and Smith; "Implementing TQM on a Shoestring"; Journal of Management Consulting; Vol. 7, No. 4; Fall 1993.

complex product development involves choosing the right partners, and managing the relationships. The Software Engineering Institute has developed a method to assess the capability of software engineering organizations.

2.4.1 Capability Maturity Model

There are some in the auto industry that look to the defense industry for guidance and this has lead to an interest in Capability Maturity Model (CMM). The Software Engineering Institute (SEI) manages the (CMM) in a way that is analogous to the ISO 9000 standards that the auto industry is familiar with. CMM has many of the same benefits and drawbacks of the ISO and QS 9000 standards. CMM has five maturity levels and a hierarchical methodology for advancing from one level to the next. The five maturity Levels are described in the following figure. Teams with a higher maturity level are expected to be better at more complex and critical projects but the literature does not uniformly support this[9].

Five Levels of Maturity from CMM

1. Initial (chaotic, ad hoc, heroic)
2. Repeatable (project management, process discipline)
3. Defined (institutionalized)
4. Managed (quantified)
5. Optimizing (process improvement)

Figure 8: CMM Maturity Levels

[9] Bach, James; "The Immaturity of CMM"; American Programmer; September 1994.

2.4.2 Supplier Assessments in the Auto Industry

The automotive industry has a lot of experience evaluating organizations on the basis of their manufacturing quality. The automotive industry has developed standards for suppliers that it uses to evaluate suppliers. The Advance Product Quality Program (APQP) of the Automotive Industry Action Group (AIAG) is a set of expectations and procedures that OEMs expect of their suppliers. The APQP process is composed of 23 engineering disciplines performed throughout the design and manufacture of the suppliers' components. These disciplines reflect the expectations that OEMs have for their partners in the supply base. The standard also provides criteria for evaluating performance of the supplier. These expectations are summarized in brief in the figure below.

Expectations for Suppliers to the Automotive Industry

Be able to mass produce the product with low variation Cpk >1.33
Ensure Robust Systems performance by understanding how your product fits into the larger system and what happens when it fails.
Document your process, especially those that ensure a safe design.
Agree early in the design process to have competitive products at Job1.
Manage the Interfaces to your components / systems.
Meet Milestone commitments of the agreed Development Process(es).
Provide high value (functionality for the cost)

Figure 9: Expectations for Suppliers in the Auto Industry

Being flexible has multiple meanings, which include allowing for innovation between model years, allowing changes during the Design Process (deal with feature creep), be adaptive to different Regulatory Standards (existing and new markets and anticipate regulatory changes in those markets, provide products that allow Market Customization and allow for 'Series"

differentiation (feature variability). Software suppliers have an added burden in terms of these expectations. The auto industry often looks to the software to meet its needs for flexibility. This speaks to the strengths of software. The fact that software is "never cast in stone" nor is the software "die" ever cast. Cognitive psychology tells us that people tend to consider concrete concerns to be more important than abstract concerns such as design, architecture and software.

2.5 Fire Fighting and the Dynamic Structure of Organizations

Nelson Repenning ("Past the Tipping Point"[10]) and others have studied the phenomenon of Fire Fighting behaviors in large organizations. His studies have shown that fire fighting initially is used only when a project gets into "trouble" but often becomes the de facto development process. The use of fire fighting tends to replace proper process execution. Even temporary increases in workloads can push an organization into a fire fighting dynamic that is self-reinforcing and results in a permanent decline in the organizations performance.

Repenning's Systems Dynamic Model for a multiple period and multiple project organization has allowed him to evaluate the effects of various management interventions in the design process. The model shows that if there is enough time and resources to do the tasks then organizational learning can occur and a larger percentage of the up-front tasks get performed. This increases the quality of work performed down stream and keeps projects progressing toward completion efficiently. When there are more tasks or insufficient resources then a small increase

---

[10] Repenning, Goncalves, Black; Past the Tipping Point: The Persistence of Firefighting in Product Development, Sloan School of Management.

in tasks (a crisis or "fire") can cause the organizational system to neglect upfront tasks thus reducing the quality of the design work leading to more quality issues (more "fires"). He also shows that rational decisions on a local level can, when coupled with resource dependence between projects, lead to situations where the multiple projects do not get the resources that they need until they reach a higher level of severity and fire fighting becomes the norm.

Managers try to deal with fire fighting by implementing various practices that are rational and appropriate in the local context. The model assumes that tasks in the PDP are of two types: Up-front Tasks and Detailed Design Tasks. The Up-front tasks are assumed to set the stage for the Detailed Design Phase. The up-front tasks include the decomposition of the system and the allocation of the specifications from higher levels down to the component specifications. If these are done properly, then the quality of the detailed design work is higher and the need for rework is less.

2.5.1 Managerial Responses to Fire Fighting

Managers often try to improve the process in response to the fires. In System Dynamics this is referred to as responding to events (in contrast to dealing with patterns or changing the dynamic structure of the system). The fires are assumed to be associated with the specific areas where they occur rather than to the dynamic structure of the organization. Dynamic Structure includes such things as resource allocation, knowledge of the team members and the relationships between the organizations that are involved (suppliers and other resources). The "improvements" to the design process often result in degradation of the system's performance (further schedule slips for instance). There are several reasons for this to occur even when the

improvements are academically sound. The reasons include unfamiliarity of the team members with the new processes or tools that require extra time to implement. In addition, the improvements generally add tasks but do not take away any tasks. This results in a larger imbalance between work to be done and resources to do it, making it more difficult to both prevent and correct problems and further institutionalizing firefighting.

One method to address the problem of too many tasks in a product design process is to combine several tasks into one. The Intent Specification tool described above does just this. By linking intent to requirement to models to code, it provides the analysis path that is required to analyze safety and other attributes. This means that non-value added documentation tasks are reduced.

2.6 Safety and Hazard Analysis

NASA has conducted and inspired much work in the area of software and systems safety. Many lessons and recommendations have been incorporated in their Guidelines for Safety Critical Software – Analysis and Development[11]. The organizational structure that is assumed in this document is very different from what exists in the automotive industry. The automotive industry can learn from both the tools and the organizational structure.

The Safety and Hazard Analysis (SHA) method is designed to improve the safety of the system by focusing design effort where it is most needed. The method calls for categorization of the system to be designed into one of three categories. Category I is the most complex category and

[11] NASA Guidebook for Safety Critical Software - Analysis and Development; NASA-GB-1740.13-96

Category III is the least complex and least critical. The figure below describes the characteristics of the systems that are used for system categorization

System Category	Descriptions
I	Partial or total autonomous control of safety critical functions by software.
	Complex system with multiple subsystems, interacting parallel processors, or multiple interfaces.
	Some or all safety-critical functions are time critical.
II	Detects hazards, notifies human operator of need for safety actions.
	Moderately complex with few subsystems and/or a few interfaces, no parallel processing.
	Some hazard control actions may be time critical but do not exceed time needed for adequate human operator response.
III	No software control of safety critical functions; no reporting of hazards to human operator.
	Simple system with only 2-3 subsystems, limited number of interfaces.
	Not time-critical.

Figure 10: NASA Table for System Complexity

It is interesting to note that the changes in the automotive systems have been moving from Category I through Category II and now into Category III. The dimension of hazard severity is the second dimension that is used in SHA to guide design and test efforts. The four levels of Hazard used in SHA by NASA are Catastrophic, Critical, Marginal and Negligible. The descriptions of the four Hazard Severity Levels are provided in the following figure.

Catastrophic	Loss of entire system; loss of human life or permanent disability
Critical	Major system damage; severe injury or temporary disability
Marginal	Minor system damage; minor injury
Negligible	No system damage; no injury

Figure 11: Hazard Severity Levels from NASA NHB 1700.1.

