SANDIA REPORT

SAND2013-10742
Unlimited Release
Printed December 2013

Integrating the Human Element into the Systems Engineering Process and MBSE Methodology

Michael S. Tadros

Prepared by Michael S. Tadros
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

NOTICE: This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government, nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represent that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof, or any of their contractors or subcontractors. The views and opinions expressed herein do not necessarily state or reflect those of the United States Government, any agency thereof, or any of their contractors.

Printed in the United States of America. This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone:	(865) 576-8401
Facsimile:	(865) 576-5728
E-Mail:	reports@adonis.osti.gov
Online ordering:	http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd.
Springfield, VA 22161

Telephone:	(800) 553-6847
Facsimile:	(703) 605-6900
E-Mail:	orders@ntis.fedworld.gov
Online order:	http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

SAND2013-10742
Unlimited Release
Printed December 2013

Integrating the Human Element into the Systems Engineering Process and MBSE Methodology

Michael S. Tadros
Sandia National Laboratories
P.O. Box 5800
Albuquerque, New Mexico 87185-MS0933

Abstract

In response to the challenges related to the increasing size and complexity of systems, organizations have recognized the need to integrate human considerations in the beginning stages of systems development. Human Systems Integration (HSI) seeks to accomplish this objective by incorporating human factors within systems engineering (SE) processes and methodologies, which is the focus of this paper. A representative set of HSI methods from multiple sources are organized, analyzed, and mapped to the systems engineering Vee-model. These methods are then consolidated and evaluated against the SE process and Models-Based Systems Engineering (MBSE) methodology to determine where and how they could integrate within systems development activities in the form of specific enhancements. Overall conclusions based on these evaluations are presented and future research areas are proposed.

There is no such thing as an ‘unmanned system’.

-Brigadier General Don D. Flickinger,
Director of Human Factors, USAF

ACKNOWLEDGMENTS

I would like to give a special thanks to my mentor, Reverend Father Symeon-Anthony (David) Beck, who’s guidance was invaluable in completing this work and without whom I would have never discovered systems engineering.

CONTENTS

1. Introduction 11

2. Methods 13

2.1 Data Collection 13

2.2 Data Analysis 13

2.3 Standards and Tools 13

3. Literature Review 15

4. Vee-model process 17

5. Mapping Analysis 21

5.1. Scoping the Vee-model 21

5.2. HSI Methods Mapped to the Vee-model 21

6. Discussion 23

6.1. Stakeholder Requirements Definition 23

6.1.1 Unique Methods 25

6.1.1.1 Interviewing 25

6.1.1.1.1 Critical Incident Study 25

6.1.1.2 Reviewing 26

6.1.1.3 Mission Analysis 26

6.1.2 Unique Methods Summary 26

6.1.3 Models-Based Systems Engineering Support 27

6.1.3.1 MBSE Stakeholder Requirements Definition Summary 27

6.1.3.2 Evaluation of HSI Methods Against MBSE 31

6.1.4 Models-Based Systems Engineering Support Summary 32

6.2. Requirements Analysis 34

6.2.1 Unique Methods 36

6.2.1.1 Analysis of Similar Systems 36

6.2.1.2 Requirements Analysis 36

6.2.1.2.1 Functional Flow Block Diagramming 37

6.2.2 Unique Methods Summary 38

6.2.3 Models-Based Systems Engineering Support 39

6.2.3.1 MBSE System Technical Requirements Summary 39

6.2.3.2 Evaluation of HSI Methods Against MBSE 41

6.2.4 Models-Based Systems Engineering Support Summary 42

6.3. Architectural Design: Functional Architecting 44

6.3.1 Unique Methods 46

6.3.1.1 Task Analysis for Knowledge Description (TAKD) 46

6.3.1.2 Functional Analysis 47

6.3.1.3 Timeline Analysis 50

6.3.1.4 Simulation 51

6.3.1.5 Action/Information Analysis 51

6.3.1.6 Operational Sequence Diagram (OSD) 52

6.3.2 Unique Methods Summary 53

6.3.3 Models-Based Systems Engineering Support 54

6.3.3.1 MBSE Functional Architectures Summary 54

6.3.3.2 Evaluation of HSI Methods Against MBSE 58
6.3.4 Models-Based Systems Engineering Support Summary 60
6.4. Architectural Design: Physical Architecting 61
6.4.1 Unique Methods 62
6.4.1.1 Function Allocation 62
6.4.1.2 Task Description and Analysis 63
6.4.1.3 Performance, Workload and Manning Level Estimation 64
6.4.1.3.1 Failure Analysis 65
6.4.2 Unique Methods Summary 66
6.4.3 Models-Based Systems Engineering Support 67
6.4.3.1 MBSE Physical Architectures Summary 67
6.4.3.2 Evaluation of HSI Methods Against MBSE 71
6.4.4 Models-Based Systems Engineering Support Summary 72

7. Conclusions 75

8. References 77

Appendix A: Mapping Analysis Diagrams 81

Appendix B: MBSE Stakeholder Requirements Definition 87

Appendix C: MBSE Functional Architectures 93

Distribution 95

FIGURES

Figure 1: The Vee-model (INCOSE SE Handbook, v 3.2.2) 17
Figure 2: Context Diagram for Stakeholder Requirements Definition Process (INCOSE SE Handbook, v 3.2.2) 23
Figure 3: Unique and Complimentary HSI Methods 25
Figure 4: Strategic context graphical example (UML or SysML *use case* diagram) (Beck, 2011). 28
Figure 5: Generic strategic context graphical example (DoDAF *CV-1* diagram; in SysML use *req* diagrams) (Beck, 2011). 28
Figure 6: A SysML Package Diagram (pkg) is used to develop simple views and viewpoints (Friedenthal, et al., Chapter 4, 2011). 30
Figure 7: Context Diagram for Requirements Analysis Process (INCOSE SE Handbook, v 3.2.2) 34
Figure 8: Vee-Model Step Two HSI Methods 36
Figure 9: A zero-order functional flow block diagram (Chapanis 1996, Ch. 4). 37
Figure 10: A first-order functional flow block diagram (Chapanis 1996, Ch. 4). 38
Figure 11: Generic ECD (Beck, 2011). 39
Figure 12: System Scenarios vs. ECD (Beck, 2011). 40
Figure 13: Elaborated Context Diagram (ECD) with design constraints (Beck, 2011). 41
Figure 14: Context Diagram for Architectural Design Process (INCOSE SE Handbook, v 3.2.2) 44
Figure 15: Vee-Model Step Three HSI Methods 46
Figure 16: Third-level Functional Flow Block Diagram (DoD 1999, §8.3.5.2.3). 47

