ABSTRACT

ZHU, DI. A New Design-Structure-Matrix Development Framework. (Under the direction of Lawrence M. Silverberg and Ewan G. D. Pritchard).

As complexity increases, model-based approaches have become standard in the systems engineering community. Although systems have become more complex, requiring more attention to testing, system methodologies have not kept up. In addition, engineers from different domains approach the system/sub-system/component development at different levels of abstraction. Information gaps are created because of their different perspectives. Design changes also create information gaps. A new framework is needed to bridge the information gaps between the different domains. To take full advantage of both model-based approaches and testing, a new Design-Structure-Matrix-based framework for complex, evolutionary, and iterative system development that utilizes information flow from model-based testing (MBT). Toward the goal of being broadly applicable, the framework was designed to be fundamental, flexible, and scalable. Since complex system development (CSD) and systems-of-systems development (SoSD) are typically complex, evolutionary, and iterative, one case study of each system development was selected and conducted to demonstrate how the proposed framework bridges gaps in early development stages. The results have shown that using the framework can close information gaps earlier, estimate duration of development elements, and predict effects of design changes.

A New Design-Structure-Matrix Development Framework

by
Di Zhu

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Mechanical Engineering

Raleigh, North Carolina

2017

APPROVED BY:

Lawrence M. Silverberg Committee Co-Chair	Ewan G. D. Pritchard Committee Co-Chair
Gregory Buckner	Alexei Saveliev
Iqbal Husain	

DEDICATION

To my family

BIOGRAPHY

The author was born in Nanjing, China, in 1985. He received the B.S. degree in automotive engineering from Shanghai University of Engineering and Science, Shanghai, China in 2007 and a M.Eng. degree in automotive engineering from Lawrence Technological University, Southfield, MI in 2010. Since 2012, he has been a Ph.D. candidate in mechanical engineering in North Carolina State University, Raleigh. He obtained much experience through working with automotive OEMs and national labs regarding system design and testing. He also had two rounds of internship with Ford Motor Company. Furthermore, he was involved in developing a modular electric generation system with the Future Electric Energy Delivery and Management (FREEDM) systems center. The above research experience helped him develop the dissertation on a new design structure matrix for model-based testing.

ACKNOWLEDGMENTS

I would like to thank Dr. Silverberg and Dr. Pritchard for offering their guidance, encouragement, and support throughout this research. I would also like to express my gratitude to Dr. Buckner, Dr. Saveliev, and Dr. Husain for all their suggestions and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Model-based design (MBD) has been recognized for its power in assisting system development. As system complexity increases so do the risks of potential timeframe overruns in overall project cost (Kalawsky et al., 2013). To be able to adapt to increasing system complexity, the evolving model-based approach is becoming standard practice in systems engineering (SE) (Kalawsky et al., 2013; Sirin, Paredis, Yannou, Coatanea & Landel, 2015; Agarwal, Hamza-Lup & Khoshgoftaar, 2012; Ramos, Ferreira & Barcelo, 2012; Holt et al., 2015). A model is typically structured to have the potential to provide different views and any of the views is a representation of a system (Hilliard, 2000). The model-based approach is to focus on certain aspects of a system while intentionally simplifying other aspects of it. It is desirable that one model is able to explore every aspect of a complex system. However, it is impractical to have such a perfect model. In fact, it is a paradox to expect a model to explore every aspect of a system and save development time and cost because developing the model equals developing the system. Today, the systems engineering community expect to engage a family of models to explore the emergent behavior in a complex system (Kalawsky et al., 2013). Thus, different model builders, especially the ones from specific domains, create their models by choosing different levels of abstraction (Browning, 2016). When these models are shared between the model builders from different domains, different perspectives and understanding from the model builders create information gaps. Information gaps exist not only between the model builders but also among other engineers who are involved in system development. Since they do not have the same level of understanding about the system, its sub-systems, and its component, information gaps are continually being generated throughout the entire system development process. Interestingly, information gaps are rarely studied as part of the system development by researchers. Some researchers have studied system development with a strong emphasis on modeling (Sirin, Paredis, Yannou, Coatanea & Landel, 2015; Agarwal, Hamza-Lup & Khoshgoftaar, 2012); however, these studies only represent plant model development. How plant model development supports the rest of system development has not been discussed. Dagdougui, Minciardi, Ouammi, Robba & Sacile (2010) proposed a dynamic system model for a hybrid power system, which will be controlled by optimal control strategies in the future. Nevertheless, this study does not explicitly demonstrate how plant model development will benefit future control development. In the dissertation, the bridges between information gaps are represented by information flow. Information flow is further extended to information loops including feedforward and feedback. An information loop is a loop of information between at least two development phases, stages, or activities. Information loops are the key to close information gaps.

Besides different perspectives, design changes also result in creating information gaps. Internal design changes are caused by uncertainties that belong to the system development process, while external design changes are requested by clients due to unforeseen events (Chua & Hossain, 2012). Predicting effects of design changes that result from external sources is extremely challenging. Failures to anticipate such impacts can cause significant delays and cost overruns (Chua & Hossain, 2012). In fact, the system development process is a process of bridging information gaps (Hari & Weiss, 2003). However, the time and successfulness of eliminating information gaps was dictated by the system complexity. As the system complexity increases so does the time to close information gaps. Although building a system model with linking information is difficult (Chua & Hossain, 2012), employing model-based approaches in system development has the potential to close the gaps earlier.

The application of MBSE is becoming well understood at the system level; however, there is a lack of research and subsequent industrial application for complex systems (CS) and systems-of-systems (SoS) (Holt et al., 2015). The complex system development (CSD) and systems-of-systems development (SoSD) differs from classic system development. Classic system development heavily focuses on conceptual design which does not weigh testing as much as design (Hari, Weiss, & Zonnenshain, 2004; Pahl, Beitz, Feldhusen & Grote, 2007; Weiss & Hari, 2015). However, as system complexity increases, testing by nature plays a much bigger role in system development. This is because components and sub-systems need to be understood and validated before being integrated into a system. Nowadays, the integration and testing phase typically takes more than 45% of the total development time of a complex system (Boumen, De Jong, Mestrom, Van De Mortel-Fronczak & Rooda, 2009). Design and testing should be treated evenly in system development. Some studies have been done with a design focus to shorten development time and reduce cost (Sirin, Paredis, Yannou, Coatanea & Landel, 2015), but how design and testing activities and results can be utilized together to save time and effort has not been studied on a system level. Since testing has become more and more costly, an ongoing research direction in the software domain is to combine the model-based approach with testing (Takala, Katara & Harty, 2011; Keranen & Raty, 2012). Our study reveals greater potential for the model-based approach in complex system development (CSD) and systems-of-systems development (SoSD) by conducting model-based testing (MBT).

The system development for CS and SoS is a perfect place to employ model-based approaches. Because of the system complexity, the system development is incremental rather than radical. More and more emerging systems such as microgrid, smart grid, and autonomous vehicles are either complex systems and/or SoS and are developed based on an evolution of previous systems (Wang, Valerdi & Fortune, 2010). Due to its incremental nature, the system development is complex, evolutionary, and iterative. It is complex because it consists of development elements across multiple domains. It is evolutionary because it inherits and deviates from the development of any of its sub-systems/components. It is iterative because it employs technologies that are new, but existent (Shenhar & Bonen, 1997). The development usually takes more than two development cycles to finalize the design (Shenhar & Bonen, 1997). Traditionally, systems are developed from a "clear slate" (Wang, Valerdi & Fortune, 2010). This differs from how complex systems and SoS are typically built today since a new complex system or SoS may be adopted from existing systems. In addition, some sub-systems/components of the new complex system or SoS may be modified from prior sub-systems/components. Moreover, the evolution of the complex system or SoS over the system life cycle may result in deleted sub-systems/component from the original system. Although the system development for complex systems and SoS differs from traditional systems, the system development paradox still holds for all the systems. The paradox is that the development starts and critical decisions are made while only partial information is available. As the development progresses, information gaps are closing, but the degree of freedom for making choices diminishes (Karniel & Reich, 2013). When dealing with complex, evolutionary, and iterative system development, the model-based approach plays a bigger role on providing additional information flow in early development stages. If information gaps can be closed earlier, there is a better chance to make correct decisions (Sirin, Paredis, Yannou, Coatanea & Landel, 2015). It needs to be pointed out that not only the system but also its development can be modeled. Employing the model-based approach should organically combine the system-level framework with domain-specific development elements such as development phases, stages, and activities.

There is no doubt that MBSE has the potential of simplifying the complexity of managing CS/SoS development across multiple disciplines (Kalawsky et al., 2013). Given the evolutionary and iterative nature, the question is what specific model-based approach is suited to bridging information gaps in CS/SoS development. Network-based models such as Critical Path Method (CPM), Program Evaluation and Review Technique (PERT), Graphical Evaluation and Review Technique (GERT), Generalized Precedence Relations (GPRs) have very limited capabilities in modeling evolutions and iterations (Cho & Eppinger, 2005). Some researchers have turned to modeling frameworks using Design Structure Matrix (DSM) to represent and analyze the models of complex systems (Smith & Eppinger, 1997a, 1997b; Browing & Eppinger 2002; Browning, 2016). A DSM developed by Steward (1981) is a square matrix with *m* rows and columns, where *m* represents the number of development activities. Each activity is assigned a unique label. The diagonal is marked with the label of the element in the row. An off-diagonal mark signifies the information flow or dependency. The DSM differs from other models in that it focuses on representing information flow or dependency. In fact, information flow and dependency should be considered as two interrelated, but distinct aspects of DSMs.

Smith and Eppinger (1997a, 1997b) developed two DSM-based models to estimate iterative development on time and cost. One is a sequential model assuming only sequential development elements (Smith & Eppinger, 1997a). The other one is a parallel model only focusing on parallel development elements (Smith & Eppinger, 1997b). Browing and Eppinger (2002) developed the first DSM-based discrete-event Monte Carlo simulation model that analyzes development iterations. However, their model does not account for resource constraints. Cho and Eppinger (2005) developed the second generation, DSM-based model that extends the previous DSM-based model to a project management tool. Lévárdy and Browing (2009) proposed a framework using DSM to model the system development as a complex adaptive system. The development elements in the framework self-organize according to simple rules. However, all the above DSM-based models or frameworks only consider dependency. The dependency between development elements is interpreted as either rework probabilities (Smith & Eppinger, 1997a, 1997b; Browning & Eppinger, 2002) or risk indices (Lévárdy & Browning, 2009). Stronger dependency implies higher rework probability or risk index, and vice versa. These models assume that rework or risk of a development element is generated due to the following causes: receiving new information after starting to work with preliminary input; probabilistic failure to meet the established criteria; probabilistic change of inputs when other development elements are reworked (Cho & Eppinger, 2005).

However, neither the rework probability nor risk index can represent information flow. Although dependency may be exchangeable with significance of information, it fails to represent the amount of information flow. For example, development element A is a benchmarking activity, while development element B is a design activity which heavily depends on the outcomes of activity A. It is obvious that both the dependency and the significance of information should be high. To be able to understand the state-of-art in this field, benchmarking only one product from a leading company is certainly not enough. Of course, the quality of the benchmarking outcome is important. However, the number of products being benchmarked also matters. In fact, dependency and information flow are not exchangeable in this case. Unlike many other models that only consider dependency (Smith & Eppinger, 1997a, 1997b; Browing & Eppinger 2002; Lévárdy & Browning, 2009; Browning, 2016), the proposed framework takes into account both the significance and the amount of information. One of the major contributions in this dissertation is the development of a framework that accounts for information flow both qualitatively and quantitatively.

Another major contribution is how the DSM-based framework or model is built. According to (Browning, 2016), there are two traditional ways to build DSMs. One way is to build two DSMs, one row-by-row and the other column-by-column by separately collecting the input and output perspectives of an expert on each development element, and then overlay these. The other way is to integrate several smaller DSMs together. Our approach deviates from the two ways. We form elemental DSMs from different domains with three fundamental types of development elements: identification, design, and testing. After that, we integrate them together to structure the framework. All the elemental DSMs have the same structure and number of information connections while the small DSMs mentioned above usually do not have. Using the elemental DSMs the framework becomes fundamental, flexible, and scalable, which, in turn, makes it broadly applicable to complex systems and SoS. In addition, the order of development elements in the framework, inspired by the Vee model, also follows the top-down identification-design, bottom-up design-testing concept which is not used in (Smith & Eppinger, 1997a, 1997b; Browing & Eppinger 2002; Lévárdy & Browning, 2009; Browning, 2016). The top-down approach serves to ensure that customer requirements are reflected in system definitions and in the solution offered in response to the specification, that each hierarchy in the system matches the one above, that the interface between sub-systems/components is properly planned, and that the integration of all sub-systems/components yields the result expected of the system (Hari & Weiss, 2003). The bottom-up approach ensures that each level in the hierarchy provides to the hierarchy above good sub-systems or components which meet their specification requirements (Hari & Weiss, 2003).