The levels of oversight and coordination as well as the specifics of the design actions (in the PDP) are dependent upon the combination of the Hazard Severity Level and the System Category for NASA projects utilizing these methods.

SYSTEM CATEGORY	HAZARD SEVERITY LEVEL			
	CATASTRO-PHIC	CRITICAL	MODERATE	NEGLIGIBLE / MARGINAL
I	Full	Full	Moderate	Minimum
II	Moderate	Moderate	Moderate	Minimum
III	Minimum	Minimum	Minimum	Minimum

Figure 12: NASA Scope of Software Safety Effort

The automobile as a system becomes a Category I system when it has multiple interacting subsystems with multiple interfaces and when safety critical functions are time dependent. This would be the case for vehicles with brake-by-wire systems and steer-by-wire systems. In general, Category I applies when a vehicle design includes a computer control system that decides when to provide power for safety-critical controls based upon multiple inputs. The Hazard Severity Level analysis is very similar to the analysis used for the severity rating in Failure Modes and Effects Analysis (FMEA) in the auto industry.

The system-level hazard analysis for Category I systems with high hazard ratings dictates to NASA engineers that subsystem and software hazard analyses are also needed. It also indicates the increased importance of linking the specifications through the decomposition process and linking and communicating the results of the verification and validation testing. Software is included in the hazard analysis at every appropriate level. It is clearly a part of the component

analysis when the component includes software. Software is also included in the analysis of a system or subsystem when it is a significant portion of the strategy to create or control the system functions. The safety of the software in the system depends not only upon the software itself but also upon the system in which it is used. The reuse of software is thus no assurance of safety or of any other systems attributes that depend upon more than a single component or system.

3 JIDOKA FOR PRODUCT DEVELOPMENT

Jidoka is the enactment of a philosophy. This philosophy is a combination of several principles. The first principle is that continuous improvement is everyone's job. The second principle is that catching errors earlier is better than catching them later. The third principle is that people avoid punishment and strive for reward. The implementation of the principle of Jidoka in U.S. auto manufacturing has required a change in management style that includes rewarding people not punishing them for finding errors. Similar or analogous management changes will be needed in product development organizations.

The use of analogy can be very powerful but misuse of analogy can result if the analogy is too literally applied from one domain into another domain. In this case, I am drawing an analogy between manufacturing and product design. The level that I believe the analogy is proper and powerful is the level of the principle.

3.1 Introducing Jidoka for PD

I would like to suggest the following generalization of the principle of Jidoka: *Effectiveness of the process (manufacturing or product design) can be enhanced (continuously improved) by empowering team members to evaluate the product that they are working on, identify problems early, identify root causes and deal with them.* This principle is especially important in a time of change when core competencies in process excellence can become core rigidities that make change more difficult.

In the manufacturing domain, Jidoka allows the line workers to stop the manufacturing line when the incoming parts are bad. The local foreman is called in to confirm that a bad part is coming in and determine the source of the bad part. The source of the bad part is then analyzed and improved. In product design (PD), there is no assembly line to stop and the PD process is generally not linear; it often requires iterative interactions between design teams.

The PDP can be considered to have three phases: Requirements, Detailed Design, and Verification and Validation. The requirements phase of product design in particular is highly iterative. The detailed design phase involves largely independent component designs, making it difficult to detect system errors. The verification and validation (V&V) phase occurs later in time; the later that errors are caught the more costly they are to fix. Thus, to apply the analogy of Jidoka to PD the philosophy needs to be applied and implemented in the requirements phase. The following sections of this paper looks at what errors we are trying to avoid, how to detect them and how to coordinate action to address them as early as possible in the design process. These techniques should be formal enough to provide value immediately but also flexible enough to be improved and adapted during the transition to a PD process that is more dependent upon software to create systems functions and attributes.

3.2 Types of System Failures

Jidoka for PD is designed to minimize failures of the system that is being designed by increasing awareness of system failure modes, and by viewing the design team and process as a system that can be continuously improved. Complex computerized systems have a history of significant accidents. The increasing complexity of automotive software systems suggests that we can use

the lessons from system safety engineering and accident analysis. Accidents can be divided into two broad categories: component failure accidents and system accidents[12]. Similarly, automotive failures can have component problems or system problems as their root causes.

3.2.1 Defining System Failures

System failures are defined as the set of failures that occur when the component parts of the system all meet their specifications but the system fails to function properly. Failure to function properly includes both failures of the system to function and the failure of the system to successfully create systems or vehicle attributes.

Several types of errors in product design lead to systems errors. Unanticipated interactions between systems can be addressed through simple tools such as System Block Diagrams and P-Diagrams. Other tools such as an Interface Matrix can provide insight into more complex systems. System errors can also result from incomplete specifications. The example of the Therac-25 accident[13] highlights the risks when designers replace hardware interlocks (safety features) with software controls. Mechanical controls are characterized by physical properties that determine the likelihood that they will continue to function. These properties can be measured and controlled. The reliability of their design can thus be assured over time (to a predictable degree). Software controls, on the other-hand, are deterministic. Safety is only

---

[12] Analyzing Software Specifications for Mode Confusion Potential, by Nancy G. Leveson, (et. al.) Workshop on Human Error and System Development, Glascow, March 1997.

[13] Leveson, Nancy; Safeware: System Safety and Computers; Addison Wesley Publishing Co. (appendix A)

assured when the safety-critical components (both hardware and software) work properly under all real world use conditions.

The book "Safeware: System Safety and Computer" by Nancy Leveson[14] provides analysis of the Therac-25 accident and several other accidents. The warnings provided by this compilation of accidents need to be heeded in the auto industry. In each case, the design of complex systems is assumed to be simplified by the use of computer controls. In many ways, this is true. Simplicity, however, can mean two things. It is the opposite of complicated but it is also the opposite of complex. Software controls often simplify our view of the system making the complicated simple by dividing it up into manageable chunks. The complexities of the interaction are, however, still present and must be managed in this new domain.

3.2.2 Examples of System Failure Modes

Specific failure modes are obviously system dependent but some general types of failure modes exist. Failures in system function can occur for a variety of reasons. These include:

1. Specification conflicts may be improperly resolved.
2. Specifications may be incomplete.
3. Teams are overconfident about software implementations.
4. There may be a lack of feedback in software control systems.
5. Teams confuse reliability with safety.
6. Failure to eliminate root causes.

---

[14] Leveson, Nancy; Safeware: System Safety and Computers; Addison-Wesley Publishing Co.,(appendecies).

7. Teams fail to confirm that the specifications are consistent (reconciliation is incomplete).

3.2.2.1 Specification conflicts may be improperly resolved

Specifications often get initially phrased in terms of rules and two or more rules may conflict with one another. This may be compounded by the principle of cognitive psychology that states that humans are more likely to value the things that they come into awareness of most recently. An example of this type of failure came to light in the interviews. The design team had developed a control strategy to ensure that the levels of the dozens of outputs of the electronic module would be in the settings expected by the driver even if the module was reset during operation. A specification for use of battery power was subsequently tightened and the software was modified to ensure that after reset unnecessary outputs were set to low power usage states. The information on the usages of each output was not easily available at the level where the control strategy was implemented. The system failure mode was that the module turned off a function expected by the driver whenever the module reset.

3.2.2.2 Specifications may be incomplete.

The incompleteness of specifications may occur in various ways. The P-Diagram approach is designed to help the team consider if the product is properly specified for the manufacturing variation, the environments of use, the customer uses (and reasonably foreseeable mis-uses), the effects of aging and the interactions with other systems in the product. Specifications have continually evolved in the auto industry to capture the lessons from the past in these areas. Software controls need to build upon those lessons. The general failure mode in this case is that

the functions are not available or act differently when the product is used outside of the expected parameters of use (e.g. high temperature or off road use).