Figure 17: Flow Process Chart (sample) (DoD 1999, §8.3.7.1). 48
Figure 18: Example from a hierarchical task analysis (Stammers, et. al, 1990)........................... 49
Figure 19: System decomposition using multiple IDEF$_0$ graphs (DoD 1999, §8.3.11.1.4). 49
Figure 20: Timeline plot (sample) (DoD 1999, §8.3.10.2)... 50
Figure 21: Example of an Action-Information Analysis: Filling an Automobile Gas Tank (Chapanis, 1996, Ch. 4). 52
Figure 22: Operational Sequence Diagram (sample) (DoD 1999, §8.3.6.2). 53
Figure 23: N^2 diagram features (Beck, 2011). 56
Figure 24: Example SysML diagram comparable to N^2 diagram (Beck, 2011). 56
Figure 25: Example logical hierarchy (Beck, 2011). 57
Figure 26: Example logical interface diagram (Beck, 2011). 58
Figure 27: Vee-Model Step Four HSI Methods... 62
Figure 28: Human-Machine Allocation Decision Space (Sanders and McCormick 1993, Ch. 22) 63
Figure 29: Workload analysis profile (sample) (DoD 1999, §8.3.13.1). 64
Figure 30: Fault tree (sample) (Chapanis 1996, Ch. 4)... 66
Figure 31: Generic physical architecture (sample) (Beck, 2011). 68
Figure 32: Trade tree for NASA Manned Mars Mission (Beck, 2011). 70
Figure 33: Physical architecture represented in a block diagram (Beck, 2011). 71
Figure 34: Use Case Diagram (uc) of a human operating a vehicle. 87
Figure 35: High-level human-system interaction captured using a Sequence Diagram (sd)........ 88
Figure 36: Detailed model of human-system interaction captured using a sequence diagram (sd). 89
Figure 37: The *Drive Vehicle* use case elaborated using a State Machine (stm) diagram. 90
Figure 38: An Activity Diagram (act) with "swim lanes" distinguishes the actions performed by the human and those performed by the vehicle. 91
Figure 39: A contextual depiction of the vehicle's environment is captured using an Internal Block Definition Diagram (ibd)... 92
Figure 40: Block definition diagram (bdd) of the Automobile Domain (Friedenthal, et al., Chapter 4, 2011)... 93

TABLES

Table 1: Vee-model Step One: HSI Methods Comparison.. 24
Table 2: Vee-model Step Two: HSI Methods Comparison... 35
Table 3: Vee-model Step Three: HSI Methods Comparison... 45
Table 4: Vee-model Step Four: HSI Methods Comparison... 61
Table 5: Morphological Box for Hammer (Beck, 2011). 69

NOMENCLATURE

BPMN	Business Process Modeling Notation
CONOPS	Concept of Operations
DoD	United States Department of Defense
EPC	Event-driven Process Chain
FFBD	Functional Flow Block Diagram
GPA	Generic Physical Architecture
HFE	Human Factors Engineering
HSI	Human Systems Integration
HTA	Hierarchical Task Analysis
I/O	Input(s)/Output(s)
INCOSE	International Council on Systems Engineering
MA	Morphological Analysis
MBSE	Models-Based Systems Engineering
OMG	Object Management Group
OOA	Object-Oriented Analysis
OOSEM	Object-Oriented Systems Engineering Methodology
OSD	Operational Sequence Diagram
SE	Systems Engineering
SME	Subject Matter Expert
SRD	Systems Requirements Document
SysML	Systems Modeling Language
T&E	Test and Evaluation
TAG	Task Action Grammar
TAKD	Task Analysis for Knowledge Description
TBR	Technical Basis Report
TCSD	Task-Centered System Design
TDD	Technical Description Document
USAF	United States Air Force

1. INTRODUCTION

In an era where the complexity and size of systems has grown exponentially, rigorous competition has demanded the need for continuous innovation, and volatile economic climates push organizations to produce more with less, the integration of humans within systems poses an ever-growing challenge. The disconnect between people and technology is well documented[1] and has led to several major disasters such as Three Mile Island, Chernobyl, and more recently, an incident involving the Patriot Missile radar system in which two friendly military aircraft were shot down.[2] These examples have shown that the integration of human considerations must begin in the early stages of the systems development life cycle and has led to multiple initiatives by organizations[3] to accomplish this objective through what is known as Human Systems Integration (HSI).

Currently, there exist multiple definitions of HSI. According to Booher (2003) HSI is the process of integrating people, technology, and an organization at a systems level, with full consideration given to the human requirements of the user. More specifically, the human, together with hardware and software, is considered an element of the system and identifies the necessary interactions between these elements to realize successful systems. "Human" according to the International Council on Systems Engineering (INCOSE), includes all personnel who interact with the system in any capacity (e.g., system owners, users, operators, maintainers, etc.) (INCOSE SE Handbook, v 3.2.2).

Although at the time not formally recognized as "HSI", the human factors community predates most of the more recent initiatives including the failure incidents referenced above. In his keynote presentation at the 1957 National Symposium on Human Factors in Systems Engineering, Brigadier General Don D. Flickinger, Director of Human Factors, USAF, stated, "[T]he impact of man's characteristics must be taken into account in the design of the equipment if the system is to possess maximum probability of achieving the goal for which it was designed in the first place" (Flickinger, 1957). While the objective of this paper is not to provide an argument for what HSI is or why it is needed[4], it should be recognized that since at least the time Br. Gen. Flickinger gave this keynote, the HSI domain is especially interested the integration of human factors in the *design*, rather than a post-solution application of human factors to the system.

In order to successfully integrate human factors within a project development life cycle, it is recommended that HSI take place within the context of systems engineering (SE). Given its formal, structured approach, HSI practitioners can work within the processes and methodologies that SE provides to ensure successful integration of the human element into systems (Muralidhar,

[1] (Madni, 2009).

[2] (Defense Science Board, 2005).

[3] Some examples include Army MANPRINT, Navy Human Performance Center, DoD HSI Initiative, U.S. Coast Guard HSI Program, NASA Human Factors Research and Technology Division, Human Factors Integration Defence Technology Centre.

[4] Many sources exist that discuss these topics, a few of which include: (Booher, 2003), (Chapanis, 1996), (Sanders and McCormick, 1993), (Stammers, et. al, 1990).

2008). More specifically, the SE *process,* as defined by one or more standards, is the means by which this is accomplished. As Burns, et al., (2005) states, a key tenet of the systems engineering process is that system-level optimization requires trade-off analyses and integration to be conducted within and across all system elements. Similarly, the potential payoff of HSI cannot be realized by providing stovepipe support, but must live within the systems engineering process.