The aim of this dissertation is to propose a new DSM-based framework and demonstrate it through one CSD case study and one SoSD case study. The CSD case study focuses on qualitative analysis, while the SoSD case study emphasizes on quantitative analysis. The framework incorporates model-based development elements into MBSE concepts and bridges information gaps between domains via qualitative and quantitative analysis of information flow. Toward the goal of being broadly applicable, the framework was designed to be fundamental, flexible, and scalable. The framework is also capable of estimating duration of development elements and effects of changes.

CHAPTER 2

METHOD

2.1 Complex System Development and Systems-of-Systems Development

As mentioned in the introduction section, CSD and SoSD are different from classic system development. Since both CSD and SoSD are incremental innovation, it is critical to understand and verify the existing system. It is also important to validate both the new sub-systems/components and the new system. However, verification and validation is challenging in CSD and SoSD because they are complex, evolutionary, and iterative. The development complexity is the degree to which a system or component has a design or implementation that is difficult to understand and verify (Jain, Chandrasekaran, Elias & Cloutier, 2008). Testing plays an important role in closing information gaps. Without having testing activities, it is difficult to know whether the system meets customer expectations. It is also essential to understand the performance of each component before and after integrating it into the system. Furthermore, if the system does not meet some of the expectations, testing activities can provide feedback to refine the system.

CSD and SoSD also relies more on accurately predicting and estimating timeline and costs because of its complex, evolutionary, and iterative nature. For example, a CS or SoS has more interactions of components/sub-system than a simple system. The additional interactions require additional time and money to be understood and implemented into the system. There is no argument that as the expense and time consuming nature of physical experiments increase, it is more desirable to employ model-based design (MBD) techniques (Kernstine, 2013). MBD enables a new dimension by opening a door into the virtual world. MBD provides a unique capability to investigate alternative sets of inputs that are impractical to independently compare in a physical world (Kernstine, 2013). When used in early development stages, MBD helps with requirement capture and design validation and verification purposes (Kalawsky et al., 2013; Wang, Valerdi & Fortune, 2010). A driving force on the cost side is to assure that the formal requirements represent the correct understanding of what the system should and should not do (Alves, Drusinsky, Michael & Shing, 2013). MBD also allows its users to predict the performance of a product without completing the product (Kalawsky et al., 2013). Furthermore, it helps its users to save time and money during the development. However, it does increase the complexity of information flow because every physical component/sub-system in the system may have a virtual representation. Every virtual representation and its development must interact with its corresponding physical component/sub-system via information flow.

Because of the above characteristics, the model-based systematic method needs to be: A. Fundamental; B. Flexible; and C. Scalable. It needs to be fundamental because the foundation of the method should hold in different systems. The modeling characteristics should flow in every vein of the development. Both system-level and component-level modeling activities should be organized organically. In addition, the method should be flexible to adapt to different system architectures or configurations. Finally, the method should be valid when the system is scaled up or down.

2.2 Information Flow

Because of the complexity, the development of CS and SoS is fundamentally iterative (Yassine & Braha, 2003). Due to the maturity of employed technologies, it usually takes at least two development cycles to deliver a system that meets customer expectations (Shenhar & Bonen, 1997). What is flowing through during the entire development is information that dictates time. New information is being discovered during the development and fed into the design to improve performance. If information can be utilized more effectively, a better system is ensured in the sense of meeting customer needs. Thus, understanding the role of information is crucial. Information does not exist if it is not created. What creates information in SE are development elements such as development phases, development stages, development activities. Once a development element is determined, time to complete that element can be estimated. MBD has the potential to create information that has earlier feedback or shorter loops. The shorter information loops enable design flaws to be identified and fixed earlier (Pahl, Beitz, Feldhusen & Grote, 2007). To obtain feedback and refine the system, testing is also essential. When considering utilization of information flow, MBT is expected to be the best candidate because it has all the advantages of the model-based approach and testing. It even extends the capability of either MBD or testing alone. When physical testing is difficult or dangerous to be conducted, MBT can provide reasonable results to help understand the system, which is extremely valuable for complex systems.

2.3 Development Elements

Levels and types of development elements are an important part of the method because they form the foundation of the elemental DSM, which later are used to structure the framework. The levels of development elements are classified from top to bottom as: development phases, development stages, and development activities. A higher-level development element may contain more than one lower-level development element. Development elements can also be classified by types. For example, design phases, design stages, and design activities are one type at three different development levels. We consider identification, design, and testing (IDT) as the three essential types of development elements in CDS and SoS development. The importance of design in system development is self-evident. However, when dealing with information gaps, testing also plays an important role. Having testing results helps verify whether a system meets clients' expectations, generates information, and reduce risk. In an example of SoS, although each sub-system and its operation may remain the same as it is a stand-alone system, its functions have different system-level meanings in an SoS. Thus, it is also essential to understand and validate the performance of each sub-system before and after integrating it into the system. Moreover, if some of clients' expectations are not met, testing results can be useful feedback to refine the SoS. However, testing results may not reveal the best performance of the testing subject simply because the test is not designed properly. In addition, testing also needs guidance so it can be conducted safely and efficiently. Identification is the key to solving the puzzle. Identification such as requirement identification helps not only design but also testing. Identified requirements draw boundaries for a design to increase the design's efficiency. Identification also provides testing criteria. Because the SoS development is evolutionary and iterative, I, D, and T are the three essential types of development elements in SoS.

In the dissertation, identification is to identify or define outcomes, design targets, and requirements for design and testing. Design is to select, develop, and implement components, sub-systems, and systems, which meet predefined requirements from identification. Testing is to test and validate design with predefined requirements. Information transmitted between two development elements is information flow. It needs to be clarified that the design includes building or assembling components, sub-systems, and a system. Literature has shown that other researchers have a type of development elements called build or realization (Hari & Weiss, 2003). The build or realization is included in our design because it is more explicit to represent information flow in this way.

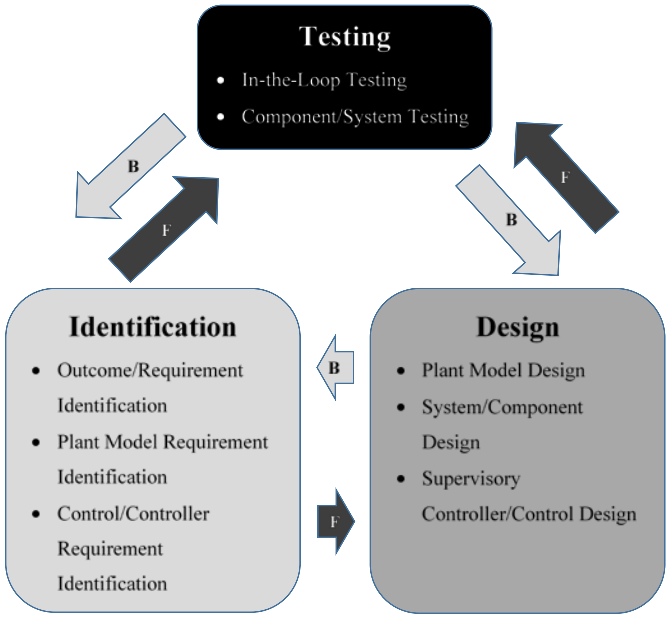

Testing
In-the-Loop Testing
Component/System Testing
B
F
B
F
Identification
Outcome/Requirement Identification
Plant Model Requirement Identification
Control/Controller Requirement Identification
B
F
Design
Plant Model Design
System/Component Design
Supervisory Controller/Control Design

Figure 1. Information flow in system development.

The information flowing between the three types of development elements in the same domain is depicted in Figure 1. Each block in Figure 1 represents one development phase, and each phase may have more than one stage. The arrow with an “F” shows the direction of the feedforward, while the arrow with a “B” indicates the direction of the feedback. To simplify the illustration of information flow utilization, information is assumed to only flow at the development phase level in the same domain. The identification phase generates requirements, which are the life-blood of system development. The first phase of SoS development is driven by collecting, composing, and analyzing system requirements (Jain, Chandrasekaran, Elias & Cloutier, 2008). These requirements are used to guide design. The system, once designed and developed, becomes a testing subject in the testing phase. The requirements are also used to generate pass/fail criteria for the testing phase. Feedforward information includes all of the information from identification to both design and testing and from design to testing. On the other hand, all the information from testing to both design and identification and from design to identification is feedback. Design experience grows organically in the design phase. The experience is used as feedback to refine unrealistic requirements so the system can be developed. The testing result also helps refine both the requirements and the design so the final system will be exactly as it is expected. The final system arrives in the testing phase.

Every flow of information in Figure 1 is unique because it contains information that any other flow of information does not have. The total of six information connections including both feedforward and feedback are defined as complete information connections because the three types of development elements at the same development level can have a maximum of six connections. In addition, the six connections provide necessary information for identification, design, and testing to move further iteratively. It needs to be pointed out that this complete connection holds for the three types at any development level. In addition, the elemental DSM is independent of any of the system attributes such as the number of sub-systems, architecture, and configuration. The independency makes the proposed framework fundamental. The six information connections form the foundation of our framework. Figure 1 illustrates the information flow explicitly when different domains are not involved. However, when illustrating information flow between different domains at different development levels, the DSM as a more powerful representation is needed.

When using DSMs to represent development elements, there are three kinds of relationships between development elements: sequential, parallel, and coupled. The sequential relationship is the relationship between development element 1 and development element 2 in Figure 2. development element 1 provides information for development element 2, while does not require information from development element 2. In other words, development element 2 is dependent on development element 1. The parallel relationship is the relationship between development element 2 and development element 3 in Figure 2. These two development elements are independent on each other. The last relationship is the coupled relationship shown by development element 3 and development element 4 in Figure 2. According to (Browning, 2016), most DSM-based analyses seek to identify an advantageous sequence of sequential development elements by reordering matrix rows and columns. Other researchers overlapped parallel development elements through concurrent engineering (CE). For coupled development elements, neither of the two approaches is effective while both the CSD and SoSD have many coupled development elements. Thus, a new framework is necessary to help analyze information flow between coupled development elements.

		1	*2*	*3*	*4*	⋯	*m*
Development Element *1*	*1*	*1*					
Development Element *2*	*2*	*x*	*2*				
Development Element *3*	*3*			*3*	*x*		
Development Element *4*	*4*			*x*	*4*		
⋮	⋮					⋱	
Development Element *m*	*m*						*m*

Figure 2. Relationships between Development Elements.

2.4 Proposed Framework

The DSM was selected in the study to carry out the role of representing and analyzing information flow between different domains. It was developed by Steward (1981) to both represent and analyze the information flow among interrelated development activities. It has been recognized as an MBSE method since 1991 (Eppinger, 1991). A DSM is a square matrix with m rows and columns, where m represents the number of development elements (see Figure 2). Each element is assigned a unique label. The diagonal is marked with the label of the element in the row. An off-diagonal mark signifies the information flow or dependency. If development element *2* depends on development element *1*, the element *21* in row *2* and column *1* is marked with an x. Otherwise, the element is empty. The DSM differs from other tools in that it focuses on representing information flow (Eppinger, 2001; Campos Silva , Santiago & Silva, 2012). It needs to be mentioned that there are two other well-known design matrices. One is the axiomatic design matrix (Suh, 1998) and the other is the N2 matrix (Bustnay & Ben-Asher, 2005). The axiomatic design matrix focuses on representing relationships between functional requirements and design parameters, while the N^2 matrix is to obtain a system presentation in a simple flow. Neither of them is appropriate for our study because they were not developed for analyzing information flow.