3.2.2.3 Teams are overconfident about software implementations.

Correctly written software always works. There are two tendencies that result in problems. The first is that if the software code has worked previously we tend to believe that was correctly written. The second tendency is to believe that if each software component works correctly alone then together they will work properly. Each of these tendencies can allow bugs to escape the development phase and possibly even escape detection during testing.

3.2.2.4 Teams confuse reliability with safety.

Similarly teams often confuse the characteristic of reliability with safety. A system may operate reliably but still be unsafe. This is discussed further in the next section.

3.2.2.5 There may be a lack of feedback in software control systems.

Hardware systems often use the physical characteristics of systems to provide feedback about the system states. This feedback includes both feedback to the operator (driver or repair technician) and to the other components in the system. Safety devises such as thermistors, detect increases in currents when a motor drives a mechanical system against a hard stop. When the hard stop is replace by software control, the safety feature needs to be re-evaluated to ensure that the system safety is not compromised.

3.2.2.6 Failure to eliminate root causes.

Time pressures in the design phase may result in the failure to eliminate the root cause. The Therac-25 accident analysis by Leveson cites the failure to eliminate root causes combined with the re-use of software in a later design. The bug had no effect in the earlier model but due to changes in the later model the bug contributed to a lack of safety.

3.2.2.7 Teams fail to confirm that the specifications are consistent

Specification reconciliation is a term used, in the automotive PDP, to describe the process of ensuring that the teams know what is meant by the specifications imposed on them by other teams. The failure of one of the NASA mars missions in the 1990's was due to the fact that each of two teams believed that they were working with the same units of measure. The reconciliation process increases in difficulty with the number and type of interactions between systems. It can be compounded by the fact that the people writing the code may have little insight into the intent of the designers concerning how the control signals are actually used.

3.2.3 Errors that lead to System Failures

The most serious system failures are those that cause safety problems. In the auto industry many recalls are due to safety but a large number are due to failure of the vehicles to meet other requirements such as fuel efficiency, and pollution controls. There are various ways that safety and other attributes can be compromised by software and specifications. The requirements can be incorrect or impossible to meet, which means even if correctly implemented the software controlled system is still unsafe. The requirements may not specify some particular behavior

required for the system or vehicle attribute. It is also possible that the system has unintended behaviors that compromise an attribute such as safety. This can happen even if the requirements are complete in what they require the system to do and the software implements those requirements correctly.

3.3 Elements of Jidoka for PD

The goal of the automotive design teams is to design quality vehicles that people want and that can be mass-produced at a profit. Early in the design process the design team determines which vehicle attributes will be most important to the customers in the target market for the new vehicle. Alternative designs may be compared in their abilities to meet these attribute goals using a tool such as the Pugh Selection Matrix. The selected high-level design includes targets for the various attributes and an initial decomposition of the vehicle design team into systems design teams. Jidoka for PD evaluates the impact of a system on the functions and attributes of the higher-level system (e.g. the vehicle), defines errors that will lead to system failures, and provides methods to detect and correct the errors and their causes.

Error catching is most effective when it is applied early in the design process. At that time, the team must determine the customer wants and vehicle attributes, translate those into engineering specifications and cascade specifications to lower level systems. This Requirements Phase sets the stage for the Detail Design Phase and the Validations and Verification (V&V) Phase. Jidoka for PD focuses on ensuring that errors made during the requirements process are caught, communicated, and resolved as early as possible.

3.3.1 Defining Goals and Responsibilities

"The primary safety problem in computer-based systems is the lack of appropriate constraints on design. The job of the system safety engineer is to identify the design constraints necessary to maintain safety and to ensure the system".[15]

Systems safety engineering in the automotive industry has historically focused on two areas. The first is crash safety, which involves minimizing the effects on the humans in the vehicles when crashes occur. The second area is the incorporation of new safety features into vehicle design.

Historically in the auto industry, each system design team could manage the interactions and interdependencies of their system with each interfacing system on a one by one basis. The effective management of the design process was possible because any given system function had a strong affect on only one or two other systems. Software control, especially distributed control has the potential to change this dramatically.

Attribute teams are used to gather engineers from various system teams together to identify dependencies and resolve conflicts. The attribute team leaders act as a corporate knowledge base for historical issues and guidelines for avoiding known problems. System and attribute failures are traditionally handled by a study of historical problems and the development of rules and standards for avoiding those problems in the future. The system engineering approach and the method of Jidoka for PD are proactive and designed to constrain the down-stream design actions

---

[15] Analyzing Software Specifications for Mode Confusion Potential, by Nancy G. Leveson, (et. al.) Workshop on Human Error and System Development, Glascow, March 1997.

in order to assure that the attributes of a system emerge during integration. This approach is designed to enhance and not to replace the historical / lessons learned types of approaches.

Software safety engineering disciplines can be implemented at the component, system or vehicle level. The specific roles and responsibilities can be tailored to the level of need using a tool similar to the Safety Hazard Analysis (SHA) described previously. The use of the modified tool will be described in a later section. The US auto industry has learned that hard way that "Quality" is everyone's job. In a more general sense, any portion of a design that can degrade a vehicle attribute must be accountable for creating that attribute and for avoiding errors that could lead to failures of the system to function or create the attribute(s).

3.3.2 Defining Errors

Errors for the purpose of this paper are those actions or failures to act that result in failures of the system to create the system or vehicle functions or attributes at the desired time and in the desired way. MIT professor Paul Carlile[16] describes the three general ways that gaps can exist between organizations as syntactic, semantic and pragmatic. Errors in system design can come about from failure to come to agreement in any one of three general types of ways.

[16] Carlile, Paul; A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development; Organizational Science; July-August 2002.

3.3.2.1 Syntactic Agreement

The syntactic view of boundaries was developed by Shannon and Weaver[17] and is the foundation of cybernetic theory. Their mathematical approach to communication focused on being able to code and decode information at the appropriate times. Cybernetic Theory has been used to assist in the study of human understanding. In the PDP for a complex software and hardware process, shared syntax must bridge both the organizational gap between the supplier and OEM and the gap between the hardware and software teams. This view sees information exchange as the passing of data through a medium in the presences of some noise. Task failures that can occur due to a lack of syntactic agreement include when the timing of the information exchange is inappropriate or when the noise in the medium is too large to distinguish the important information. This may occur when the OEM has too many high priority tasks and no clear method for effective prioritization.

3.3.2.2 Semantic Agreement

Semantic difficulties can arise even if there is a common syntax and language. The auto industry and the computer industry use different meanings for similar terms. Both industries have developed methods for individuals to define their tasks, identify their interdependencies and to identify issues. These standard practices in the auto industry include statements of work, functional standards and FMEAs as well as other tools. Computer tools include State Diagrams and relational databases, which link requirements to models to test methods. Different teams of hardware and software engineers need to coordinate their actions using these semantic boundary

---

[17] Shannon and Weaver; The Mathematical Theory of Communications; University of Illinois Press; 1949.

objects. Even when these objects are well defined and appropriate for the tasks, issues between the teams may still arise.

3.3.2.3 Pragmatic Agreement

"The phenomenon of communication depends, not on what is transmitted, but on what happens to the person who receives it." **Maturana, Varela**

Pragmatic interactions are similar to the idea of knowledge as coordinated action. Coordinating action in the software development during a vehicle development involves hardware to software relationships, supplier to OEM relationships, and component to systems interactions. Each of these relationships involves decisions about who needs to know what and when they need to know it. Metaphors of a design chain make it difficult to manage each of these distinct sets of interactions. Boundary objects of this type include prototypes, models and databases including specifications linked to test methods and results.