To date, there exist a large number of HSI methods[1] that span a wide variety of competencies[2]. While many of these methods would be beneficial for defining and analyzing human factors during systems development, the organization of these methods into a coherent framework is lacking. Madni (2009) asserts that the HSI domain is fragmented and the challenge for HSI practitioners is to mature and consolidate HSI practices for "prime-time" use. In order for HSI to be effective within the context of systems engineering, the related methods must be assimilated to the SE process. The first objective of this paper is to provide a framework to reach this goal.

In addition to integrating HSI methods within the SE process, attention must also be given to Models-Based Systems Engineering (MBSE). Like other engineering disciplines, such as mechanical and electrical engineering, systems engineering is transitioning from a document-based approach to a models-based approach (Friedenthal, et al., 2012). The application of the MBSE approach results in improved design quality, efficient reuse of development artifacts, and effective communications within the development team, leading to increased productivity, quality, and reduced risk. Within the HSI context, INCOSE provides examples of such benefits, "validated HSI modeling and simulation also can pay large dividends early in the development process…" and, "decisions about whether or not to automate certain functions can be evaluated with modeling and simulation to identify and reduce risk, or at least scope the types and levels of risk involved" (INCOSE SE Handbook, v 3.2.2). Therefore, the second objective of this paper is to identify how HSI methods could enhance the MBSE methodology to enable a more thorough and effectual consideration of human factors.

The structure of this paper is as follows: research methods are described with a summarization of the literature reviewed; the steps of the SE process as illustrated by the Vee-model are briefly described; the analysis used to map the identified HSI methods to the SE process (Vee-model) is illustrated; a discussion section is presented in which the SE process and the MBSE method are summarized, unique HSI methods are described and evaluated against the SE process and MBSE, and recommended integration 'enhancements' are proposed; and finally, concluding remarks are provided along with references and appendices.

[1] The U.S. Department of Defense alone identifies forty-five methods in its MIL-HDBK-46855A, *Human Engineering Program Process and Procedures* (DoD, 1999).

[2] The U.S. Air Force identifies twenty-eight HSI competency areas in its *Air Force Human Systems Integration Handbook* (USAF 2009, Table 1).

2. METHODS

2.1 Data Collection

The first step in researching this topic began by conducting a literature review, which initially required identifying a few topical papers in the published domain. Based on the references found in these papers, citation searches were conducted to identify additional relevant sources. These citation "branches" were followed to a level of depth that was thorough, yet manageable. Overall, fifty-seven resources were cited.

2.2 Data Analysis

Sources were then reviewed for content, specifically to identify HSI methods. The list of sources was then culled down to six sources that, although not exhaustive, were representative of the published HSI methods. In total, ninety-seven methods were identified. As a means of organization, these methods were then mapped to the SE Vee-model. Given this large volume of methods, the scope of the analysis was limited to include only the HSI methods that mapped to the first half of the Vee-model, *Stakeholder Requirements Definition* through *Architectural Design*. The HSI methods mapped to each respective step were then analyzed across sources. Methods that showed strong similarities with regards to activities, inputs, outputs, and tools, were grouped into unique, cross-cutting methods and evaluated as one method.

Comparative analyses of the HSI methods against the SE process and MBSE methodology were conducted. The results of these analyses were then evaluated to determine whether general conclusions or suggested enhancements could be drawn. Where appropriate, a summary of enhancements is provided in each section.

2.3 Standards and Tools

The Systems Engineering process used in the HSI methods evaluation was based on the technical processes found in ISO/IEC 15288 and INCOSE Systems Engineering Handbook, v.3.2.2. It should be noted that due to a limitation of resources, two different versions ISO/IEC 15288 were used based on their availability: 2002 and 2008. Specifically, ISO/IEC 15288:2002 was fully available and ISO/IEC 15288:2008 was partially available through a secondary source, the INCOSE SE Handbook.

Text and figures make reference to diagrams from the Object Management Group (OMG) Systems Modeling Language (SysML) Version 1.3.

Resource citation information was collected and organized using the Zotero (www.zotero.org) research tool.

3. LITERATURE REVIEW

As mentioned in §2 *Methods*, beginning with a few HSI sources, citation searches were conducted to identify a representative set of HSI methods that could be evaluated against the SE process and MBSE methodology.

A common theme in the literature reveals the need for a stronger representation of human factors within systems. More specifically, human capabilities and their implications on the design, deployment, operation, and maintenance of systems have not been explicitly addressed in systems engineering and acquisition lifecycles (Madni, 2009). In response to this, many organizations began human factors initiatives, the most notable among them being the DoD's recent push to incorporate the human factors discipline into systems engineering. DoD 5000.2-R, *Mandatory Procedures for Major Defense Acquisition Programs*, states, "Human factors engineering requirements shall be established to develop effective human-machine interfaces…[and] the capabilities and limitations of the operator…shall be identified prior to program initiation…and refined during the development process" (DoD, 1999). Madni (2009) asserts that this led to the creation of the new multidisciplinary field of Human Systems Integration, which is intended to remedy the disconnect between human factors and systems development.

Since that time, the definition and scope of HSI has remained unclear. As McGovern, et al. (2008) explain, this is largely due to the fact that it is an amalgamation of pre-existing diverse technical disciplines (domains) with established vocabularies. Despite this fact, the discipline of HSI has matured, and, as Wilson (1990) states, "after many years of discussion of its nomenclature, direction and so on…the methods we use are more the focus of attention". It is quite clear that the HSI domain covers a wide range of methods, techniques, and tools that apply to the research, design, and evaluation of human-centered systems. For example, Booher (2003) identifies fifty subcategories, the U.S. Department of Defense (DoD) (1999) identifies forty-five, and the U.S. Air Force (USAF) (2009) identifies twenty-eight. Overall, nearly a hundred HSI methods were initially identified for evaluation for this paper.

However, while there is no lack of methods and associated tools available for capturing the human element in systems development, there is still the need for organization among these methods and an accurate mapping to the SE process. The literature shows and Madni (2009) asserts that the HSI domain is fragmented and needs to first be internally integrated in order to assess the impact of human performance on system cost and schedule. Most sources organize methods around human factors categories and do not directly relate them to the SE process. For example, both Wilson and Corlett (1990) and Booher (2003) group methods based on areas of human factors, such as task analysis, accident and incident analyses, assessment and design of the physical workplace, etc. Some sources, such as Stanton, et. al (2012), provide a general indication of where the methods could be applied in the development process (i.e., during the design process and not after system production). There are a few sources that specifically point out where certain HSI methods should be applied. For example, Sanders and McCormick (1993) define a serial set of development "stages" and identify applicable HSI methods for each "stage." McNeely, et al. (2006) take this one step further and show where certain HSI methods should integrate into a standardized SE process (e.g., IEEE 1220). Still, even with a more formal organization, these latter sources lack the required analysis to determine *how* HSI methods can be integrated within systems engineering, and by extension models-based systems engineering (MBSE), in order to reach the desired state of a systems development process that fully integrates human considerations.