Since the DSM was developed, many researchers have studied it and broadened its capability as a MBSE tool (Eppinger, 1991, 2001; Browning, 2001; Yassine & Braha, 2003; Campos Silva, Santiago & Silva, 2012; Qian & Lin, 2014). Summarizing their work, there are two approaches to DSMs. One is to minimize either the number of feedback connections or the length of feedback. The other is to overlap development activities through concurrent engineering. In our approach, we assume the three basic types of development elements hold at any development level in any domain. In addition, we assume each of the six information connections is essential and unique. Furthermore, the three categories and six connections that form the elemental DSM are also essential and unique. The system development for any complex system can be structured from elemental DSMs. It needs to be pointed out that none of MBT techniques was employed in any of the examples in (Steward, 1981; Eppinger, 2001; Campos Silva, Santiago & Silva, 2012; Qian & Lin, 2014). This approach is fundamentally different than the two traditional approaches because of the roles of identification and testing in terms of bridging information gaps. Instead of reducing feedback connections, this approach takes full advantage of the feedback connections. Both MBD and testing can be incorporated into the system DSM without limiting their capability of bridging information gaps in different domains. The method that uses elemental DSMs to assemble a DSM-based framework has the potential to be fundamental, flexible, and scalable, which is needed for CSD and SoSD. The framework can be used qualitatively or quantitatively due to the needs of applications. The quantitative framework is an extension of the qualitative framework. When it is used quantitatively, the framework does require additional information about the system development.

2.4.1 Qualitative framework. To illustrate the proposed method and framework, a physical system DSM and its corresponding physical component DSM are shown at the top and in the middle of Figure 3, respectively. The first, second, and third rows in these two DSMs represent the identification, design, and testing stages, respectively. Note that both DSMs are elemental DSMs. Assembling the two DSMs together, a DSM for physical system and component development is shown at the bottom of Fig 3. It is noted that the “x’s” in Figure 3 represent horizontal relationships while the “o’s” represent vertical relationships. All of the horizontal relationships are inherited from both the system DSM and the component DSM. The horizontal relationships also represent feedforward and feedback connections at the same level. All the vertical relationships are created when the two DSMs are merged. The vertical relationships represent feedforward and feedback connections between the system level and component level. The two elemental DSMs require complete feedforward and feedback connections. However, the assembled DSM does not require complete feedforward and feedback connections. It is observed that when vertical relationships are involved in DSMs, the requirement for complete connections is broken.

		a	b	c
System Outcome/Requirement Identification	A	A	x	x
System Design	B	x	B	x
System Testing	C	x	x	C

		a	b	c
Component Requirement Identification	A	A	x	x
Component Design	B	x	B	x
Component Testing	C	x	x	C

		a	b	c	d	e	f
System Outcome/Requirement Identification	A	A			x	o	x
Component Requirement Identification	B	o	B	x		x	
Component Design	C		x	C	o	x	
System Design	D	x		o	D		x
Component Testing	E		x	x		E	
System Testing	F	x			x	o	F

Figure 3. Physical system and component DSM assembly.

In addition to physical system/component development, CSD and SoSD have plant model development where virtual components and a system are designed. Furthermore, CSD and SoSD have both physical and virtual control development. The complete DSM for CSD and SoSD is shown in Figure 4. It is observed that the testing stages have more vertical relationships (shown as an "o"). In addition, most of the added vertical relationships are feedback connections. These two observations indicate that CSD may have more feedback resulting from vertical relationships. Also observe that the testing phase including stage k, l, m, and n may generate more feedback than the other two phases. The additional feedback creates new information loops, which means that design corrections can be made earlier. Kalawsky et al. (2013) states that information captured in component models are also related in system models that can relate this information back to system requirements, helping to support system validation. Thus, information gaps between different domains can be bridged before closing them becomes too costly.

		a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	*j*	*k*	*l*	*m*	*n*
System Outcome/Requirement Identification	***A***	***A***					***o***		***x***		***o***	***o***	***o***	***o***	***x***
Component Requirement Identification	***B***	***o***	***B***					***x***	***o***			***o***	***o***	***x***	***o***
Component Model Requirement Identification	***C***		***o***	***C***		***x***						***x***	***o***	***o***	
System Model Requirement Identification	***D***			***o***	***D***		***x***					***o***	***x***		***o***
Component Model Design	***E***			***x***		***E***	***o***					***x***			
System Model Design	***F***				***x***	***o***	***F***						***x***		
Component Design	***G***		***x***					***G***	***o***					***x***	
System Design	***H***	***x***						***o***	***H***						***x***
Controller/Control Requirement Identification	***I***	***o***								***I***	***x***		***x***		***x***
Supervisory Controller/Control Design	***J***									***x***	***J***		***x***		***x***
Component Model In-the-Loop Testing	***K***			***x***		***x***						***K***			
System Model In-the-Loop Testing	***L***				***x***		***x***			***x***	***x***	***o***	***L***		
Component Testing	***M***		***x***					***x***						***M***	
System Testing	***N***	***x***							***x***	***x***	***x***			***o***	***N***

Figure 4. DSM for complex systems at development stage level.

2.4.2 Quantitative Framework. The proposed quantitative framework extends the capability of the qualitative framework. Besides being used to bridge information gaps and identify design flaws in early development stages, it can also be used to estimate the duration of development elements and the effect of design changes. The proposed framework works to close information gaps by treating system development as a process which generates information. This framework essentially tracks information and gaps, along with their priority. Once all the necessary information gaps are closed, the system development is complete. The system development usually requires several cycles to be complete, typically less than 5. The number of cycles is larger in projects that require more aggressive technology advancement or higher tech (Shenhar & Bonen, 1997). Similar to the simulation of a discrete-event model, the first development cycle begins with a single development element and ends once each of the development elements of the DSM has been executed. After each has been executed, new information generated in one development element is then available to all the other dependent development elements, and another cycle begins. The entire development process is only complete once each element has achieved its target level of information or fulfilled its information gap. This target level of information is called the “information target” and is abbreviated mathematically as s_f, therefore they form a vector called the information target vector, $\mathbf{s_f}$**.** The information target is a threshold to determine whether a development element is complete. Larger information targets lead to more iterations. The value of an information target s_f for a development element can be determined by using the minimum value of the significance of information (in the same row in the DSM) and the level of technology used in the development element. For example, development element 1 is dependent on development element 2 with a significance of information at 5. The level of technology is low, which implies 1 iteration is enough to complete the development element (Shenhar & Bonen, 1997). The information target for the development element can be assumed to be any value between 1 and 5 because the development element only needs 1 iteration to reach its information target. The value of an information target for a development element can also be determined by using historical data. Since the development is evolutionary, the information target or gap for the same development element in the previous system development provides a good reference for determining the information target in the current system development. It is noted that other factors such as the level of skills and the level of experience of the team also have influence on determining the value of an information target s_f. To conduct the same development element, an experienced team is expected to have a smaller information gap to bridge than an inexperienced team. Ultimately, the assignment of values is at the discretion of the manager.

Since the proposed model is a system-level framework that simulates an iterative process, every development element (rows in the DSM) in a development model may take more than one iteration. It is important to clarify the different definitions for development cycles and development iterations in the dissertation before going into details about the framework. The development cycle starts when any development element starts; it ends when all of the development elements are complete. The development iteration starts counting when new information flows into a specific development element; it stops counting when the development element is complete. When executed, the framework runs cycle by cycle. However, an iteration for a particular development element may not start until new information flows into the development element; it may end before the entire development is complete.

The framework is an *m*-by-*m* matrix, where m is the number of development elements. The amount of information in the *i*th development cycle for the elements from 1 to m is collected into the information vector:

$$\mathbf{d}(i) = \begin{bmatrix} \mathrm{d}_1(i) \\ \mathrm{d}_2(i) \\ \vdots \\ \mathrm{d}_m(i) \end{bmatrix}$$

where the vector, $\mathbf{d}(i)$**,** represents a count of information. It is dimensionless because it is designed to represent different kinds of information, and is merely a count of pieces of information. The amount of information includes but is not limited to numbers of requirements, number of component/sub-systems/systems specifications, components/sub-systems/systems, numbers of testing results and so on.

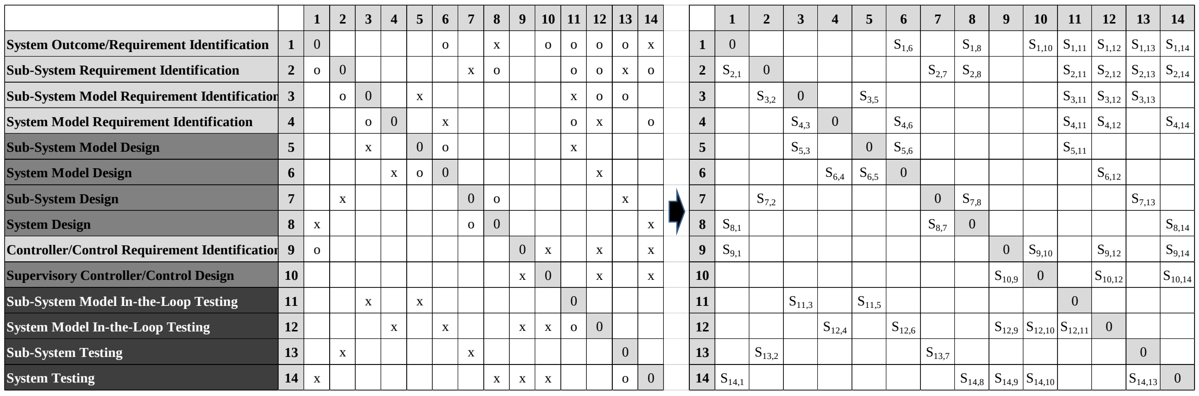
The new information can be used for its dependent development elements. However, some development elements may be significantly affected by the new information, while the others may not. The significance of information dictates how significant a development element is affected by the information generated in its dependent development element(s). The significance of information flowing from one development element to another are collected into the significance matrix:

$$\mathbf{S} = \begin{bmatrix} 0 & \cdots & S_{1,n} \\ \vdots & \ddots & \vdots \\ S_{m,1} & \cdots & 0 \end{bmatrix}$$

where the diagonal terms ($S_{1,1}, S_{2,2}, \dots, S_{m,n}$) are set to be zeros and the non-diagonal terms represent the significance of information between development elements. Similar to the method of determining the rework probability (Smith & Eppinger, 1997a, 1997b; Browning & Eppinger, 2002), the value of significance of information can be determined through conducting surveys with experts. In this dissertation, the values for the significance of information range from 5 to 1. The most significant information is rated at 5. New information is generated from two sources. It can come from internal sources such as other development elements. It can also be produced from external sources. To calculate the amount of new information generated in the ith development cycle, the following equation is proposed:

$$\mathbf{d}(i) = \mathbf{S} \times \mathbf{d}(i-1) + \mathbf{e}(i) \qquad (1)$$

where $\mathbf{S} \times \mathbf{d}(i-1)$ represents information generated internally in the (i-1)th development cycle and the external information vector, $\mathbf{e}(i)$, represents information from external sources in the ith development cycle. The value of external information for a development element is dictated by the information target of the development element. If the information from external sources is enough to completely bridge the gap of the development element, it should equal to the information target. For example, if a company realizes that it can purchase all the required sub-systems in the 3rd development cycle, the information from external sources for development element 7 (sub-system design), $\mathrm{e}_7(3)$, is set equal to the information target of development element 7, $\mathrm{s}_{\mathrm{f}7}$. The information vector before the 1st development cycle, $\mathbf{d}(0)$, is set to be the zero vector because no new information has been generated yet. It is noted that when a development element is complete, there is no information being generated from the development element in the next development cycle and the iteration counting of the development element stops. This is because the development element has already closed its information gap and there is no need to execute it again. For example, if development element m closes its information target in the (i-1)th development cycle, $\mathrm{d}_m(i)$ is set to zero. It also needs to be pointed out that the vector, $\mathbf{e}(i)$, plays an important role in the framework for two reasons. First, the information from external sources, such as clients' expectations, drives the system development. If the clients' expectations are not available, any system development is meaningless. Second, the information continuously flowing into the system development from external sources is very common while the system development is undertaken. The flow of information may cause design changes. As mentioned in the previous section, the design change is one of the two major causes resulting in information gaps; the proposed framework is to bridge information gaps. Thus, having information from external sources in the formula reflects the actual system development. Having new information per development cycle is the first step. The next step is to calculate the sum of information from the beginning until the ith development cycle. The proposed formula is expressed as:



		1	2	3	4	5	6	7	8	9	10	11	12	13	14
System Outcome/Requirement Identification	1	0					o		x		o	o	o	o	x
Sub-System Requirement Identification	2	o	0					x	o			o	o	x	o
Sub-System Model Requirement Identification	3		o	0		x						x	o	o	
System Model Requirement Identification	4			o	0		x					o	x		o
Sub-System Model Design	5			x		0	o					x			
System Model Design	6				x	o	0						x		
Sub-System Design	7		x					0	o					x	
System Design	8	x						o	0						x
Controller/Control Requirement Identification	9	o								0	x		x		x
Supervisory Controller/Control Design	10									x	0		x		x
Sub-System Model In-the-Loop Testing	11			x		x						0			
System Model In-the-Loop Testing	12				x		x			x	x	o	0		
Sub-System Testing	13		x					x						0	
System Testing	14	x							x	x	x			o	0

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	0					$S_{1,6}$		$S_{1,8}$		$S_{1,10}$	$S_{1,11}$	$S_{1,12}$	$S_{1,13}$	$S_{1,14}$
2	$S_{2,1}$	0					$S_{2,7}$	$S_{2,8}$			$S_{2,11}$	$S_{2,12}$	$S_{2,13}$	$S_{2,14}$
3		$S_{3,2}$	0		$S_{3,5}$						$S_{3,11}$	$S_{3,12}$	$S_{3,13}$	
4			$S_{4,3}$	0		$S_{4,6}$					$S_{4,11}$	$S_{4,12}$		$S_{4,14}$
5			$S_{5,3}$		0	$S_{5,6}$					$S_{5,11}$			
6				$S_{6,4}$	$S_{6,5}$	0						$S_{6,12}$		
7		$S_{7,2}$					0	$S_{7,8}$					$S_{7,13}$	
8	$S_{8,1}$						$S_{8,7}$	0						$S_{8,14}$
9	$S_{9,1}$								0	$S_{9,10}$		$S_{9,12}$		$S_{9,14}$
10									$S_{10,9}$	0		$S_{10,12}$		$S_{10,14}$
11			$S_{11,3}$		$S_{11,5}$						0			
12				$S_{12,4}$		$S_{12,6}$			$S_{12,9}$	$S_{12,10}$	$S_{12,11}$	0		
13		$S_{13,2}$					$S_{13,7}$						0	
14	$S_{14,1}$							$S_{14,8}$	$S_{14,9}$	$S_{14,10}$			$S_{14,13}$	0

Figure 5. An extension from the qualitative framework (left) to the quantitative framework (right).

$$\mathbf{s}(i) = \sum_{j=0}^{i-1} \mathbf{d}(j) \tag{2}$$

where $\mathbf{s}(i)$ is the sum of information for all the *m* development elements from the beginning until the *i*th development cycle. If and when an element $\mathrm{s}(i)$ is greater than the information target of that element, the sum of information of that element is set to be equal to the information target. For example, if $\mathrm{s}_3(\mathrm{i}) \geq \mathrm{s}_{\mathrm{f}3}$, $\mathrm{s}_3(\mathrm{i})$ is set to be $\mathrm{s}_{\mathrm{f}3}$.

As mentioned before, the qualitative framework focuses the relationships between development elements while the quantitative framework focuses on how significant the flow of information is. Both the qualitative framework and the quantitative framework are shown in Figure 5. The qualitative framework distinguishes the horizontal and the vertical relationships. The x’s and o’s represent horizontal relationships and vertical relationships in the qualitative framework, respectively. In the quantitative framework, $\mathrm{S}_{\mathrm{m,n}}$ (m,n from 1 to 14) represents significance of information between two development elements. For example, $\mathrm{S}_{14,1}$ represents the significance of information flowing from development element 1 to development 14. The quantitative framework also has the physical system/sub-system development, the virtual system/sub-system development, and the controls development. The order of the development elements follows the top-down identification-design, bottom-up design-testing concept. It is observed that most of the development elements are coupled in Figure 5.

One of the three aims for the proposed framework is to estimate the duration of development elements. This can be done in three steps. The first step is to calculate the number of iterations per development elements. It boils down to when to start and stop each development element (Browning, 2016). There are two start-and-stop scenarios. In one scenario, every development element starts when its prerequisite information is available, it stops when its information target is met. This scenario is very common in large companies where every skill team is dedicated to a distinct development element. In the other scenario, every development element starts when its previous development element is done; it stops when its information target is met. This scenario is preferred by small companies in which man power and resources are limited. The difference between the two scenarios is when the development element starts, which may affect the number of iterations and duration of the development element. In this dissertation, only the first scenario is considered because the second scenario is actually a sequential workflow. Knowing both the information targets and the information generated per iteration, the numbers of iterations for all the development elements from 1 to m until the i^{th} development cycle are collected into the iteration vector:

$$\mathbf{a}(\mathrm{i}) = \begin{bmatrix} \mathrm{a}_1 \\ \mathrm{a}_2 \\ \vdots \\ \mathrm{a}_m \end{bmatrix}$$

where $\mathrm{a}_1, \mathrm{a}_2, \dots, \mathrm{a}_m$ represent the number of iterations for the *1*st, *2*nd,…, *m*th development elements, respectively. The second step is to calculate the duration of all the development elements. We denote the vector for the duration of all the development elements in the framework by:

$$\mathbf{t} = \begin{bmatrix} \mathrm{t}_1 \\ \mathrm{t}_2 \\ \vdots \\ \mathrm{t}_m \end{bmatrix}$$

where each element represents the duration of a development element per iteration. The duration of a development element can be in any time scale such as hours, days, weeks, and years. The final step is to calculate the duration of each development element and the total duration of the development elements. The duration of each development element is a product of the number of iterations and the duration of the development element per iteration. The total duration of all the development elements until the *i*th development cycle is the sum of the duration for all the development elements:

$$t_{\mathrm{f}} = \mathbf{a}(i)^{\mathrm{T}}\mathbf{t} \tag{3}$$

Another aim of the proposed framework is to estimate the effects of changes. As mentioned earlier, the changes can be internal or external. The internal changes affect the information targets, while the external changes result in changes on the values of information from external sources. Since both types of changes have impact on the sum of information, the change in the sum of information, $\Delta\boldsymbol{s}(i)$, reflects the effects of changes. As iterations take place, $\Delta\boldsymbol{s}(i)$

may increase. The last aim of the proposed framework is to incorporate model-based approaches at both the system-level and the domain-specific level to bridge information gaps. The benefits of incorporating these model-based approaches can be evaluated by comparing the total duration of development with and without employing model-based approaches. An example of a 3-by-3 case study, which demonstrates how to assign values to each term can be found in Appendix C.

CHAPTER 3

RESULTS

3.1 Case Study of Complex Systems

The DSM of complex systems described above has allowed our research to look at feedback and its utilization from a different angle. A powertrain design and validation for a series plug-in hybrid electric vehicle to meet its acceleration requirement was conducted as a case study. The powertrain development was selected because it involves interactions between different domains. In addition, the success of the powertrain development depends on when and how information gaps are bridged. The aim of the case study is to demonstrate that the proposed MBT framework bridges the information gaps earlier than other frameworks that do not employ MBT.

The powertrain development of a series plug-in electric hybrid vehicle was undertaken in the EcoCAR2 competition, which is a three-year collegiate student engineering competition focusing on powertrain integration with advanced vehicle technologies. The vehicle architecture is the same as the architecture shown in Figure 6. The front axle of the vehicle is powered by a peak 100 kW Magna electric motor coupled to a single speed gearbox that drives the front wheels. A 33 kW Kubota diesel engine is mechanically coupled with a continuous 37 kW TM4 generator using a herringbone belt drive at a ratio of 2.7:1. The combination of the engine and the generator is called the range extender. An 18.9 kWh lithium ion nanophosphate battery pack allows the vehicle to drive approximately 50 miles in an all-electric mode. A HV junction box couples the battery to the generator and the traction motor and is also connected a 3.3 kW BRUSA on-board charger. The complete EcoCAR has a test mass at 1979 kg. Other vehicle dynamic properties are the same as the stock 2013 Malibu.

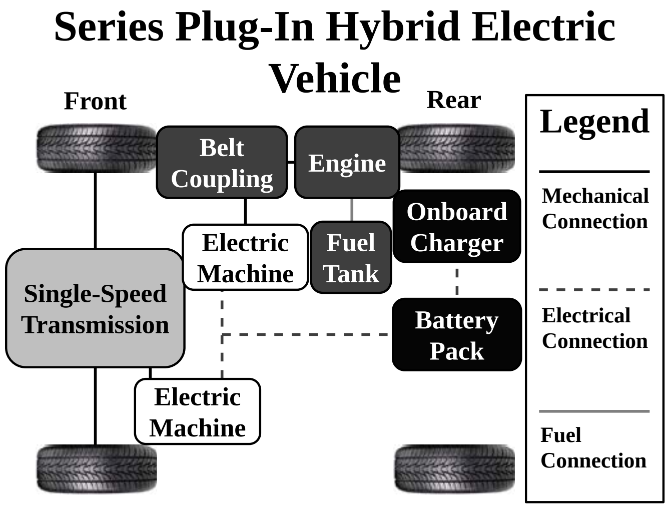


Figure 6. Series plug-in hybrid electric vehicle.

A system model was developed along with vehicle development to help refine the system design and decouple distributed system control development from physical system development. The system model has a component model for each physical component. Since the focus of the case study is powertrain design to meet the acceleration requirement, only the relevant models such as the glider model, the traction motor model, and the battery model are described in the Appendix A.

To succinctly examine the hypothesis, one of the system requirements was selected to show how the powertrain was designed to support system performance (see Table 1). Given the hybrid system architecture and the system requirement, the HV battery pack and the motor are the two powertrain components that have a major impact on the acceleration performance. The battery pack was designed from battery modules. Each module has a specified maximum output of 25 kW. The design question is: how many modules are needed per pack to support the power demand to meet the 0 to 96 km/h acceleration requirement? To answer this question accurately, it is essential to know the maximum power required by the vehicle, which is the maximum power output of the motor. According to Figure 5, if the MBT was not employed, the earliest feedback from the system testing stage requires the completeness of the system. The information gap is the power desired to meet the system requirement from both the motor and the battery pack. To be able to size the battery pack and the motor in terms of power before integrating both the pack and the motor into the vehicle, employing MBT is necessary. In addition, the information loop that consists of the feedback from MBT can bridge the information gap earlier.

The selected system requirement and its interrelated component requirements, the model requirements and the control requirements are listed in Table 1. The first column provides the category and number of the requirements. For example, “s-1” means the requirement is the first system-level requirement. The second column provides the descriptions of the requirements. The third column provides the versions of the requirements. When new information became available, the corresponding requirement was updated. As the feedforward connections show in Figure 5, the system requirement dictates the component requirements. For example, the component requirements in “b-1” describe how much power is needed to accelerate the vehicle to meet the system requirement. The component model requirements and the system model requirements were created to support the component requirements and the system requirement, respectively. The purpose of having modeling activities was to create reasonable results as feedback to help improve the CSD. Since the system controller is responsible for guaranteeing efficient and safe operation of the system, the distributed system control requirement was created to prevent the vehicle acceleration from being interrupted by the operation of any other components in the vehicle.

Three sets of battery and the motor power outputs are shown in Figure 7. The dark and light solid curves are the actual motor and battery power outputs from the same acceleration

Table 1

Requirements

Category/ number	Description	Version
	System Requirement	
s-1	The vehicle shall accelerate from 0 to 96 km/h in 12 seconds	a
	Battery Requirement	
b-1	The battery pack shall support enough power for 0 to 96 km/h acceleration in 12 seconds	a
	The battery pack shall support 116 kW maximum power	b
b-2	The battery pack shall have at least 5 modules	a
	Motor Requirement	
m-1	The motor shall support enough power for 0 to 96 km/h acceleration in 12 seconds	a
	The motor shall support 100 kW peak power	b
	Battery Model Requirement	
b-m-1	The battery model shall have pack open-circuit voltage	a
b-m-2	The battery model shall have pack voltage	a
b-m-3	The battery model shall have losses	a
b-m-4	The battery model shall SOC	a
	Motor Model Requirement	
m-m-1	The motor model shall have power output	a
	System Model Requirement	
s-m-1	The vehicle model shall have vehicle velocity	a
	Distributed System Control Requirement	
s-c-1	The distributed system control shall not sabotage the vehicle acceleration performance	a

event. The two dotted curves were generated by the vehicle model, while the two dashed curves were generated by the vehicle model with a maximum motor power map. It is clear that the battery pack power output can be determined by using the light dotted curve. Its maximum is approximately 116 kW. Thus, version “b” of requirement “b-1” was updated. Given the power value and the maximum power output per module, the battery pack needs to have at least five modules to meet the vehicle acceleration requirement. Similarly, the motor power output can be determined as 100 kW by using the dark dotted curve. Therefore, version “b” of requirement “m-1” was updated. Although the feedback from physical testing shows the motor has a power derating, it does not weaken the necessity to have the feedback from MBT to determine the desired motor output power. The feedback from MBD was helpful in determining the estimated battery pack power and motor power to satisfy the vehicle acceleration requirement. Without this feedback, the battery pack power output would not be able to be determined before the vehicle is built. Although only information loops in component requirement identification stage in Figure 5 have been demonstrated in the dissertation, all the information loops in the case study match the information loops in the framework. Other systems have distinct components from the case study, however, the framework still holds because of its independency on system attributes.