3.4 Conditions for Jidoka for PD

The conditions necessary for Jidoka for PD are similar to the characteristics of healthy design teams. Edwards Deming[18] provides a description of the principles that result in healthy design teams that has become familiar to the auto industry. This description has been codified as

---

[18] *Out of the Crisis*, By W. Edwards Deming, Published by M.I.T., 1986

Deming's Theory of Profound Knowledge and his 14 points for the Transformation of Management. The four aspects of the Theory of Profound Knowledge are:

1) Appreciation of Systems
2) Understanding of Variation
3) Understanding of how people know and understand things
4) Understanding of how to motivate and lead

Catching system errors requires a systems approach. This is enhanced by formal definitions of subsystems, interactions and roles at the various levels of the system-subsystem hierarchy. Finding the root causes of the errors is greatly enhanced by traceability of requirements and reconciliation of specifications. Reconciliation of specifications is the process of ensuring that interfacing systems meet the design needs of each other. Closed-loop communication ensures that the teams' decisions are based on results and not just on historical agreements and assumptions.

4 MODIFYING DESIGN TOOLS FOR JIDOKA

System engineering and design considers the vehicle (as a whole system), its features and attributes and how it is created from its systems and subsystems. The choice of tools depends upon the specifics of the design task and the need to assure success. The need to assure success is based upon the severity of the failure to do so. Thus a first step is to understand the failure modes.

4.1 Understanding Product Failures

One way to understand the failure of electronic modules is to analyze the reported field failures. There are two types of information available: warranty data and analysis of returned parts. Analysis of each of these types of information suggests that service technicians may be having difficulty finding the root cause of the customer complaints. In an effort to gain insight into real world failure modes, the warranty payment systems have been expanded to collect information about the customer symptoms, and the part conditions in addition to the information required to pay the warranty claims. The warranty system provides a large amount of information that is not very detailed. The analysis of returned parts provides more in-depth information on a much smaller set of parts.

4.1.1 Warranty Analysis

Warranty data contains two types of information on the repair that may provide insight into the failure that led to the repair. The first piece of information is the customer symptom, which is

recorded as the Customer Concern Code (CCC). These symptoms are grouped together and assigned to the design teams responsible. The second type of information is part-related information. This information includes the part(s) that the technician replaced. For electronic modules the part information may include diagnosis information sometimes called Diagnostic Trouble Codes (DTCs). The DTCs may be stored by module self-tests during customer usage or during technician requested diagnostic tests. DTCs record both faults in the electronic modules and the systems that they control.

One measure of the interactions between systems is the extent that customer experienced symptoms are assigned to multiple design teams (rather than to the specific team that had the design responsibility). The following figure is a table showing the design teams that release the parts that have symptom codes assigned to the Electrical Team. If the symptom codes were unambiguous the mapping of symptoms to causal parts would be direct for all component failure modes. This data could thus support either that there is ambiguity in the symptoms or that there are system failure modes that do not have their own codes

Percentage of Repairs with Electrical Symptoms by	
Hardware Design Team	**%**
B3 - ELECTRICAL	47%
B1 - BODY STRUCTURE	20%
B7 - CLIMATE CONTROL	18%
B5 - POWERTRAIN	6%
B2 - BODY INTERIOR	5%
B4 - CHASSIS	2%
B8 - FUEL SYSTEMS	2%

Figure 13: Hardware Team responsibility for Repairs with Electrical Symptom Codes

The author also analyzed data associated with warranty repairs of three different electronic modules on four different vehicles (a truck, a minivan, a sedan and a luxury sedan). Electronic modules were chosen that store DTCs. Diagnostic Trouble Codes are designed into the system to assist the technicians in identifying the root cause of the customer complaints. The following figure shows that about 2/3 of the time that the electronic modules were replaced, there was no DTC that indicated a problem. The lack of DTCs suggests that the causes of some of the symptoms reported by the customers are not due to failed parts, but rather they are system failures. Analysis of the returned parts is an important tool to confirm this hypothesis. The next section reports on the analysis of parts that have been retrieved from dealerships and analyzed by suppliers.

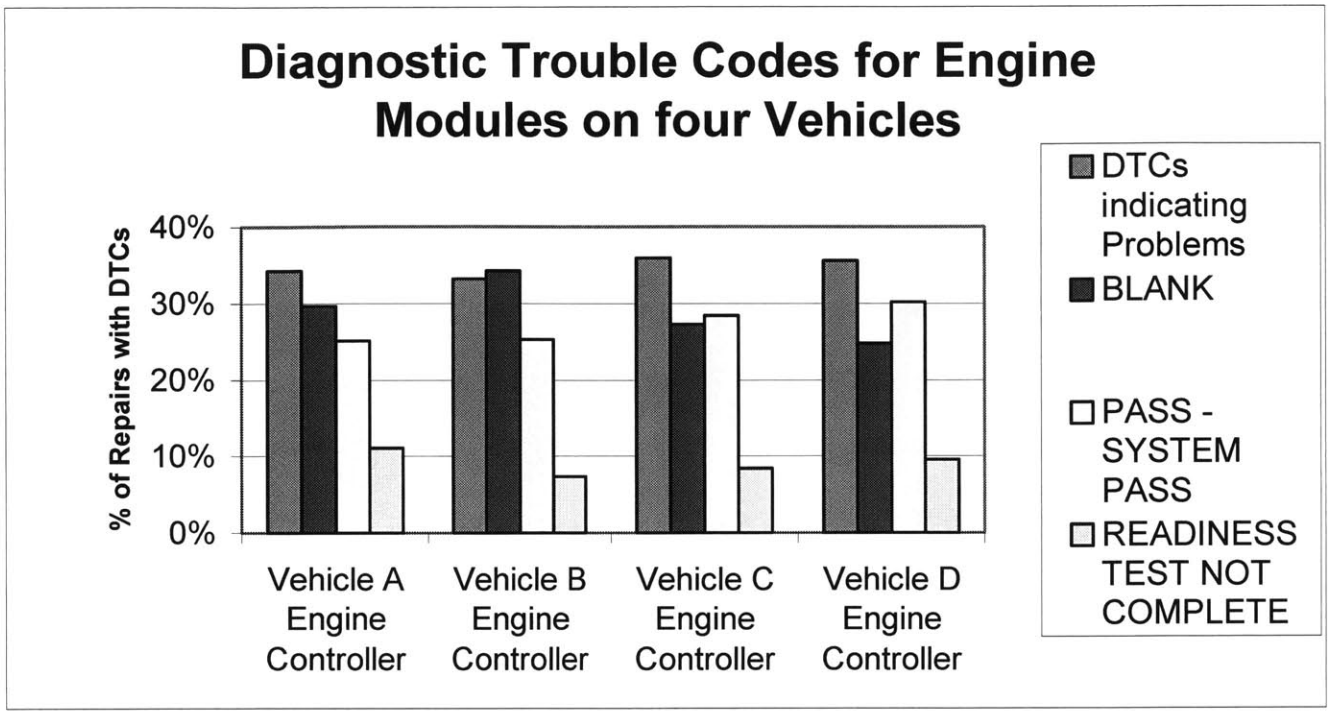


Figure 14: Pareto of Diagnostic Trouble Codes for Module Warranty

4.1.2 Root Cause Analysis of Returned Parts

The analysis of returned parts for defects can provide insight into the root causes of the symptoms that customers experience The author's experiences and interviews with engineers responsible for the failure analysis of the returned parts are described in this section. A large percentage of the electronic modules that have been returned are identified as having "No Trouble Found" (NTF). By this phrase, the suppliers mean that the parts that they analyze meet specifications. The percentage of parts returned labeled NTF varies from 50% to 90% of the parts analyzed for electronic modules.