4. VEE-MODEL PROCESS

As stated previously, the SE process described in this paper is based on the technical processes found in ISO/IEC 15288:2002 and INCOSE Systems Engineering Handbook, v.3.2.2 standards. Although various[1] life cycle models could be used to portray this process, the Vee-model adequately captures and illustrates the verification and validation needed between a system in its last stages of development to its requirements. In addition, the SE Process is arranged and depicted as a "Vee" in both ISO/IEC 15288:2002, Annex C, Figure C.1 and INCOSE Systems Engineering Handbook, v. 3.2.2, Chapter 3, Figure 3-4.

As shown in Figure 1, the left side of the "Vee" follows the waterfall model in that it decomposes and defines the user or stakeholder requirements in terms directly applicable to the system to be designed, then allocates the requirements to functional and then physical architectures down to the component level, generally developing derived requirements in the process. The right side of the "Vee" illustrates how components are integrated back up to the system level, and explicitly relates testing to the verification and validation of requirements at each successive level.

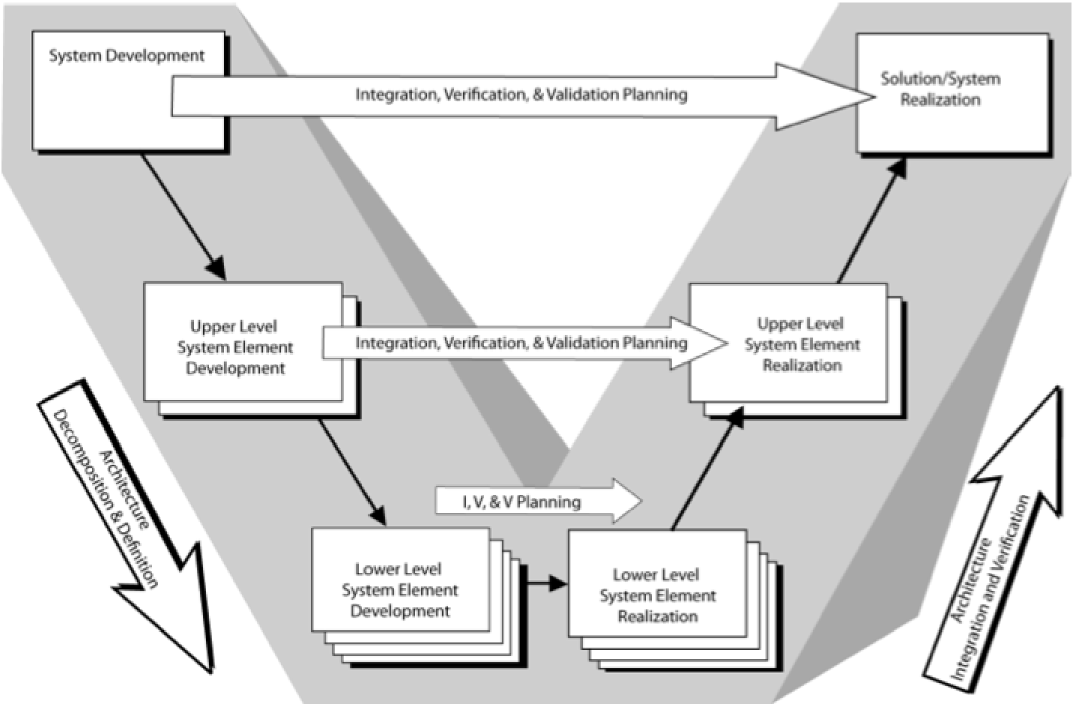

System Development
Integration, Verification, & Validation Planning
Solution/System Realization
Upper Level System Element Development
Integration, Verification, & Validation Planning
Upper Level System Element Realization
Architecture Decomposition & Definition
I, V, & V Planning
Lower Level System Element Development
Lower Level System Element Realization
Architecture Integration and Verification

Figure 1: The Vee-model (INCOSE SE Handbook, v 3.2.2).

The Vee-model steps can be summarized as follows:

The first step (Step One) of the VEE-model, *Stakeholder Requirements Definition*, is intended to develop an understanding of the user (mission) needs and reconcile it to the "input" requirements

[1] INCOSE SE Handbook, v.3.2.2, §3.3, *Life-Cycle Stages*

obtained from the user. Understanding the system interfaces and establishing a validation plan are also core activities during this step.

The next step (Step Two), *Requirements Analysis,* identifies and defines the required functionality of the system based on the user (mission) needs and system interfaces from Step One. A "black box" model is established which clearly identifies the technical inputs and outputs of the system. A verification plan is established defining the test and evaluation (T&E) activities to be used to demonstrate that the system meets the defined measures of performance (MOPs).

Step Three, *Architectural Design: Functional Architecting,* is primarily concerned with developing a system functional design, which identifies and describes "all functions to be provided, along with the associated quantitative requirements to be met by each [functional] subsystem in order that the prescribed system-level capabilities can in fact be achieved" (Kossiakoff and Sweet, 2003). During this step, a "white box" model is created that establishes the context functions must operate within (i.e., interfaces and performance requirements). Justification in the form of trade studies or trade-off analysis must be provided for the selected functional and logical solution(s).

During Step Four, *Architectural Design: Physical Architecting,* the system design undergoes "translation into hardware and software components, and the integration of these components into the total system" (Kossiakoff and Sweet, 2003). This physical architecture establishes the context components must operate within. In addition, trade study results provide justification for the selected technologies and physical architecture(s) selected, and component verification objectives are established to ensure specifications are met.

The next step (Step Five), *Implementation*, develops detailed design definition of the components for the selected physical architecture(s), which will then be implemented (manufacture or procurement).

Step Six, *Integration*, integrates the components and verifies that the result can satisfy the system functional verification objectives (from Step Three) based on the interactions of the said components.

The next step (Step Seven), *Verification,* demonstrates that the design conforms to the system verification objectives established during Step Two with the use of the integrated and functionally-verified system produced under Step Six.

Step Eight, *Transition*, establishes a capability to provide services in the operational environment. This usually includes installing the verified system with relevant enabling systems and is used at each level in the system structure. Note that while *Transition* precedes *Validation* as listed herein, it does not necessitate that it occurs before *Validation*. Simply, this is how it was ordered within the INCOSE and ISO/IEC standards.

The final step (Step Nine), *Validation*, ensures that the system, which met the performance objectives of Step Seven, also satisfies the validation objectives (user mission requirements) established in Step One.

These nine steps comprise the VEE-model, which is applied iteratively and recursively to each development stage of the system life cycle. However, in addition to the steps defined by the INCOSE and ISO/IEC standards, an additional sub-step, *Technical Basis Reports*, has been added to the model given its application to the *Physical Architecting* step within the context of MBSE. Technical Basis Reports explore various physical implementation options that populate the decision space with the intent of identifying an optimal alternative, thus informing the physical architecting activity.