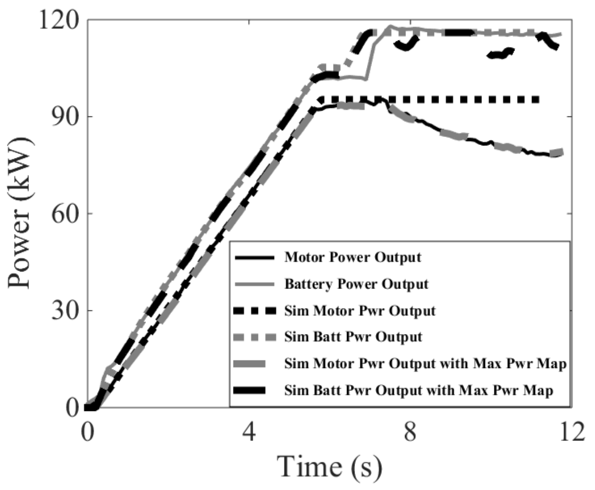


Figure 7. Power output comparison.

3.2 Case Study of Systems-of-Systems

Since the proposed framework was created to deal with evolutionary and iterative system development, a smart micro-grid project was selected as a case study to demonstrate the framework. Besides being evolutionary and iterative, the project was selected also because the development elements in the proposed framework are able to address any development issue and deliver a SoS that meets clients' expectations. The smart micro-grid project was undertaken by the Future Renewable Electric Energy Delivery and Management (FREEDM) Systems center and Pos-En to develop an intelligent micro-grid which can improve grid stability and power reliability for unstable electric power systems. The micro-grid takes in energy from the grid and renewable resources and supports both the grid and the load. The core of the project was to develop an energy generation and storage system (EGSS) evolved from a series plug-in hybrid electric vehicle (PHEV). Although the energy generation and storage system has a similar system architecture, it differs from the PHEV in both its operation and component sizes. Development information including data and experience from the PHEV development was used to accelerate the EGSS development. The EGSS is an evolution of the PHEV. The development of EGSS is also iterative because it took several iterations to finalize the production-ready SoS.

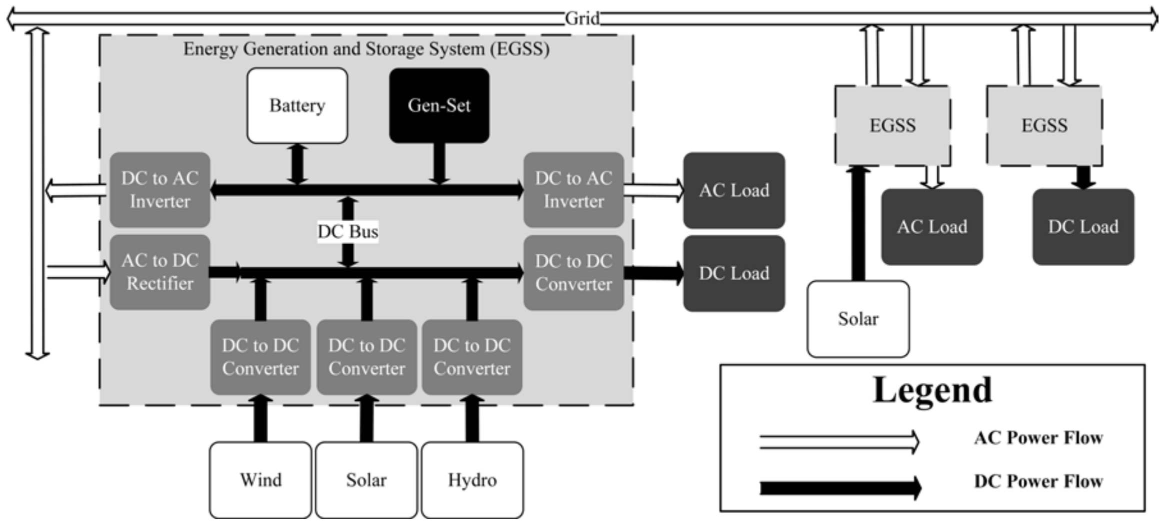


Figure 8. Three-Node Smart Micro-Grid.

A three-node smart micro-grid architecture is shown in Figure 8. Each node represents a EGSS. All the three EGSSs together form a distributed system. One of the three EGSS is shown in an expanded view. The EGSS has a dedicated DC-to-DC converter for each renewable input. It also has one DC-to-AC inverter and one AC-to-DC rectifier. Both of them are connected with the grid. Furthermore, the EGSS has a high voltage (HV) battery pack and a gen-set. At last, it has a DC-to-AC inverter and a DC-to-DC converter connecting to AC loads and DC loads, respectively. All the components are coupled on a common DC bus. The HV battery pack voltage dictates the DC bus voltage. It needs to be mentioned that the architecture of the EGSS depends on its applications. Some can have less renewable energy inputs and can only support either AC loads or DC loads. It also needs to be pointed out that any of these sub-systems can support the load independently. In fact, when the SoS is disassembled, each sub-system can and does operate on its own to support the load.

The DSM-based framework aims to bridge the information gaps earlier with Model-Based approaches, estimate development duration, and predict effects of design changes. It also sheds light on addressing under what conditions model-based approaches are necessary. The case study was conducted to demonstrate the following three hypotheses:

1) The proposed framework can bridge information gaps earlier than frameworks that do not employ Model-Based approaches,

2) The proposed framework can estimate development duration,

3) The proposed framework can predict effects of design changes in terms of the sum of information.

To test the three hypotheses and explicitly explore the advantages of using the proposed framework, user cases need to be defined. We designed five user cases and every user case has two different sets of information targets. Having two different sets of information targets is necessary. Larger information targets mean more development cycles; more development cycles can lead to more iterations for some development elements which implies bigger information gaps. Eight conditions for the user cases are listed in Table 2. The 'Y's and 'N's indicate which condition(s) are applied to which case(s). Conditions 1, 2, 3, and 6 are applied to all the cases. The first user case is a baseline case which does not have any design change

Table 2

USER CASES

Number	Condition	Case 1	Case 2	Case 3	Case 4	Case 5
1	The set of smaller information targets: [70 90 40 70 25 15 40 70 55 40 9 15 20 40]^T	Y	Y	Y	Y	Y
2	The set of larger information targets: [500 682 495 434 182 282 447 443 360 330 71 211 142 35	Y	Y	Y	Y	Y
3	The clients' expectations, $\mathbf{e_1}(1)$, are set to be 1	Y	Y	Y	Y	Y
4	Design change, $\mathbf{e_2}(3)$ is set to be -5	N	Y	N	N	Y
5	Design change, $\mathbf{e_7}(3)$ is set to be -10	N	Y	N	N	Y
6	The rest of $\mathbf{e}$ is set to be 0	Y	Y	Y	Y	Y
7	The case has additional information from the PHEV development: $\mathbf{e_3}(1)$, $\mathbf{e_4}(1)$, $\mathbf{e_5}(1)$, $\mathbf{e_6}(1)$, $\mathbf{e_{11}}(1)$, and $\mathbf{e_{12}}(1)$ are 22, 37, 14, 8, 5, and 9, respectively	N	N	Y	N	N
8	The case only has non-model-based development elements 1, 2, 7, 8, 9, 10, 13, and 14	N	N	N	Y	Y

from external sources. In addition, it does not have helpful data from the PHEV development to accelerate the micro-grid development. The second case has design changes from the client. This case demonstrates how the framework predicts effects of design changes. Conditions 4 and 5 are applied to case 2. The third case is the same as the first case except it has helpful data from the Model-Based development elements in the PHEV development. It demonstrates how the data from the previous system development assists the current system development as the development is evolutionary and iterative. Condition 7 is applied to case 3. Comparing with the baseline case, the fourth case does not have any development element regarding Model-Based approaches. This case demonstrates under what conditions model-based approaches are necessary. Condition 8 is applied to case 4. The last case is the same as the fourth case except it has design changes. The last case demonstrates without the help from Model-Based approaches, how the design changes slow down system development. It also demonstrates the system development paradox by showing a failure example. In the example, critical decisions are made when the information gaps are still big. A sub-system needs to be completely changed in the third development cycle, which heavily affects both development element 2 and 7. The system development is not able to adapt to the design changes fast enough. Thus, the duration of development elements is heavily extended. Condition 4, 5, and 8 are applied to case 5. Although the baseline case is case 1, the actual micro-grid development followed case 3 because the framework has predicted that case 3 has the minimum duration of development elements by closing information gaps earlier. In addition, useful information from model-based development elements in the PHEV development was available to help the micro-grid development.

To be able to execute the framework in simulation, values are also needed to be assigned to the framework. Since the EGSS is an evolution of the PHEV and the EGSS development has the same development elements as the PHEV development, the duration of every development element from the PHEV development was used to generate a set of smaller information targets, $\mathbf{s}_{\mathrm{f}}^{(1)}$, for the micro-grid development. Using the duration of the PHEV development elements was under the assumption that information had been generated until the development was complete. These values of the duration indicate the information gaps for the PHEV development. The set of information targets, $\mathbf{s}_{\mathrm{f}}^{(2)}$, was created using the baseline case to study how the number of cycles affects the hypotheses. $\mathbf{s}_{\mathrm{f}}^{(2)}$ is set to equal $\mathbf{s}(6)$ to from the baseline caseis listed in Figure 9. The other set of information targets was created to study how the number of development cycles affects the hypotheses. The other set is also listed in Figure 9. It needs to be pointed out that the information targets may vary if the system or the skill teams are different. However, the values of information targets used in the case study do not affect the proof of the hypotheses. The significance of information was collected through surveys within the center experts. The values for the significance of information are also shown in Figure 9. The clients' expectations are represented by $\mathbf{e_1}(1)$ for all five cases. In addition, $\mathbf{e_2}(3)$ and $\mathbf{e_7}(3)$ are set to be -5 and -10, respectively for case 2 and case 5. The values indicate 5 sub-system requirements and 10 sub-system specifications need to be revised. At last, to reflect additional information coming from the model-based development elements in the PHEV development, we assume 50% of the information can be used in the micro-grid development. $\mathbf{e_3}(1)$, $\mathbf{e_4}(1)$, $\mathbf{e_5}(1)$, $\mathbf{e_6}(1)$, $\mathbf{e_{11}}(1)$, and $\mathbf{e_{12}}(1)$ are set to be 22, 37, 14, 8, 5, and 9 in case 3, respectively. The duration per iteration for all the development elements are in the last column from left to right in Figure 9. The duration per iteration for all of the model-based development elements is assumed to be one. The duration per iteration for all of the other development elements are assumed to be three times greater than the duration per iteration for all of the model-based development elements. The model-based development elements require less time because models only focus on certain aspects of sub-system and systems. If the modeling takes the same amount of time as building the system, it is meaningless to build a model. The units of the duration are weeks in the case study. It is noted that the duration per iteration may also vary by development elements and team experience. Some development elements may take less time than others. Some teams may conduct the same development element faster than others. Having the above values, the framework can be executed in simulation. An example of calculating the sum of information, the amount of information and the total duration of development elements for case 1 can be found in Appendix B. Since the other four cases only alter parameters and can be solved by following the same procedure in Appendix B, only results from these cases are shown in Results.