The analysis of returned parts can be complicated by the fact that the analysis is done in a lab in a different environment than the environment where the failure occurred. The best failure analysis techniques check the operation of the returned part at high and low temperature and during vibration. These are the environmental noise factors that most often affect the performance of electronic modules. Improving the failure analysis techniques reduces the number of NTF units to 30 to 70%. Solder joint failures are one of the most common root causes of these newly confirmed failures. These failures may be intermittent and depend upon heat or vibration to occur, making them difficult to detect. Systems Failures are defined as those failures that cannot be attributed to the failure of parts to meet their individual specifications. Traditional reliability disciplines are appropriate to address the part failures, but systems approaches are needed to avoid system failures. The auto companies need their suppliers' expertise in their specific areas to ensure high quality components and systems and to act as a source of innovation that will continue to provide competitive advantage. This raises the question of how can and should the industry evaluate their suppliers.

4.2 Evaluating Software Suppliers

The auto industry has traditionally evaluated their suppliers on the basis of past performance. The past performance is measured in several ways. One traditional measure used is the number of quality issues (quality rejects - QRs) discovered during vehicle assembly. Another measure is the suppliers' record of meeting design milestones. Neither of these metrics measures the suppliers' awareness of and partnership in dealing with systems issues. The defense industry has adopted a method to evaluate software companies called the SEI Capability Maturity Model. CMM attempts to evaluate the capability of a company to supply software.

4.2.1 CMM could it help?

Capers Jones (in "Assessment & Control of Software Risks", Prentice-Hall, 1994) describes some of the weaknesses of CMM. The weaknesses he cites include the management of the people and of the innovation process. CMM does not address the factors that contribute to the productivity of individual engineers. James Bach in his 1994 article points out that CMM "*reveres process and ignores people.*" He also criticizes it for failing to develop structures that enhance innovation and worse he states that: *" Preoccupied with predictability, the CMM is profoundly ignorant of the dynamics of innovation."*

These points are critically important for the automotive OEMs who count on their software suppliers for partnership and for innovation. In addition, the automotive industry requires system design tools that allow coordination between many teams and several suppliers. CMM offers no direct assistance in these areas.

The CMM does help define a common language, which can be used to reduce syntactic and even semantic differences between the design teams. A common language is of great importance but is generally obtainable in other ways such as through the Product Development Process of the OEM and the design guidelines in both the electronics industry and the auto industry. The PDP of the OEM sets the tempo and defines the points for interaction between the working partners in the design process. Using the CMM in the automotive industry would risk the loss of technical leadership and would substitute a process focus for the technical focus in product development.

4.3 Reliability and Robustness Expectations and Tools

Suppliers and the OEM electronic module design teams need an understanding of component reliability techniques to ensure that the parts work in their use environment for their design life. The design teams also need awareness that those tools and a component focus is not sufficient to produce reliable and safe systems especially when the systems are characterized by organized complexity. The teams need to be willing to extend traditional reliability and robustness (R&R) methods and tools beyond components to systems. Existing tools are necessary to deal with many of the historical and recurring issues in product design. These tools include Design Failure Modes and Effects Analysis (DFMEA), P-diagrams, and worst-case tolerance analysis. Using these tools the automotive industry has become expert at providing components that meet specifications and can be mass-produced. This industry needs to adopt new tools and adapt existing tools in order to assure that vehicle attributes (such as safety, emissions, fuel efficiency, robustness and in general usability) emerge when the components are assembled into the vehicle.

4.3.1 P-diagram Revisited

The P-diagram was developed for and creates real value in systems characterized by organized simplicity. It such systems the outputs are well understood functions of the inputs. The design team then uses the P-diagram to help understand how that function is modified in the presence of the noise factors. Taguchi [19] recommends using the signal-to-noise metric to minimize wasted energy in order to minimize the energy available to create error states. This technique is very

---

[19] Taguchi, G.: *Introduction to Quality Engineering—Designing Quality Into Products andProcesses*, 1986.

applicable to systems that translate one type of energy into another (such as pistons to crankshafts or drive shafts to axels). As a result this tool has found a home in automotive system design. The approach of minimizing energy loss is not possible when the error states differ from the ideal function only in terms of timing and coordination with other outputs, as is the case in most software control systems.

Successful change management begins with a realistic awareness of the current organization and its established practices. During a time of change it is often best to extend familiar tools rather than to try to replace them with new tools that anticipate future needs. As describe earlier in the paper, the P-diagram is currently used in the auto industry. The familiarity of the format to management allows the use of this tool to communicate information about systems to systems interactions. It is also possible to include information about the system and vehicle level attributes that the system has an impact upon as well as the attributes that may cause architectural risk to this system. I use the term architectural risk to describe requests for change to the architecture of a system after the detailed design phase is underway. In the headlight example, a late realization that fuel efficiency targets were not being met resulted in requests for changes to the electronic module hardware and software in order to reduce the power used by the headlights. Early understanding of the system-to-system interactions and the attribute-to-system interaction can allow for more design flexibility and earlier detection of issues.

4.3.1.1 P-Diagram Applied to the Module as a System

The P-diagram can be used in systems with multiple functions but it soon becomes impossible to use it in the intended way, which includes optimizing the primary function and prioritizing the

noise factors. Prioritizing the noise factors becomes problematic since the noise factors that are critical to one system function may be irrelevant to another system function. Keeping track of the system functions and linking the noise factors becomes a job more suited to a relational database than to a diagram. Since the noise factors are constraints that operate on the system they can also be thought of as specifications that need to be reconciled with other systems or components.

Electronic Modules may have 30 to 100 inputs and outputs that go through one or more connector. The software in these modules may control or affect dozens of functions and each of these functions may have restrictions on functionality based upon regulations or markets. The modules may play roles of different strengths in various vehicle attributes. The example of controlling headlights will serve to illustrate the complexity that can arise from a simple control problem. The issues will get even more complicated with Interactive Vehicle Dynamics and Engine-Transmission Controls where the timing constraints are even greater.

A single customer function such as headlight operation may be controlled by a single module or by two modules working together. The operation of this function may affect various vehicle attributes such as safety, ergonomics, fuel efficiency and dependability. The rules for controlling the headlights and fog lamps vary from country to country. The outputs of the module, used to control the various lamps, are dependent upon the current states of the outputs, inputs from the driver, and the country of sale. Two additional factors are the series level and the key position. The higher-cost series of a vehicle model is often distinguished from the lower cost version (or series) by additional features. Those additional features might include automatic dimming of the

headlight high beams or automatic shut-off of all lights after the key is removed (possibly after a driver selectable delay time).

The need for improvement in another attribute may influence the design choices for the modules(s) controlling the headlamps. To increase fuel efficiency it is often desirable to minimize the unnecessary use of the alternator. Headlights draw significant power that can be reduced using pulse width modulation control techniques. Thus the desire for an improvement in a vehicle attribute such as fuel efficiency can influence the design choices for the module design and the software control strategies.

Another layer of complexity is added to provide error state detection. The detection of a condition which indicates that a lamp is not working needs to be communicated to the driver (usually through a different module) and the diagnostic trouble code needs to be stored for later retrieval by the service technician.

Imagine the increasing complexity as driver inputs are translated into electrical signals by sensors (drive-by-wire), interpreted by computers that control the historically electrical, mechanical and hydraulic functions in the vehicles. The simple tools such as the P-Diagram are inadequate to define all the primary functions these modules control. A relational database is a more appropriate tool for this purpose.