5. MAPPING ANALYSIS

5.1. Scoping the Vee-model

Since the focus of this research was on methods for modeling human considerations, which typically takes place during the Concept, Development and Production Stages of the SE life cycle, emphasis was placed on the technical processes that have the highest level of effort during these stages:

1. Stakeholder Requirements Definition
2. Requirements Analysis
3. Architectural Design
4. Implementation
5. Integration
6. Verification
7. Transition
8. Validation

HSI methods were individually analyzed within the context of the SE process steps in order to determine where they fit within the Vee-model. Specifically, method activities, inputs, outputs, and tools served as the criteria for making these determinations. After an initial attempt to map the HSI methods to the Vee-model, it was recognized that the majority of methods fit within the *Architectural Design* step, necessitating more granularity. As a result, this step was decomposed into two sub-steps: *Functional Architecting* and *Physical Architecting*. The reason for decomposing the step in this way was two-fold: 1) there was a clear dividing line between methods (i.e., some strongly focused on functional analysis, decomposition, etc. while others were focused on the physical design of the system); and 2) this breakdown would be better tailored to the MBSE methodology, facilitating a more effective evaluation. Additionally, the sub-step, *Technical Basis Reports*, was added to distinguish between HSI methods that informed the *Physical Architecting* activities through technology assessments and those that should be directly integrated into that step.

Due to the large volume of methods mapped to the Vee-model and to allow for sufficient effort to be applied to evaluating the HSI methods against the SE process and MBSE, a determination was made to limit the scope of subsequent analysis to only those methods that mapped to the first four steps: *Stakeholder Requirements Definition, Requirements Analysis, Functional Architecting*, and *Physical Architecting*.

5.2. HSI Methods Mapped to the Vee-model

For each of the six sources selected, a diagram was generated showing where the HSI methods mapped to the Vee-model. These diagrams are located in Appendix A.

During the methods mapping analysis, it was recognized that a few HSI methods combined the activities of multiple methods into one. Similar to the decomposition of the Vee-model step, there were some HSI methods that were decomposed to provide a more detailed level of granularity and allow for equitable comparisons between methods to determine similarities. One such method, *Task-Centered System Design (TCSD)* (Stanton, et al., 2012, Ch. 11), was identified as being too high level in order to adequately compare it to other methods. As a result, this method was decomposed into three ‘sub-methods’. Other methods that were decomposed included *Allocation of Function Analysis* (Stanton, et al., 2013, Ch. 11) and *STAGE 3: Basic Design* (Sanders and McCormick, 1993).

In the *Discussion* section that follows, each Vee-model step includes a detailed comparison between HSI methods as illustrated in the comparison tables embedded within each section.

6. DISCUSSION

6.1. Stakeholder Requirements Definition

The purpose of the *Stakeholder Requirements Definition Process* is to define the requirements for a system that can provide the services needed by users and other stakeholders in a defined environment (ISO/IEC 15288:2008). As the first step in the Vee-model, this process (and its outputs) serves as the foundation for defining and clarifying the system throughout the lifecycle. It is in this step that stakeholder needs (documented or undocumented) are converted into high-level requirements, concept documents (e.g., CONOPS), measures of effectiveness (MOEs), etc. as shown in Figure 2.

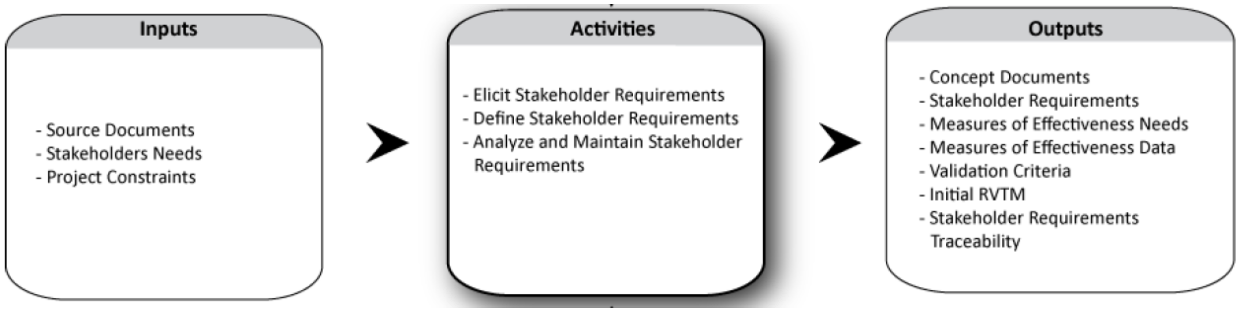

Inputs
- Source Documents
- Stakeholders Needs
- Project Constraints
Activities
- Elicit Stakeholder Requirements
- Define Stakeholder Requirements
- Analyze and Maintain Stakeholder Requirements
Outputs
- Concept Documents
- Stakeholder Requirements
- Measures of Effectiveness Needs
- Measures of Effectiveness Data
- Validation Criteria
- Initial RVTM
- Stakeholder Requirements Traceability

Figure 2: Context Diagram for Stakeholder Requirements Definition Process (INCOSE SE Handbook, v 3.2.2)

The outputs from this process will serve as the basis for performance requirements, functional requirements, non-functional requirements, and architectural constraints determined to meet the mission.

After mapping the HSI methods to the Vee-model process, nine methods were identified as applicable to this first step in the Vee-model (see Table 1 for a comparison matrix).

Table 1: Vee-model Step One: HSI Methods Comparison

HSI Methods	Mission Analysis	Mission Analysis	Focus Groups	Scenario-Based Design	User ID & Sample Task Definition	Operational Analysis	Critical Incident Study	STAGE 1: Determine Obj & Perf Specs	Mission Analysis
Mission Analysis		⊙	■	⊙	■	⊙	■	■	⊙
Mission Analysis			■	⊙	■	⊙	■	■	⊙
Focus Groups				■	■	■	■	■	■
Scenario-Based Design					■	⊙	■	■	⊙
User ID & Sample Task Definition						■	■	⊙	■
Operational Analysis							■	■	⊙
Critical Incident Study								■	■
STAGE 1: Determine Obj & Perf Specs									■
Mission Analysis									

Same	⊙
Different	■
McKneely, et al., 2006	🟦
Stanton, et al., 2012	🟨
Chapanis, 1996	⬜
Sanders and McCormick, 1993	🟥
DoD, 1999	🟧

After comparative analysis, four unique methods were identified and given the following categorical names: 1) Interviewing; 1A) Critical Incident Study (a type of interviewing); 2) Reviewing; 3) Mission Analysis. These methods were shown to be complimentary to each other, rather than competing. As shown in Figure 3, Interviewing, Critical Incident Study, and Reviewing are inputs to Mission Analysis.