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	$s_f^{(1)}$	$s_f^{(2)}$	**t (weeks)**
1	0					1		1		1	3	4	3	5	70	500	3
2	3	0					1	1			3	4	5	3	90	682	3
3		3	0		1						5	4	3		40	495	1
4			3	0		1					4	5		4	70	434	1
5			3		0	2					2				25	182	1
6				3	1	0						2			15	282	1
7		3					0	1					4		40	447	3
8	3						1	0						4	70	443	3
9	2								0	1		2		3	55	360	3
10									3	0		2		3	40	330	3
11			1		1						0				9	71	1
12				1		1			1	1	2	0			15	211	1
13		1					1						0		20	142	3
14	1							2	1	1			1	0	40	354	3

Figure 9. Simulation parameter settings.

To succinctly examine the first hypothesis, results from case 1 and case 4 were compared in Figure 10. Two sets of information targets were used to generate the results. It is needed to be pointed out that case 4 does not have Model-Based development elements. Thus, the results from the Model-Based development elements in case 1 are excluded in Figure 10. The results from case 1 were normalized to 100% so the results with both sets of information targets from case 4 can be compared with the same reference. It is observed that all the development elements in case 4 generated less information than the development elements in case 1. Lower sums of information indicate that case 4 with either set of information targets requires more time to close the information gaps. It is also observed that as the information targets increase, case 1 has more advantages over case 4 in closing information gaps except for development element 8 and 14. As illustrated in both Figure 4 and Figure 9, development elements 8 and 14 do not rely on information from any of the Model-Based development elements. Therefore, unlike any other development elements, these two development elements with larger information targets in case 4 have higher sums of information than they have with smaller information targets. It is concluded that with employing Model-Based approaches the framework can bridge information gaps earlier. In addition, the more iterative the SoS development is, the more valuable employing model-based approaches is.

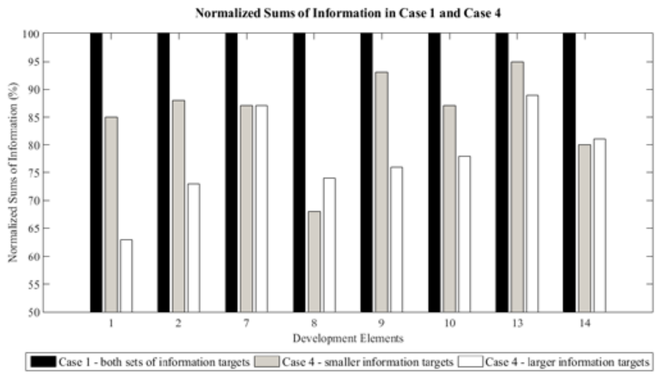


Figure 10. Normalized sums of information in case 1 and case 4.

The total duration and duration of each development element with the set of smaller information targets in all five cases are depicted in Figure 11. The duration of each development element uses the left *y*-axis, while the total duration uses the right *y*-axis. Although the duration per development element varies case by case, case 3 has the minimum total duration of development elements. It is expected that case 3 has less total duration than case 1 because all the Model-Based development elements in case 3 inherit 50 % of information from the Model-Based development elements in the PHEV development (see condition 7 in Table 2). The additional information helped these development elements to reach the information targets faster. Design changes are also expected to slow down the development. As a result, case 2 requires more weeks than case 1. As demonstrated earlier, the development with Model-Based development elements (case 1) took less time than the development without Model-Based development elements (case 4). Case 5 required the most time because it does not have Model-Based development elements while it also has design changes as case 2 does. It is shown that the framework can estimate development duration. The results with the set of larger information gaps are not discussed here because it does not make a difference to the test of the second hypothesis.

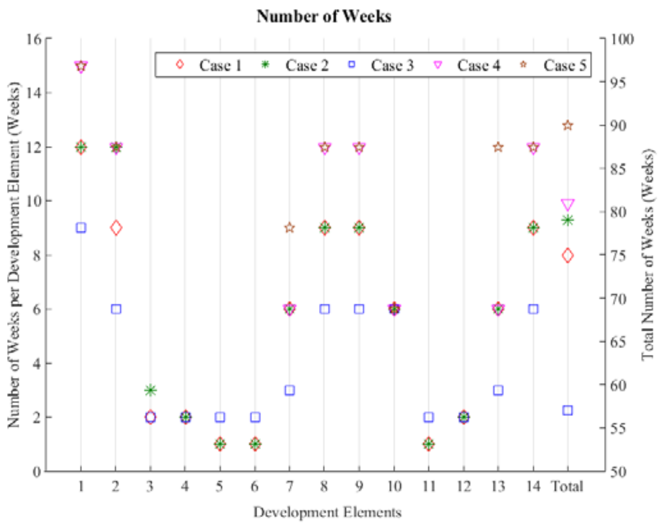


Figure 11. Numbers of weeks with the set of smaller information targets in all five cases.

Results from cases 1, 2, 5 are depicted in Figure 12 to illustrate that the framework can predict effects of design changes. Sums of information per development cycle were captured and depicted in Figure 12. Only case 1 reached the set of smaller information targets in the fifth development cycle and the set of larger information targets in the sixth development cycle, respectively. Case 5 has the lowest sums of information in each of these two development cycles, while case 1 has the highest sums of information. Comparing with case 1, the differences in the sum of information for non-model-based development elements in both

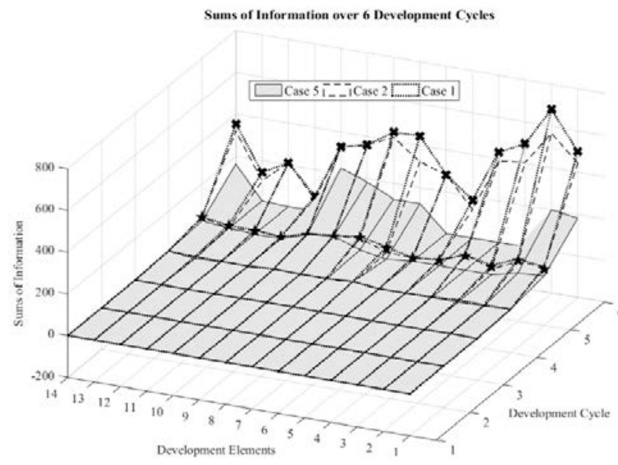


Figure 12. Sums of information over five development cycles.

case 2 and case 5 are listed in Figure 13. Model-based development elements are not listed and compared because case 5 does not have any. It is observed that as the number of development cycles increases from 5 to 6, the differences in the sums of information also increases. Case 5 has larger gaps in both development cycle 5 and 6 because it does not have any model-based development elements to help close the gaps. It is clear that the framework is capable of capturing effects of design changes in every step of the SoS development.

DC 5	Case 2	Case 5	DC 6	Case 2	Case 5
$\Delta s_1(5)$	0	-42	$\Delta s_1(6)$	-55	-318
$\Delta s_2(5)$	-15	-67	$\Delta s_2(6)$	-115	-482
$\Delta s_7(5)$	-25	-59	$\Delta s_7(6)$	-125	-325
$\Delta s_8(5)$	-10	-64	$\Delta s_8(6)$	-25	-321
$\Delta s_9(5)$	0	-4	$\Delta s_9(6)$	0	-178
$\Delta s_{10}(5)$	0	-4	$\Delta s_{10}(6)$	0	-102
$\Delta s_{13}(5)$	-15	-15	$\Delta s_{13}(6)$	-40	-136
$\Delta s_{14}(5)$	0	-18	$\Delta s_{14}(6)$	-35	-193

Figure 13. Comparing with case 1, differences in the sums of information for development elements 1, 2, 7, 8, 9, 10, 13, and 14 in the development cycles 5 and 6 in case 2 and case 5.

CHAPTER 4

CONCLUSION

The aim of this work was to propose and demonstrate a new DSM-based framework, which incorporates MBT to address system development issues for complex, evolutionary, and iterative system development such as CSD and SoSD. Unlike previous DSM-based work, which only focused on using dependency, this method takes advantage of more effectively utilizing the other aspect of DSM, which is information flow. This work built a DSM-based framework using significance and amount of information and tested three hypotheses: the framework bridges the information gaps earlier than other frameworks that do not employ model-based approaches; the framework can estimate duration of development elements; the framework can predict effects of design changes. To test the proposed hypotheses, two case studies were conducted. A powertrain development in a hybrid vehicle was undertaken to represent CSD as one case study. The other case study is an intelligent micro-grid development representing SoSD. The results from the two case studies showed that the proposed framework closed the information gap earlier. In addition, duration of every and all development elements can be estimated before any of them starts. Furthermore, how design changes affect corresponding development elements was illustrated. Last, employing model-based approaches brings in more values on addressing design issues when the system development is more iterative. Because the characteristics of the framework are fundamental, flexible, and scalable, the framework is broadly applicable to CSD and SoSD. Given the evolutionary and iterative nature of CSD and SoSD, the framework also has potential in improving information reuse.

Data from the automotive industry will be used to improve the accuracy of the framework. The simulation results will be used to refine the framework and make it more realistic to real-world large scale system development. Beside the experiments in the industry, adaptive significance matrix with Monte Carlo approach is also considered. Selected parameters are givens a distribution and then the combinations are run to see the histories. At last, the capacity of the quantitative framework will be extended to be able to estimate effects of internal design changes.

REFERENCES

Agarwal, A., Hamza-Lup, G. L., & Khoshgoftaar, T. M. (2012). A System-Level Modeling Methodology for Performance-Driven Component Selection in Multicore Architectures. *IEEE Systems Journal*, *6*(2), 317-328.

Alves, M. C. B., Drusinsky, D., Michael, J. B., & Shing, M. T. (2013). End-to-end formal specification, validation, and verification process: A case study of space flight software. *IEEE Systems Journal*, *7*(4), 632-641.

Browning, T. R. (2001). Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *IEEE Transactions on Engineering management*, *48*(3), 292-306.

Browning, T. R., & Eppinger, S. D. (2002). Modeling impacts of process architecture on cost and schedule risk in product development. *IEEE transactions on engineering management*, *49*(4), 428-442.

Bustnay, T., & Ben-Asher, J. Z. (2005). How many systems are there? using the N2 method for systems partitioning. *Systems engineering*, *8*(2), 109-118.

Boumen, R., De Jong, I. S., Mestrom, J. M. G., Van De Mortel-Fronczak, J. M., & Rooda, J. E. (2009). Integration and test sequencing for complex systems. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *39*(1), 177-187.

Browning, T. R. (2016). Design structure matrix extensions and innovations: a survey and new opportunities. *IEEE Transactions on Engineering Management*, *63*(1), 27-52.

Cho, S. H., & Eppinger, S. D. (2005). A simulation-based process model for managing complex design projects. *IEEE Transactions on engineering management*, *52*(3), 316-328.

Chua, D. K., & Hossain, M. A. (2012). Predicting change propagation and impact on design schedule due to external changes. *IEEE Transactions on Engineering Management*, *59*(3), 483-493.

Dagdougui, H., Minciardi, R., Ouammi, A., Robba, M., & Sacile, R. (2010). A dynamic decision model for the real-time control of hybrid renewable energy production systems. *IEEE Systems Journal*, *4*(3), 323-333.

Eppinger, S. D. (1991). Model-based approaches to managing concurrent engineering. *Journal of Engineering Design*, *2*(4), 283-290.

Eppinger, S. D. (2001). Innovation at the Speed of Information. *Harvard business review*, *79*(1), 149-158.

Estefan, J. A. (2007). Survey of model-based systems engineering (MBSE) methodologies. *Incose MBSE Focus Group*, *25*(8).

Hilliard, R. (2000). Ieee-std-1471-2000 recommended practice for architectural description of software-intensive systems. *IEEE, http://standards. ieee. org*, *12*, 16-20.

Hari, A., & Weiss, M. P. (2003). Analysis of Risk and Time to Market During the Conceptual Design of New Systems. In *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*.

Hari, A., Weiss, M. P., & Zonnenshain, A. (2004). ICDM-an integrated methodology for the conceptual design of new systems. In *SETE 2004: Focussing on Project Success;*

Conference Proceedings; 8-10 November 2004 (p. 35). Systems Engineering Society of Australia.

Holt, J., Perry, S., Payne, R., Bryans, J., Hallerstede, S., & Hansen, F. O. (2015). A model-based approach for requirements engineering for systems of systems. *IEEE Systems Journal*, *9*(1), 252-262.

Jain, R., Chandrasekaran, A., Elias, G., & Cloutier, R. (2008). Exploring the impact of systems architecture and systems requirements on systems integration complexity. *IEEE Systems Journal*, *2*(2), 209-223.