4.3.2 Specifications

The architecture sets the foundation for the specifications. In the auto industry it is common to bookshelf and re-use specifications. The practice of book shelving institutionalizes corporate memory. This practice has been done for generations of component designs and is now being applied at the systems and even the vehicle level. Computerized document control helps link related specifications and can provide links to information on the intent of the specification. This approach has been taken a step further in the software engineering profession where the specifications, models of behaviors and code are all included in the same design tool. The Safeware tool and process described below provides a formal hierarchy for the specification and (most importantly for Jidoka for PD) a formal linkage between specifications (and between specifications and models).

4.3.3 Vehicle to Component Hierarchy of Specifications

Traditionally the specification hierarchy in the auto industry is defined as in the system V model described above. The hierarchy sometimes begins with a platform, which has specifications that are flowed down to the vehicle. Vehicle level specifications are flowed to the Level 1 systems, which are usually decomposed into subsystems. These subsystems are decomposed into components. In a similar way, targets are cascaded from the desired vehicle level attributes to the systems, subsystems and components. Design verification methods are used to provide evidence that the targets were met and that the design team has completed their tasks and can move the design into mass production. Thus the teams need specifications that can be traced from their origin. The origin of the specification may be from a needs statement for this system (which

should be captured at a higher level in the hierarchy), or from an interfacing system on the same level of the hierarchy.

4.3.4 Specification Hierarchy based on Logical Typing

A complete set of specifications has various different logical types of information. The first is the context information. Context information includes program context, references to the statement of work and other agreements between the companies involved. Context also includes such things as agreements to work to standards such as ISO9001 or Q1.

Below the context information is the level of general or systems goals and constraints. The set (or superset) of environments, markets and customer level features are in this logical type. This type or level of information may also include the architectural principles and priorities established by the systems architects (both vehicle and software). Design principles are a more concrete level, which include a decomposition of the control system to the various modules and an indication of which things should be performed with more flexibility and which can be hardware or firmware. The next level of specification involves even more concrete issues such as communication protocol(s), interrupt priorities, I/O specifications, diagnostic strategies and other black box behaviors

.

In the course of this research I became aware of Intent Specifications; a tool being designed to provide a design team with a hierarchical approach to specification and implementation of software projects.

4.3.5 Intent Specifications

Safeware Engineering developed the Intent Specification Tool using a seven level specification hierarchy. This tool allows the design team to link specifications at the various levels of the hierarchy with each other in a formal and traceable manner. The specifications can also be linked to models of the desired behaviors, the code level components and even the test methods. The tool provides a method to check for consistency, to identify differences and to evaluate the impact of changes. These three properties are important aspects of catching and avoiding errors and are thus critical for Jidoka in product development.

The logical levels used in this tool are defined in the figure below.

Specification Levels in Intent Specifications

Level 0: Program Management Information
Level 1: System-Level Goals, Requirements, and Constraints
Level 2: System Design Principles
Level 3: Blackbox Behavior
Level 4: Physical and Logical Design Representations
Level 5: Physical Implementation
Level 6: System Operations

Figure 15: Specification Levels from Leveson's Intent Specifications

The use of this tool is relatively straightforward. Existing specification can easily be cut and pasted into the tool. The tool also allows for models of systems behavior to be attached and linked to the specifications. It also allows the actual code to be linked to the models and specifications.

4.3.5.1 Level 0: Program Management Information

This level is designed to include the Program Management Plan and the System Safety Plan. It can easily be expanded to describe Platform and Vehicle level assumptions and constraints as well as market expectations, which drive the regulatory environment that the product must conform to in order to be saleable.

4.3.5.2 Level 1: System-Level Goals, Requirements, and Constraints

This level is intended to include various assumptions, goal statements, constraints and requirements. An example specification can be used as a template and includes an introduction and sections for: historical information, environment description, assumptions, and constraints, system functional goals, high-level requirements and safety constraints.

4.3.5.3 Level 2: System Design Principles

This section can be used for specifying the System Interface Design, Operator Task Design Principles, System Design Principles, Simulations, and Analyses including System Hazard Analysis.

4.3.5.4 Level 3: Blackbox Behavior

Behavioral Assumptions and Models of the System Environment can be specified in this section along with Communication Protocols, Message Contents (such as diagnostic trouble codes) and Failure Indications and handling methods.

4.3.5.5 Level 4: Physical and Logical Design Representations

The Software and Hardware Design Specifications at a representational level come next. The linkages to the source of the specification can also be added.

4.3.5.6 Level 5: Physical Implementation

The physical representation includes Hardware Schematics, the source code, Verification and Validation results and Operational Safety Analysis.

4.3.5.7 Level 6: System Operations

Systems operation includes the specification of the states where a normal function is not allowed.

Historically rules were used to define when functions could occur. As the number of states and interactions increased, state diagrams became important tools. In very complex systems a relational database approach is needed to manage the set of rules and states that are used to implement the specifications.

5 ORGANIZATIONAL ISSUES IN THE APPLICATION OF JIDOKA TO AUTOMOTIVE MODULE DESIGN

"Organizational change almost inevitability becomes a learning process in which unanticipated obstacles and opportunities emerge."[20]

Jidoka is a commitment to continuous improvement of design process rather than a specific process. In this sense, it requires leadership at all levels and a commitment to resolve conflicts earlier rather than later. Change involving computer control of automotive systems is occurring but there are competing trends. One trend is toward centralized computer controls while another trend is toward the incorporation of larger and larger numbers of smaller and smaller computers with decentralized computational abilities. The key for the management of the change process is to focus on creating systems that are increasingly safe and that consistently create the emergent attributes that are desired in the vehicle.

Managing change is difficult even when the end state of the change process is known and agreed upon. Brynjolfsson (et. al) said, "It may not even be clear whether the best course is to strive for radical change, incremental change or no change at all, even if a potential organizational goal is precisely envisioned and represents an unambiguous improvement." [21] This quote illustrates the motivation for applying a continuous improvement / Jidoka philosophy to the PD organization.

---

[20] Orlikowski, W., & Hofman, D. (1996). An Improvisational Model of Change Management: The Case of Groupware Technologies. Sloan Management Review

[21] Brynjolfsson , Austin, Renshaw and van Alstyne; The Matrix of Change: A Tool for Business Process Reengineering, MIT Sloan School of Management January, 1997

5.1 Tailoring Systems Approach and Jidoka

System complexity and failure criticality are used in system safety engineering to determine the scope of the safety program. This is done to provide an effective use of resources and to ensure that critical parts of the design receive the attention required to an acceptable level of safety. The scope of the systems engineering effort in the auto industry should also be tailored to ensure that the goals of the design are met both effectively and efficiently.

Criticality in automotive electronics needs to extend beyond safety to include regulatory, liability and other issues that are critical to the success of a new design. Jidoka and other system engineering approaches do not need to be universally applied. The following figure was adopted from NASA Guidebook for Safety Critical Software - Analysis and Development NASA-GB-1740.13-96. In chapter two, I presented two figures from NASA: System Categories and Hazard Severity Levels. Those tables are used by system safety engineering teams to tailor the safety engineering tasks for a particular system design. The following approach uses the System Category presented earlier. The Attribute Criticality Levels presented below are substituted for the Hazard Severity Levels in order to ensure that the appropriate design tools and methods are used.

Figure 16 Attribute Criticality Levels

Attribute Criticality Levels

Critical to Safety or Regulation – The, or one of the critical contributors to the attribute.
Important to safety and other attributes – An important contributor to the attribute
Moderate – Some impact on vehicle level attributes
Minimal - No impact on vehicle level attributes

The attribute criticality levels can now be combined with the System Categories to determine the scope of the activities for assurance of the success of the vehicle design. The figure is presented with criticality increasing from left to right and system complexity increasing from bottom to top in order to be more consistent with auto industry practices.