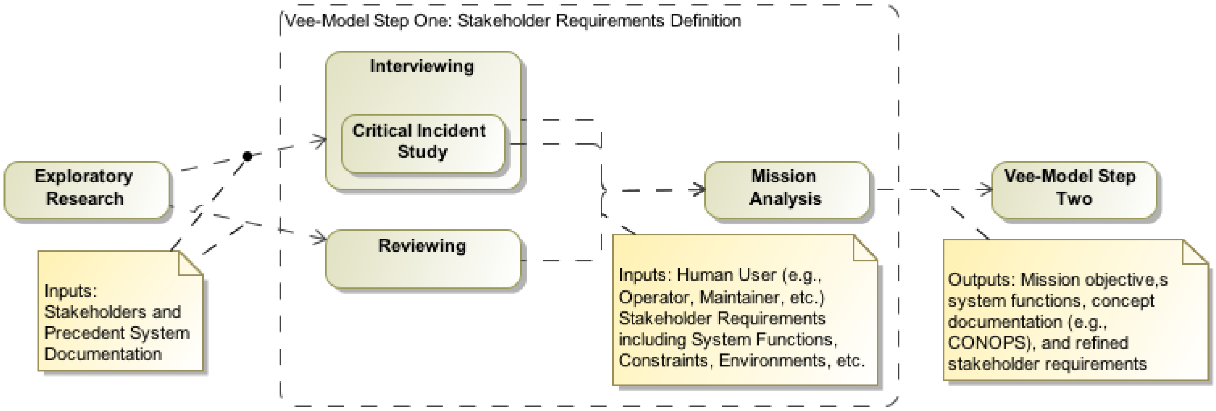

Vee-Model Step One: Stakeholder Requirements Definition
Interviewing
Critical Incident Study
Exploratory Research
Reviewing
Mission Analysis
Vee-Model Step Two
Inputs: Stakeholders and Precedent System Documentation
Inputs: Human User (e.g., Operator, Maintainer, etc.) Stakeholder Requirements including System Functions, Constraints, Environments, etc.
Outputs: Mission objective,s system functions, concept documentation (e.g., CONOPS), and refined stakeholder requirements

Figure 3: Unique and Complimentary HSI Methods

6.1.1 Unique Methods

6.1.1.1 Interviewing

Method Objective: Gather raw data regarding stakeholder requirements as input for the concept development phase of the system design.

Approach: Interview stakeholders; this includes both one-on-one and group interviews of the HSI methods reviewed. Stanton, et al. (2012, Ch. 11) provides a structured process for formally conducting group interviews that he termed, “Focus Groups”.

Inputs: Potential stakeholders, subject matter experts (SMEs), other relevant project personnel (e.g., project manager, designers, etc.), and facilitator.

Outputs: Transcripts of interviews, which include agreed upon stakeholder requirements including users and human-user interfaces.

6.1.1.1.1 Critical Incident Study

Method Objective: Identify and assess risk factors and possible risk management actions.

Approach: Conduct one-on-one or group interviews with operators of existing systems with the intent of eliminating or mitigating sources of operational failures in the system-of-interest. Chapanis (1996, Ch. 4) provides a method for collecting and analyzing data related to critical incident situations and discusses a related example.

Inputs: Legacy or existing systems and their users, operators, and/or maintainers.

Outputs: Sources of serious human-system difficulties (risk factors), suggested solutions to critical incident situations (risk management actions).

6.1.1.2 Reviewing

Method Objective: Define system objectives, identify and describe a representative list of potential users or user groups, and define system functions.

Approach: Observe/review an existing system similar to the system-of-interest. Gather relevant information for mission analysis, which may include users (Stanton, et al. 2012, Ch. 11), system performance specifications (Sanders and McCormick 1993, Ch. 22), or task descriptions (Stanton, et al. 2012, Ch.11).

Inputs: Existing systems and their users, operators, and/or maintainers.

Outputs: Objectives, representative list of user groups (e.g., operator, maintainer), detailed task descriptions including human-user interfaces.

6.1.1.3 Mission Analysis

Method Objective: Determine the mission objectives and define the basic functions that the total system (hardware, software, humanware) must perform to accomplish these objectives.

Approach: Develop mission scenarios, that may take several forms, including narratives (DoD 1999, §8.3.1.2), graphics or pictorial models (Chapanis 1996, Ch. 4) (DoD 1999, §8.3.1.1), 'storyboarding' (Stanton, et al. 2012, Ch. 11) or Hierarchical Task Analysis (HTA) (Stanton, et al. 2012, Ch. 11). These scenarios are then used to clarify the mission objectives and identify basic mission functions, inputs, outputs, environments and constraints (McKneely, et al. 2006, §3.1).

Inputs: Human user (e.g., operator, maintainer), human interfaces, stakeholder requirements, including system functions, environment, and other constraints.

Outputs: Mission objectives, system functions, concept documentation (e.g., CONOPS), and refined stakeholder requirements.

6.1.2 Unique Methods Summary

While the sources reviewed undoubtedly intended to provide methods to effectively integrate HSI factors within systems, they do not offer anything unique apart from the standard SE process.[1] Specifically, the inputs, outputs, and activities of these methods are captured within *Stakeholder Requirements Definition* (Vee-Model Step One), as is shown previously in Figures X and X. ISO/IEC 15288:2002, in particular, recognizes the need to "identify the interaction between users and the system" and that "scenarios are used to analyze the operation of the system…"

[1] As defined by ISO/IEC 15288:2002 and INCOSE SE Handbook v. 3.2.2 standards.

While these HSI methods may discuss reviewing or observing operators of similar existing systems, they neglect to address conducting reviews of source documents, a crucial input to this process step (INCOSE Handbook 2011, §4.1.1.3). Documentation from precedent systems such as a System Requirements Document (SRD), concept documentation, operating procedures, etc. can provide an excellent starting point for extracting and clarifying relevant information for the system-of-interest and should be obtained whenever possible.

Although these methods may not be novel to the SE process and despite their lack of inclusion of source documentation, it is evident that their overarching goal is to explicitly consider the human element when capturing and analyzing source requirements. Human factors knowledge and experience are necessary in this first step to ensure that human interfaces and interactions are identified early on and are well integrated and documented within the major outputs. It is unlikely that human considerations identified later on in the SE process will be appropriately[1] integrated into the system given the continually increasing costs of redesign as a project progresses through its lifecycle.