Keranen, J. S., & Raty, T. D. (2012). Model-based testing of embedded systems in hardware in the loop environment. *IET Software*, *6*(4), 364-376.

Karniel, A., & Reich, Y. (2013). Multi-level modelling and simulation of new product development processes. *Journal of Engineering Design*, *24*(3), 185-210.

Kernstine, K. H. (2013). Inadequacies of Traditional Exploration Methods in Systems-of-Systems Simulations. *IEEE Systems Journal*, *7*(4), 528-536.

Kalawsky, R. S., O'Brien, J., Chong, S., Wong, C., Jia, H., Pan, H., & Moore, P. R. (2013). Bridging the gaps in a model-based system engineering workflow by encompassing hardware-in-the-loop simulation. *IEEE Systems Journal*, *7*(4), 593-605.

Lévárdy, V., & Browning, T. R. (2009). An adaptive process model to support product development project management. *IEEE Transactions on Engineering Management*, *56*(4), 600-620.

Pahl, G., & Beitz, W. (2013). *Engineering design: a systematic approach*. Springer Science & Business Media.

Qian, Y., & Lin, J. (2014). Organizing interrelated activities in complex product development. *IEEE Transactions on Engineering Management*, *61*(2), 298-309.

Ramos, A. L., Ferreira, J. V., & Barceló, J. (2012). Model-based systems engineering: An emerging approach for modern systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(1), 101-111.

Steward, D. V. (1981). The design structure system: A method for managing the design of complex systems. *IEEE transactions on Engineering Management*, (3), 71-74.

Shenhar, A. J., & Bonen, Z. (1997). The new taxonomy of systems: Toward an adaptive systems engineering framework. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *27*(2), 137-145.

Smith, R. P., & Eppinger, S. D. (1997). Identifying controlling features of engineering design iteration. *Management Science*, *43*(3), 276-293.

Smith, R. P., & Eppinger, S. D. (1997). A predictive model of sequential iteration in engineering design. *Management Science*, *43*(8), 1104-1120.

Suh, N. P. (1998). Axiomatic design theory for systems. *Research in engineering design*, *10*(4), 189-209.

Silva, D. D. C., Santiago, L. P., & Silva, P. M. S. (2012). Impact of premature information transfer on cost and development time of projects. *IEEE Transactions on Engineering Management*, *59*(4), 692-704.

Sirin, G., Paredis, C. J., Yannou, B., Coatanéa, E., & Landel, E. (2015). A model identity card to support simulation model development process in a collaborative multidisciplinary design environment. *IEEE Systems Journal*, *9*(4), 1151-1162.

Takala, T., Katara, M., & Harty, J. (2011, March). Experiences of system-level model-based GUI testing of an Android application. In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation* (pp. 377-386). IEEE.

Wang, G., Valerdi, R., & Fortune, J. (2010). Reuse in systems engineering. *IEEE Systems Journal*, *4*(3), 376-384.

Weiss, M. P., & Hari, A. (2015). Extension of the Pahl & Beitz systematic method for conceptual design of a new product. *Procedia CIRP*, *36*, 254-260.

Yassine, A., & Braha, D. (2003). Complex concurrent(engineering and the design structure matrix method. *Concurrent Engineering*, *11*(3), 165-176.

Zhu, D., Pritchard, E., & Silverberg, L. (2016). A new system development framework driven by a model-based testing approach bridged by information flow. Manuscript submitted for publication.

Weicker, P. (2013). *A systems approach to Lithium-Ion battery management*. Artech house.

Reid, C. (2007). Lithium Iron Phosphate Cell Performance Evaluations for Lunar Extravehicular Activities.

Väyrynen, A., & Salminen, J. (2012). Lithium ion battery production. *The Journal of Chemical Thermodynamics*, 46, 80-85.

A123 Systems. (2011). *Nanophosphate High Power Lithium Ion Cell ANR26660M1-B*. Retrieved from http://www.a123systems.com/Collateral/Documents/English-US/A123%20Systems%20ANR26650%20Data%20Sheet.pdf.

APPENDICES

Appendix A

The glider model or the vehicle dynamic model is a model used to find the power and energy needed at the wheels for a specified vehicle in a given drive cycle. The base equation for the glider model is:

$$m_i \frac{dv}{dt} = F_{tr} - F_{rr} - F_{aero} - F_{grade} \tag{4}$$

$$m_i \frac{dv}{dt} = F_{tr} - mgC_{rr} - \frac{1}{2}\rho C_d A_f v^2 - mg\sin\alpha \tag{5}$$

where m_i is the inertial mass of the vehicle in kg, v is the vehicle velocity in m/s, F_{tr} is the traction force at the wheels in N, F_{rr} is the rolling resistance in N, F_{aero} is the aerodynamic drag in N, F_{grade} is the force generated by a grade in N, m is the vehicle mass in kg, and i is the rotating inertia factor. If vehicle properties such as vehicle mass and front area are substituted into the base equation, the equation becomes:

$$m_i \frac{dv}{dt} = F_{tr} - mgC_{rr} - \frac{1}{2}\rho C_d A_f v^2 - mg\sin\alpha \tag{6}$$

where g is the gravitational constant in N/kg, C_{rr} is the coefficient of rolling resistance, ρ is air density in kg/m^3, C_d is the coefficient of drag, A_f is the frontal area of the vehicle in m^2, and α is road angle in degree.

The black box modeling technique was used to create the motor model because feedback from testing stages was expected. When feedback is available for model refinement, the black box modeling technique can update the model to provide enough fidelity in a short period. The inputs of the model are accelerator pedal position, brake pedal position, and vehicle velocity. The outputs of the model are motor torque, motor speed, motor power input, motor power output, and motor losses. An efficiency map was used in the model. The data includes motor voltage input, motor current input, motor torque output, and motor speed output. The efficiencies were calculated using the following equation:

$$\eta = \frac{V_{motor} I_{motor}}{T_{motor} \omega_{motor}} 100\% \tag{7}$$

where η is the motor efficiency in %, V_{motor} is the voltage input of the motor in V, I_{motor} is the current input of the motor in A, T_{motor} is the torque output of the motor in Nm, and ω_{motor} is the speed output of the motor in rpm. In addition, a maximum output power map was created and used to reflect the maximum power output of the motor at every speed point.

A simple internal resistance model was used to calculate cell open-circuit voltage, battery pack open-circuit voltage, cell voltage, battery pack voltage, battery losses, and state-of-charge (SOC). The model also helped size the battery capacity and the battery maximum power, which is desired for the battery pack design. The model is described below:

$$P_{batt} = 1000(R_{in} I_{batt}^2 + V_{oc} I_{batt}) \tag{8}$$

$$SOC = \frac{\int I_{batt} V_{oc}}{3600000 C_{batt}} 100\% \tag{9}$$

where P_{batt} is the battery pack power in kW, R_{in} is the battery pack internal resistance in ohms, I_{batt} is the battery pack current in A, V_{oc} is the battery pack open-circuit voltage in V, SOC is the state of charge in %, and C_{batt} is the battery pack capacity in kWh. Since cell testing data is expected, the non-linear relationship between the SOC and the cell voltage is described using a look-up table. The SOC-voltage data was collected at 100 ms sample rate by charging the cell at 0.006 A. Because the cell minimum capacity is 19.6 Ah, which means its 1 C-rate is 20 A, the charging current used in the cell testing does not affect the accuracy of the voltage at every SOC point.

Appendix B

To calculate the sum of information, the amount of information and the total duration of development elements, the significance matrix, the two sets of information targets, and the duration vector from Figure 9 are used. In addition, the information from external sources in Figure A1 is also used. As shown in Figure A1, only $\mathbf{e_1}(1)$ equals to one. The rest of $\mathbf{e}_{\boldsymbol{m}}(i)$, where *m* is from 1 to 14 and *i* is from 1 to 5 or 6 depending on which set of information targets is used, is zero in the baseline case. Having the above, the following procedure needs to be taken to calculate the sum of information and the amount of information until the set of smaller information target is reached. All the **s**, **d**, **a** vectors generated from the procedure for all the development elements are given in Figure A2. The shaded cells indicate the values of the sums of information change from the previous development cycle.

	e(1)	**e(2)**	**⋯**	**e(6)**
DE 1	1	0	⋯	0
DE 2	0	0	⋯	0
DE 3	0	0	⋯	0
DE 4	0	0	⋯	0
DE 5	0	0	⋯	0
DE 6	0	0	⋯	0
DE 7	0	0	⋯	0
DE 8	0	0	⋯	0
DE 9	0	0	⋯	0
DE 10	0	0	⋯	0
DE 11	0	0	⋯	0
DE 12	0	0	⋯	0
DE 13	0	0	⋯	0
DE 14	0	0	⋯	0

Figure A1. Information matrix in the baseline case.

	s(1)	d(1)	s(2)	d(2)	s(3)	d(3)	s(4)	d(4)	s(5)	a(5)	a(6)
DE 1	0	1	1	0	1	8	9	70	70	4	5
DE 2	0	0	0	3	3	6	9	87	90	3	4
DE 3	0	0	0	0	0	9	9	35	40	2	3
DE 4	0	0	0	0	0	4	4	69	70	2	3
DE 5	0	0	0	0	0	0	0	27	25	1	2
DE 6	0	0	0	0	0	0	0	16	15	1	2
DE 7	0	0	0	0	0	12	12	34	40	2	3
DE 8	0	0	0	3	3	4	7	68	70	3	4
DE 9	0	0	0	2	2	3	5	53	55	3	4
DE 10	0	0	0	0	0	9	9	37	40	2	3
DE 11	0	0	0	0	0	0	0	9	9	1	2
DE 12	0	0	0	0	0	2	2	16	15	2	3
DE 13	0	0	0	0	0	3	3	18	20	2	3
DE 14	0	0	0	1	1	8	9	31	40	3	4

Figure A2. Simulation results for the baseline case.

1. Calculate $\mathbf{s}(1)$. As stated earlier, there is no new information generated before the development starts. Thus, $\mathbf{d}(0)$ is assumed to be a zero vector. With that assumption, $\mathbf{s}(1)$ is also a zero vector as shown in Figure A1.
2. Calculate $\mathbf{d}(1)$. Knowing $\mathbf{d}(0)$ and $\mathbf{e}(1)$, $\mathbf{d}(1)$ can be calculated using (1).
3. Calculate $\mathbf{s}(2)$. Once having $\mathbf{d}(1)$, $\mathbf{s}(2)$ can also be determined using (2).
4. Calculate $\mathbf{d}(2)$. Knowing $\mathbf{S}$, $\mathbf{d}(1)$, and $\mathbf{e}(2)$, $\mathbf{d}(2)$ can be calculated using (1). Since only development element 1 has a change from $\mathbf{s}(1)$ to $\mathbf{s}(2)$, according to the significance matrix, the new information from development element 1 affects development element 2, 8, 9, and 14. S times $\mathbf{d}(1)$ adding $\mathbf{e}(2)$ gives the vector of $\mathbf{d}(2)$ in Figure A2.
5. Calculate $\mathbf{s}(3)$. $\mathbf{s}(3)$ is the sum of $\mathbf{d}(0)$ or $\mathbf{s}(1)$, $\mathbf{d}(1)$, and $\mathbf{d}(2)$.

6. Continue calculating the sum of information and the amount of information in the following development cycles until every element in the set of smaller information target is reached. In the baseline case, the set of smaller information targets is reached in development cycle 5.

The number of iteration for any development element can be calculated by counting how many iterations the development element takes from it start having new information until it reaches its information target. In the example of development element 12, it took two iterations to reach its information target 18. Therefore, its number of iterations for the set of smaller information targets is 2 which is also shown in Figure A1. Having the number of iterations, the total duration of development elements can be determined using (3). It is 75 weeks. Similarly, the sum of information, the amount of information, and the total duration of development elements for the set of larger information targets can be obtained using the same procedure. The total duration of development elements for the set of larger information targets is 105 weeks.

Appendix C

To demonstrate how to use the quantitative framework, consider the following scenario. A Technical Lead oversees the development of a lithium iron phosphate battery sub-system. The development is divided into three tasks: identification, design, and testing. Each task needs to be completed by a dedicated team under the Technical Lead. Based on his/her experience and other considerations, the Technical Lead understands how one task influences the other. Requirements for power rating, maximum voltage, and minimum voltage need to be identified. Once they are generated, they are used to draw boundaries in the design to produce a sub-system which can meet maximum voltage, minimum voltage, and power rating specifications. The requirements are also used to generate pass/fail criteria for voltage tests and a power rating test. The sub-system, as an outcome of the design, must meet the specifications in Table A1. The requirement for maximum and minimum voltage and the power rating are updated after each design iteration. Testing results, such as maximum voltage, minimum voltage, and power rating are also used to refine the voltage and power rating requirements. Furthermore, testing results are checked with the specifications.