SYSTEM CATEGORY	ATTRIBUTE CRITICALITY LEVEL			
	Critical to Safety or Regulation	Important to safety and other attributes	Moderate	Minimal
I	Full	Full	Increased	Normal
II	Increased	Increased	Normal	Normal
III	Normal	Normal	Normal	Normal

Figure 17: Table for Tailoring Design Activities

The figure above describes a method for tailoring the scope of a software systems design approach for the auto industry. The details of the program scope levels "normal, increased and full" can legitimately vary from company to company. In a later section I propose general guidelines for the two enhanced levels; the normal level will be assumed to be the standard approach for the industry.

5.2 Bridging the Organizational Gaps

Gaps between the design teams need to be bridged by the leaders of the vehicle and system design teams. These teams include members of different professions with different skills. The

teams also contain people from different organizations and different companies, each with their own roles, cultures and goals.

Academics who study Organizational Behavior define Boundary Objects as the set of tools, processes and records that are used to communicate between organizations. Carlile[22] asserts that to be affective, the boundary objects need to ensure that there is Syntactic, Semantic and Pragmatic agreement between the organizations. Syntactic boundary objects are characterized by representation; semantic boundary objects are characterized by both representation and learning; Pragmatic boundary objects are characterized by representation, learning and transformation.

5.2.1 Syntactic Agreement

The syntactic view of boundaries was described and turned into a set of tools by Shannon and Weaver[23]. It is the foundation of cybernetic theory. Their mathematical approach to communication focused on being able to code and decode information at the appropriate times Cybernetic Theory has been used to assist in the study of human understanding. In the PDP for a complex software and hardware process, shared syntax must bridge both the organizational gap between the supplier and OEM and the gap between the hardware and software teams. This view sees information exchange as the passing of data through a medium (in the presence of some noise) from a sender to a receiver. Task failures can occur due to a lack of syntactic agreement in terms of time or format of the information. Many companies use tools to manage this sort of

---

[22] Carlile, Paul; A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development; Organizational Science; July-August 2002

[23] Shannon and Weaver; The Mathematical Theory of Communications; University of Illinois Press; 1949.

communication and more tools are not the answer. The commitment to full and open communication is required to make this work. The tools assist in reducing redundant work by linking the proper people to the issues.

5.2.2 Semantic Agreement

Semantic difficulties can arise even if there is a common syntax and language. The auto industry and the computer industry use different meanings for similar terms. Both industries have developed methods for individuals to define their tasks, identify their interdependencies and to identify issues. These standard practices in the auto industry include statements of work, functional standards and FMEAs as well as other tools. Computer tools include state diagrams and relational databases, which link requirements to models to test methods. Different teams of hardware and software engineers need to coordinate their actions using these semantic boundary objects. The use of common modeling tools and common specification tools can lead to more successful semantic agreement because the tools force common definitions of many terms. Even when these objects are well defined and appropriate for the tasks issues between the teams may still arise due to the lack of pragmatic agreement.

5.2.3 Pragmatic Agreement

"The phenomenon of communication depends on not on what is transmitted but on what happens to the person who receives it." **Maturana, Varela**

There are many types of relationships in and among the design teams. Each of these relationships involves decisions about who needs to know what and when they need to know it. Boundary objects are the things that the teams use to coordinate their actions; they include prototypes, models and databases including specifications linked to test methods and results.

5.3 Defining the Program Scope Levels

Three levels for the program scope "normal, increased and full" were introduced in an earlier section. These levels are intended to allow the scope of the design effort to be appropriate to the complexity of the systems and the risk involved with the design. The idea of scooping the design process can be applied to all phases of the PDP. The concept of Jidoka for PD is most applicable to the early phases of the design process although the philosophy should be used throughout the PDP.

5.3.1 Normal Program Scope

Normal product design scope has been standardized in the automotive industry. Most companies have an established Product Development Process (PDP). The general expectations have been

agreed to and developed into standards for suppliers. The primary standard is the AIAG APQP[24]. The expectations are defined as 23 elements or disciplines. These elements take place throughout the design process and are intended to ensure that the components supplied to the assembly plants will consistently meet the customers' expectations in the environments of use. Customers in this definition are the assembly plant and the service technician as well as the buyers and drivers of the vehicles.

5.3.2 Full and Increased Program Scopes

Full program scope should start with a commitment from the OEM management team to create an organizational culture where admitting and finding mistakes is considered to be positive. Dr. W. Edwards Deming's 8th Point (of his 14 Points for the Transformation of Management[25]) is to "Drive Out Fear". Another key point is Point 5 "Continuously Improve Systems. Jidoka for PD is built upon these two points. The design team, both suppliers and OEM, must work together as a partnership to pull off the design of successful products in today's competitive environment.

The following are the scope increases for the Full and Increased program scope (these are in addition to the standard elements defined by the standard PDP and APQP).

- Vehicle and System Architectural Tasks.

---

[24] APQP Advanced Product Quality Planning & Control Plan (APQP); Automotive Industry Action Group, Detroit, Michigan 48277-0839 USA; February 1995

[25] Deming, W. Edwards; *Out of the Crisis*, M.I.T. Press, 1986. (chapter 2)

- Vehicle and System Level Attribute Criticality and Safety Hazard Analysis.
- Classification and Cascading of Requirements.
- Verification of safety and attribute requirements, designs and product performance

5.3.2.1 Vehicle and System Architectural Tasks

Full program scope should include vehicle level architecture including software and control strategies. The increased scope may only require a system architectural description. Previously, I listed Prof. Crawley's deliverables of the architect. The IEEE (Institute of Electrical and Electronic Engineers) professional society has developed a Recommended Practice for Architectural Description of Software-Intensive Systems [26]. The following figure presents some of the major elements of the architectural description.

Naming Standard Context Diagram Introduction and Requirements Functional Architecture Principles Safety Strategy Generic Model Modeling Style Standard Functionality Development Guideline Architecture Development Guideline. Model Description Function Component Principles Tools and Tool Descriptions

Figure 18: Recommended Practices for Architectural Description from IEEE

[26] IEEE Recommended Practice for Architectural Description of Software-Intensive Systems [IEEE Std 1471-2000].

The book Safeware Systems, Safety and Computers[27] provides more detail on software safety and design approaches that are consistent with the philosophy of Jidoka for PD. In chapter18, Leveson argues for the use of formal methods in software engineering[28]. She states: "Formality provides the ability to reason about, analyze and mathematically manipulate system descriptions." Automotive engineers are used to mechanical systems where it is hard for them to conceive of systems that do not have these traits or the need to require formality from the suppliers of software engineering services. Formal methods including structured and object-oriented software methodologies emphasize early testing of facts and strategies. The use of these methods has started to reduce major problems and expensive corrections during vehicle level testing. They have also been cited as reducing the probability of recalls.[29]

5.3.2.2 Attribute Criticality and Safety Hazard Analysis

The concepts of Attribute Criticality Analysis (ACA) and Safety Hazard Analysis (SHA) were presented earlier. The ACA and SHA can each be performed at the vehicle or system level. In addition, a preliminary analysis can be done early in the design process, even before the final decomposition is created. The preliminary analysis will determine which system or subsystems need to perform detail analysis of the impact on the critical attributes.

---

[27] Saferware: Systems, Safety and Computers; Nancy Leveson; Addison Wesley Publishing Co. 1995. page 225.