6.1.3 Models-Based Systems Engineering Support

The purpose of this section (and similar subsequent sections) is to briefly summarize the first step in the Model-Based Systems Engineering (MBSE) method and then evaluate the identified HSI methods against this method to elucidate similarities and differences that will help determine how human factors can be modeled in a system. While there exists at least half a dozen[2] leading MBSE methodologies, the method summarized here will not refer to any one in particular, but will provide a general description that may contain similar elements from each. Similarly, although there are multiple SE process standards that vary in certain aspects, most have the same foundational approach.

6.1.3.1 MBSE Stakeholder Requirements Definition Summary

The activities of the first step in the MBSE method are basically the same to SE Process Step One: describe system (product) scope; identify system stakeholders; elicit stakeholder requirements; validate stakeholder requirements. While these activities define the “what”, MBSE specifies the “how”.

If relevant strategic guidance is available (e.g., strategic vision, goals, policies, standard, etc.), a preliminary system scope may be developed prior to identifying all system stakeholders. In order to describe the system scope, a contextual analysis of the strategic problem in terms of operational capabilities is performed (Beck, 2011). While textual descriptions may be used to develop a strategic context, it is recommended that graphical models such as use cases or requirements diagrams be used, as shown in Figure 4 and Figure 5 (Beck, 2011).

[1] According to Madni (2009), HSI advocates a full lifecycle view of the integrated human-machine system during system definition, development and deployment.

[2] (Estefan, 2008).

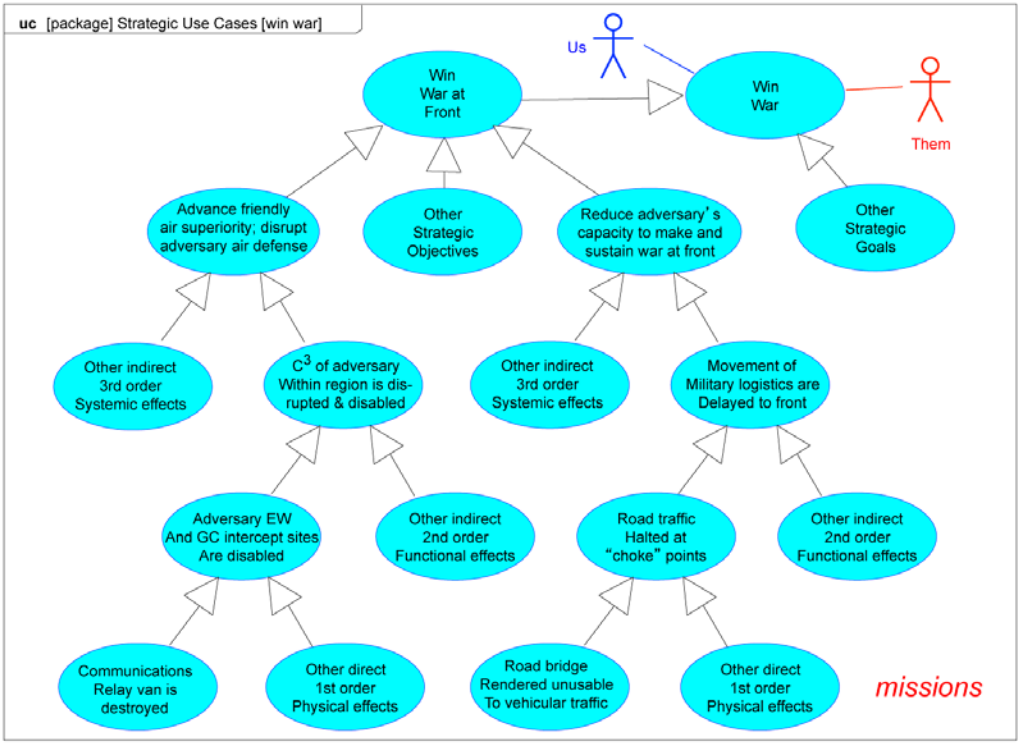
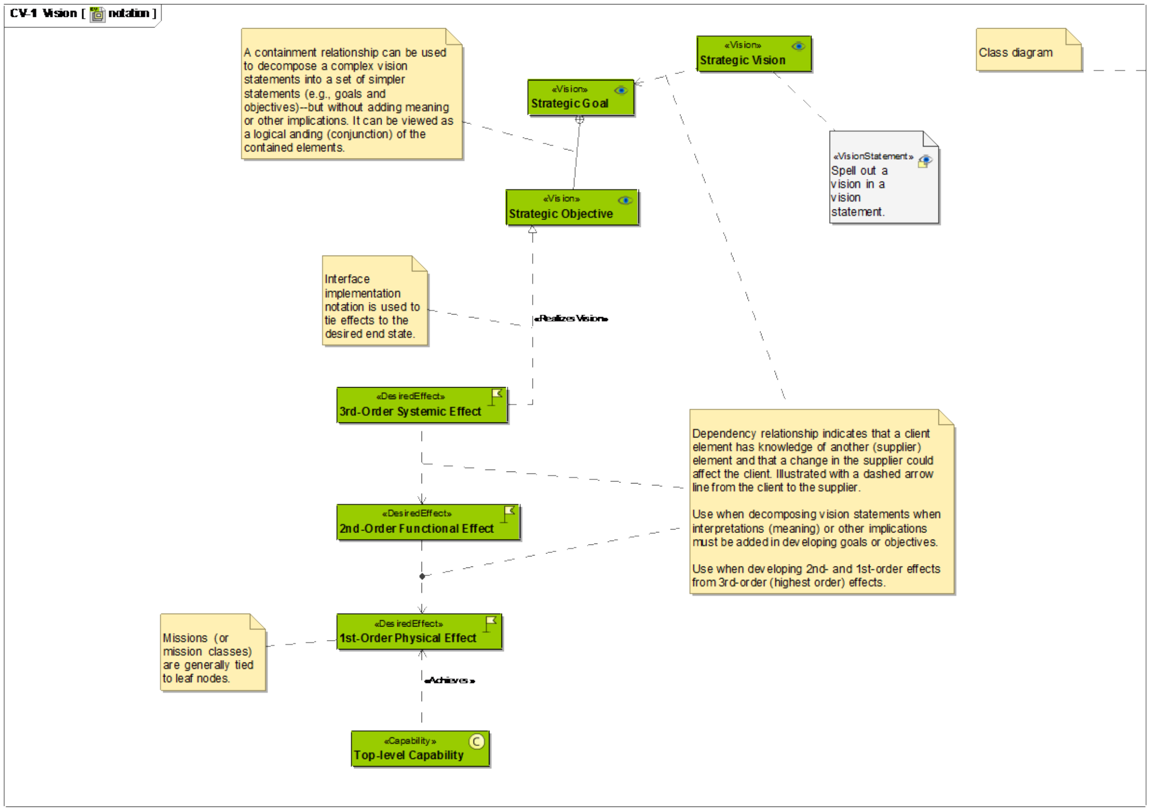

uc [package] Strategic Use Cases [win war]
Us
Them
Win War at Front
Win War
Advance friendly air superiority; disrupt adversary air defense
Other Strategic Objectives
Reduce adversary's capacity to make and sustain war at front
Other Strategic Goals
Other indirect 3rd order Systemic effects
C3 of adversary Within region is dis-rupted & disabled
Other indirect 3rd order Systemic effects
Movement of Military logistics are Delayed to front
Adversary EW And GC intercept sites Are disabled
Other indirect 2nd order Functional effects
Road traffic Halted at "choke" points
Other indirect 2nd order Functional effects
Communications Relay van is destroyed
Other direct 1st order Physical effects
Road bridge Rendered unusable To vehicular traffic
Other direct 1st order Physical effects
missions

Figure 4: Strategic context graphical example (UML or SysML *use case* diagram) (Beck, 2011).