The technical objective is to develop a sub-system which can meet requirements (acceptable range of values) for power rating greater than 55 kW, maximum voltage less than 360 V, and minimum voltage greater than 190 V. The power rating requirement is given by the system engineer. The voltage requirements are identified during the development. The voltage range of the sub-system, which depends upon the number of cells in series, is a function of the individual cell voltage range (Weicker, 2013). For most of lithium iron phosphate batteries, the cell voltage typically ranges from 2.5 V to 3.65 V due to the chemistry selection

(Reid, 2007; A123 Systems, 2011). The cell supplier provides battery cells with the cell specifications in (A123 Systems, 2011). To be able to meet the requirements, the sub-system is designed to have 100 cells per string and each string has 3 cells in parallel. The sub-system has 300 cells in total with a configuration of 100S3P. The cell maximum voltage (3.5 V) and minimum voltage (2.0 V) (A123 Systems, 2011) times the number of cells per string (100 cells) gives the maximum and minimum voltage values in Table A1, respectively. The discharge pulse power (200 W) (A123 Systems, 2011) times the total number of cells (300 cells) gives the power rating in Table A1. The outcome of the development is a sub-system that meets the specifications (specific values) shown in Table A1**.**

Table A1

BATTERY SUB-SYSTEM SPECIFICATIONS

Parameter	Value	Unit
Battery Sub-System Max Voltage	350	V
Battery Sub-System Min Voltage	200	V
Battery Sub-System Power	60	kW

Without the help from model-based approaches, the Technical Lead relies heavily on testing when updating the requirements and validating the voltage and power rating of the sub-system. He/she is also interested in knowing under what conditions the sub-system matches the design specifications. If any of the specifications is not met, he/she needs to know why.

The DSM-based quantitative framework can be used to guide each cycle of the battery sub-system development. The amount of new information generated in development cycle *i* can be characterized:

$$\mathbf{d}(i) = \mathbf{S}\times\mathbf{d}(i-1) + \mathbf{e}(i) \tag{10}$$

where $\mathbf{S}$ is the significance matrix and $\mathbf{e}(i)$ is the information from external sources in development cycle *i*. The sum of information for all the tasks before development cycle *i* can be characterized:

$$\mathbf{s}(i) = \sum_{j=0}^{i-1} \mathbf{d}(j) \tag{11}$$

The total duration of time of all the tasks can be obtain by:

$$\mathrm{t_f} = \mathbf{a}(i)^{\mathrm{T}}\mathbf{t} \tag{12}$$

where $\mathbf{a}(i)$ is the number of iterations vector and $\mathbf{t}$ is the duration of iteration-time vector.

To use the DSM-based quantitative framework, parameter values need to be assigned to $\mathbf{S}$, $\mathbf{s}_\mathrm{f}$, $\mathbf{t}$, $\mathbf{e}(1)$, $\mathbf{e}(2)$, $\mathbf{e}(3)$, $\mathbf{d}(0)$, and $\mathbf{s}(1)$ by the Technical Lead, where $\mathbf{s}_\mathrm{f}$ is the information target vector. All the parameters are determined based on the Technical Lead's experience with previous plug-in hybrid electric vehicle (PHEV) system development, the level of the required skills, and the level of the team experience. Other factors such as historical data may be considered for some of these parameters. The values of significance matrix **S** are shown in Figure A3.

Task Description	**Task ID**	**1**	**2**	**3**
Identification of Requirements	**1**	0	1	5
Design	**2**	3	0	4
Testing	**3**	1	1	0

Figure A3. The simplified quantitative framework for the battery sub-system. The significance matrix is included.

The Technical Lead understands that the significance of information has different levels ranging from 5 to 0: most significant, significant, mild, weak, weakest, and no significance at all. In this method, the diagonal terms ($S_{1,1}= S_{2,2}=S_{3,3}=0$) are set to zeros because no information flows across different tasks.

$$\mathbf{S} = \begin{bmatrix} 0 & 1 & 5 \\ 3 & 0 & 4 \\ 1 & 1 & 0 \end{bmatrix} \tag{13}$$

The requirements generated in the identification have mild influence on the design ($S_{2,1}$=3) because they set boundaries for the design and improve design efficiency, whereas the specifications generated in the design also have the weakest influence on the identification ($S_{1,2}$=1) because the specifications need to be validated first. The requirements have the weakest influence on the testing ($S_{3,1}$=1) because testing can be conducted regardless how specific the requirements are, whereas the testing results have the most significant influence on the identification of requirements ($S_{1,3}$=5) because they are strong evidence to correct unrealistic requirements. The specifications have the weakest influence on the testing ($S_{3,2}$=1) because the success of conducting voltage tests and power tests does not rely on the specifications whereas the testing results have significant influence on the design ($S_{2,3}$=4) because they can help refine unrealistic design.

The information target, s_f, is a threshold to determine whether a task is complete (see (14)). Larger information targets lead to more iterations. The value of an information target for a task would often be determined from historical data. The duration of a task is the time to close information gaps for that task. Larger information gaps generally require more time than smaller information gaps. For example, testing 100 battery cells requires 10 times more time than testing 10 battery cells with the same procedure and set of tests. It is reasonable to assume that the duration required to close an information gap can represent the amount of information required. The Technical Lead knows the duration of the identification, design, and testing tasks based on the previous PHEV development is 3, 3, and 4 weeks, respectively. Thus, these values are put into $\mathbf{s}_{\mathrm{f}}$ representing the information targets.

$$\mathbf{s}_{\mathrm{f}} = \begin{bmatrix} 3 \\ 3 \\ 4 \end{bmatrix} \tag{14}$$

Next, the Technical Lead selects the values of the duration-per-iteration vector, $\mathbf{t}$ (see (15)). Before introducing how the values are selected, it is important to explain how the numbers of development cycles and iterations are counted. The development cycle starts when any task starts; it ends when all the tasks are complete. The iteration starts counting when new information flows into a specific task; it stops counting when the task is complete. When executed, the framework runs cycle by cycle. However, an iteration for a particular task does not start until new information flows into the task; it may end before the entire development is complete. Given the readiness of the identification team, the design team, and the testing team, the duration for each team to run 1 iteration is 3 weeks. However, in general, the unit could be milestones, hours, days, months, etc.

$$\mathbf{t} = \begin{bmatrix} 3 \\ 3 \\ 3 \end{bmatrix} \tag{15}$$

The value of external information for a task is dictated by the information target of the task. If the information from external sources is enough to completely bridge the gap of the task, it should equal the information target. In the case study, $\boldsymbol{e}(1)$, $\boldsymbol{e}(2)$, and $\boldsymbol{e}(3)$ are the zero vector except for $e_1(1)$ (see (16), (17), and (18)). The Technical Lead expects 3 requirements from the identification ($\mathrm{s}_{\mathrm{f}1} = 3$). $e_1(1)$ is set to 1 representing the initial power rating requirement given by the system engineer. $\mathbf{s}(1)$ and $\mathbf{d}(0)$ are zero vectors because no information has been generated before the development started (see (19) and (20)).

$$\mathbf{e}(1) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{16}$$

$$\mathbf{e}(2) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{17}$$

$$\mathbf{e}(3) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{18}$$

$$\mathbf{s}(1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{19}$$

$$\mathbf{d}(0) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{20}$$

Once having $\mathbf{S}$, $\mathbf{s}_{\mathrm{f}}$, $\mathbf{t}$, $\mathbf{e}(1)$, $\mathbf{e}(2)$, $\mathbf{e}(3)$, $\mathbf{s}(1)$, and $\mathbf{d}(0)$, simulation can be carried out to calculate $\mathbf{d}(i)$, $\mathbf{s}(i)$, and t_{f} until the identification, design, and testing reaches their information targets. With the chosen parameter settings, the development takes 3 cycles to reach $\mathbf{s}_{\mathrm{f}}$. In development cycle 1, a requirement, $\mathrm{d}_1(1)$, regarding the power rating of the battery sub-system is transferred from the system engineer to the Technical Lead. No new information is generated in the design and the testing, so $s_1(2) = 1$ representing the transferred requirement. In development cycle 2, the requirement is used to design a prototype battery sub-system. The prototype has three specifications ($d_2(2) = 3$): the maximum voltage is 350 V, the minimum voltage is 200 V, and the power rating is 60 kW (these three values are created based on the sub-system configuration and the data in (A123 Systems, 2011)). The prototype is also visually inspected in the testing ($d_3(2) = 1$). Before development cycle 3, the sum of information vector, $\mathbf{s}(3)$, has 1 requirement, 3 specifications, and 1 test in the identification, design, and testing, respectively. It is noted that the design has reached its information target ($s_{f2} = 3$ in gray) so no new information is generated from the design to improve the battery development ($d_2(3)$ is set to be zero). In development cycle 3, the three specifications are validated in the testing ($d_3(3) = 3$). The sub-system meets its specifications. In addition, the testing results are used to refine the power rating requirement and add two new requirements for maximum and minimum voltage. The values of the voltage requirements are selected based on the validated specifications with a 10 V tolerance. It is noted that the testing results generate eight requirements ($d_1(3) = 8$), however, only the two voltage requirements are needed. Finally, the identification, design, and testing all reach their information targets by the end of development cycle 3 ($\mathbf{s}(4) = s_f$ in gray).

Cycle 1:

$$\mathbf{d}(1) = \begin{bmatrix}1\\0\\0\end{bmatrix} \Rightarrow \begin{bmatrix}1\\0\\0\end{bmatrix},\ \mathbf{s}(2) = \mathbf{d}(0) + \mathbf{d}(1) = \begin{bmatrix}1\\0\\0\end{bmatrix} \Rightarrow \begin{bmatrix}1\\0\\0\end{bmatrix}$$

Cycle 2:

$$\mathbf{d}(2) = \begin{bmatrix}0\\3\\1\end{bmatrix} \Rightarrow \begin{bmatrix}0\\3\\1\end{bmatrix},\ \boldsymbol{s}(3) = \mathbf{d}(0) + \mathbf{d}(1) + \mathbf{d}(2) = \begin{bmatrix}1\\3\\1\end{bmatrix} \Rightarrow \begin{bmatrix}1\\3\\1\end{bmatrix}$$

Cycle 3:

$$\mathbf{d}(3) = \begin{bmatrix}8\\4\\3\end{bmatrix} \Rightarrow \begin{bmatrix}8\\0\\3\end{bmatrix},\ \boldsymbol{s}(4) = \mathbf{d}(0) + \mathbf{d}(1) + \mathbf{d}(2) + \mathbf{d}(3) = \begin{bmatrix}9\\3\\4\end{bmatrix} \Rightarrow \begin{bmatrix}3\\3\\4\end{bmatrix} \qquad (21)$$

$$\mathbf{a}(4) = \begin{bmatrix}3\\1\\2\end{bmatrix}$$

$$\mathrm{t_f} = \mathbf{a}(4)^{\mathrm{T}}\mathbf{t} = [3\ 1\ 2]\begin{bmatrix}3\\3\\3\end{bmatrix} = 9 + 3 + 6 = 18$$

All of the **d** vectors and the **s** vectors are summarized in Table A2 with the number-of-iteration vector. The shaded cells indicate the iteration(s) for the identification, design, and testing. Counting the number of shaded cells for each task gives the number-of-iteration vector, $\mathbf{a}(4)$. The total duration of all of the tasks before development cycle 4 is 18 weeks.

Table A2

Summary

Task	s(1)	d(1)	s(2)	d(2)	s(3)	d(3)	s(4)	a(4)
Identification	0	1	1	0	1	8	3	3
Design	0	0	0	3	3	0	3	1
Testing	0	0	0	1	1	3	4	2

The Technical Lead learns from going through the simulation that the duration of the identification, design, and testing and the total duration are 9 weeks, 3 weeks, 6 weeks, and 18 weeks, respectively. In addition, he/she learns how many iterations are needed for each task (see $\mathbf{a}(4)$). Furthermore, he/she recognizes how significant one task is over another during the development.