[28] Saferware: Systems, Safety and Computers; Nancy Leveson; Addison Wesley Publishing Co. 1995. page 496.

[29] Winant, Becky; "Getting to Grips with Software Engineering Trends"; Automotive Manufacturing International; 1999

5.3.2.3 Classification and Cascading of Requirements

The Requirements Phase of the automotive PDP takes the various customers, regulatory and corporate needs and wants and translates them into engineering requirements. The intent of these requirements may not be captured in the requirement. If there is a need for an increased or full program than the intent needs to be captured. One way to capture the intent is to classify the requirements (critical, important, moderate or minimal) with respect to the various vehicle attributes (safety, emissions, fuel efficiency, security, etc.). This classification can be added to a document such as the P-diagram. A better and more complete approach is to use a relational database tool to link the analysis with the requirements and to the components that implement the design.

5.3.2.4 Verification and Validation

Verification and validation of design choices that have an impact on safety and attribute requirements can be started in the requirements and early design phases using modeling techniques. To assure consistency of models and the ability for models to interact, companies are creating Vehicle Model Architectures (VMAs). VMAs are design frameworks (or agreements) that define fundamental components of the models, the relationships between these components, and the environments for the models used on a vehicle. The purpose of models is to check design assumptions and behaviors in the design phase rather than waiting for prototypes. Compatible models allow the design teams to discover and "iron out" the wrinkles in the design process before they make it into the product. The models can be reused in many cases adding efficiency to the design phase but their primary function is to ensure effective design coordination and early error detection.

6 CONCLUSIONS

6.1 Main Conclusions

Software controlled systems are allowing automotive designers to do more and more with less and less. This increases the complexity of the systems that they are designing by increasing the number of interfaces and the tightness of the coupling between the systems. Vehicle and System attributes are often created by the interaction of multiple systems that are increasingly dependent upon software for control and coordination. Assuring these attributes, which include safety, emissions, and fuel efficiency, requires the cooperation and partnership of the various component design teams and the management team responsible for the vehicle design.

Jidoka for Product Development offers leaders a vision to articulate when they are trying to pull a diverse product development team together into a partnership. The philosophy of Jidoka – discovering and fixing errors and their root causes - requires a system approach to implement.

Building safe software requires attention to safety throughout the entire software development process. When software interacts at the system level, architectural decisions are needed at the system level. When software interacts at the vehicle level, architectural decisions are needed at the vehicle level. Thus the vehicle architecture needs to include a vehicle level control architecture that includes software.

The most effective tool for making things safer is simplicity. Simple systems can be thoroughly modeled and are thus predictable. When system complexity increases the design effort also needs to increase to assure emergent properties such as the vehicle attribute of safety. The assurance of safety can be evaluated only in the context of the system in which it operates. This suggests the need for a preliminary analysis of the risk prior to detailed design phase and a final analysis when the design concept has been solidified. These practices have been established in industries that already have complex software controlled systems. The automotive industry will either need to keep the systems simple (uncoupled and few interfaces) or adapt method to deal with the complexity.

If the system cannot be made to be less complex then new tools such as Intent Specification and relational databases that link specifications to verification methods can be used to make the management of the complexity less complicated. Making these tools capable of meeting several needs at once will allow them to reduce the number of independent tasks in the PDP, which will help the teams fight the drift toward firefighting.

The earlier safety is considered in development, the better the results. Choosing to use formal methods in the software design is an early decision that has been cited as reducing the risk of recalls. Determining the errors that lead to product failures in the requirements phase provides an opportunity to address root causes both of the technical problems and of the organizational problems that may lead to them.

Concentrating only on technical issues and ignoring managerial and organizational deficiencies will not result in effective safety programs. The automotive manufacturers rely upon their suppliers for innovation, design expertise and partnership in making the trade-off decisions that are needed in any complex design process. This means that the leaders of the teams need to be aware of the organizational gaps that can cause communication problems. The leaders need to promote coordination between teams and within team (between the team members). Leadership is needed to ensure that the teams have the tools and desire to overcome syntactic, semantic and pragmatic issues that result when people have different expertise. This is critical because safety is a system problem that can only be solved by experts in different disciplines working together.

REFERENCES

APQP Advanced Product Quality Planning & Control Plan (APQP); Automotive Industry Action Group, Detroit, Michigan 48277-0839 USA; February 1995

Argyris, Chris; Overcoming Organizational Defenses: Facilitating Organizational Learning, April 1990

Bach, James, The Immaturity of CMM; American Programmer; September 1994.

Bateson, Gregory; Steps to an Ecology of Mind; Ballentine; 1972.

Brynjolfsson , Austin, Renshaw and van Alstyne; The Matrix of Change: A Tool for Business Process Reengineering, MIT Sloan School of Management January, 1997

Browning, T.R., J.J. Deyst, Steven.D.Eppinger, and Daniel E. Whitney Complex System Product Development: Adding Value by Creating Information and Reducing Risk, Proceedings of the Tenth Annual International Symposium of INCOSE, Minneapolis MN, July 2000

Carlile, Paul; A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development; Organizational Science; July-August 2002.

Deming, W. Edwards; *Out of the Crisis*, M.I.T. Press, 1986.

Hamilton and Smith; "Implementing TQM on a Shoestring"; Journal of Management Consulting; Vol. 7, No. 4; Fall 1993.

Harrington, Fred; Web glossary; http://www.fredharriman.com/services/glossary/jidoka.html ; 2002.

IEEE Recommended Practice for Architectural Description of Software-Intensive Systems [IEEE Std 1471-2000].

Jones, Capers, Assessment & Control of Software Risks, Prentice-Hall, 1994

Leveson, Nancy; Safeware: System Safety and Computers; Addison-Wesley Publishing Co., 1995.

Leveson, Nancy; System Safety in Computer-Controlled Automotive Systems. Society of Automotive Engineers; Invited Paper, 2002 Xadfh –check reference

NASA Guidebook for Safety Critical Software - Analysis and Development; NASA-GB-1740.13-96

Orlikowski, W., & Hofman, D. ;An Improvisational Model of Change Management: The Case of Groupware Technologies. Sloan Management Review, Winter 1996.

Maturana, H. R.; and F. J. Varela. The Tree of Knowledge; *Boston and London*, 1992.

Paulk, Mark, et al, Capability Maturity Model 1.1 (CMU/SEI-93-TR-24)

Peters, Tom; Thriving on Chaos: Handbook for a Management Revolution, HarperCollins, 1987

Phadke, Madhev, Quality Engineering Using Robust Design, Prentice Hall PTR, 1989.

Repenning, Nelson P., Resource Dependence in Product Development Improvement Efforts, Sloan School of Management, December 1999.

Repenning, Goncalves, Black; Past the Tipping Point: The Persistence of Firefighting in Product Development, Sloan School of Management.

Shannon and Weaver; The Mathematical Theory of Communications; University of Illinois Press; 1949.

Senge, Peter, The Fifth Discipline, Doubleday, 1990

Senge, Peter, The Fifth Discipline Fieldbook, Doubleday, 1994

Taguchi, G.: *Introduction to Quality Engineering—Designing Quality Into Products and Processes*, 1986.

Ulrich, Karl and Eppinger, Steven; Product Design and Development; McGraw Hill (2000).

Wallace and Ippolito; "A Framework for the Development and Assurance of High Integrity Software "; National Institute of Standards and Technology; NIST Special Publication 500-223 December 1994

Weinberg, Gerald. (1975). An Introduction to General Systems Thinking. New York: Wiley.

Winant, Becky; "Getting to Grips with Software Engineering Trends"; Automotive Manufacturing International; 1999.

Winograd, Flores; Understanding Computers and Cognition; Ablex Publishing Co.; 1986.