A containment relationship can be used to decompose a complex vision statements into a set of simpler statements (e.g., goals and objectives)--but without adding meaning or other implications. It can be viewed as a logical anding (conjunction) of the contained elements.

Spell out a vision in a vision statement.

Interface implementation notation is used to tie effects to the desired end state.

Dependency relationship indicates that a client element has knowledge of another (supplier) element and that a change in the supplier could affect the client. Illustrated with a dashed arrow line from the client to the supplier.

Use when decomposing vision statements when interpretations (meaning) or other implications must be added in developing goals or objectives.

Use when developing 2nd- and 1st-order effects from 3rd-order (highest order) effects.

Missions (or mission classes) are generally tied to leaf nodes.

Figure 5: Generic strategic context graphical example (DoDAF *CV-1* diagram; in SysML use *req* diagrams) (Beck, 2011).

From this analysis, a concise problem statement and the capabilities necessary to solve this problem can be developed. Based on the strategic context and required system capabilities, a concept of the main operational context is developed in the form of a graphical depiction. This initial model should not be confused with the suite of related operational models that comprise the CONOPS defined later in this step. Rather, such a model establishes the context for developing these mission scenarios. Examples of an operational context model include DoDAF *OV-1* model, internal block diagrams (SysML) and block definition diagrams (SysML).

Once the system scope has been described or if strategic guidance does not exist, stakeholders must then be identified. Generally, stakeholders may include an enterprise, organization, team, or individual, or classes thereof who will have an interest or stake in the outcome of the project (Beck, 2011). For each stakeholder, one or more concerns should be developed that describe the interests a stakeholder has that pertains to the system's development, its operation, or any other important aspect.

Identified stakeholders should then be polled (e.g., interview, request for information) to collect system requirements. After obtaining requirements from all stakeholders, a necessary risk-reducing activity is to rewrite the requirements to ensure the necessary attributes[1] are considered. In defining stakeholder requirements, it is helpful to develop *views* and *viewpoints* for the stakeholders. A *view* is a representation of a whole system from the perspective of a related set of concerns (IEEE Std 1471 §3.9). A *viewpoint* is a pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis (IEEE Std 1741 §3.10). The systems engineer can package multiple views by types of models (e.g., behavior, structure) or by model elements (e.g., classes, typed blocks) in order to demonstrate that the system will satisfy the concerns of the respective stakeholder (Beck, 2011). Similar to the idea of *views* and *viewpoints* are concept documents (e.g., production, deployment, operations, support, disposal), which serve as major outputs in this step.[2] An example diagram of simple views and viewpoints is shown in Figure 6.

[1] INCOSE SE Handbook v. 3.2.2, §4.2.2.2 *Characteristics of Good Requirements*

[2] INCOSE SE Handbook v. 3.2.2 §4.1.1.4 *Outputs*

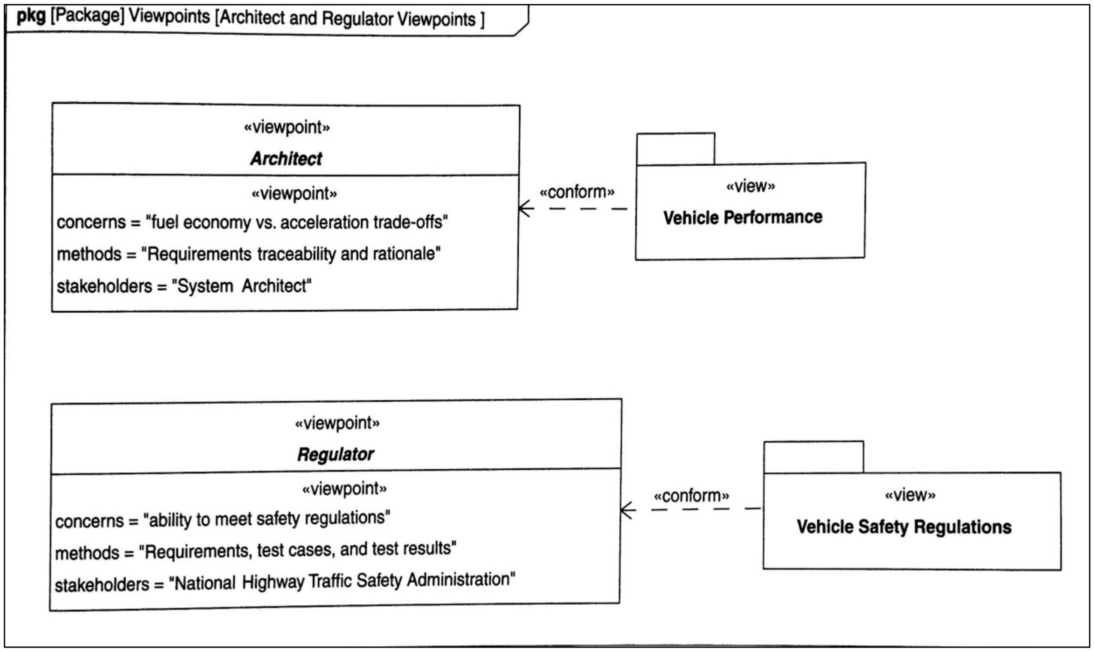

pkg [Package] Viewpoints [Architect and Regulator Viewpoints]
«viewpoint»
Architect
«viewpoint»
concerns = "fuel economy vs. acceleration trade-offs"
methods = "Requirements traceability and rationale"
stakeholders = "System Architect"
«conform»
«view»
Vehicle Performance
«viewpoint»
Regulator
«viewpoint»
concerns = "ability to meet safety regulations"
methods = "Requirements, test cases, and test results"
stakeholders = "National Highway Traffic Safety Administration"
«conform»
«view»
Vehicle Safety Regulations

Figure 6: A SysML Package Diagram (pkg) is used to develop simple views and viewpoints (Friedenthal, et al., Chapter 4, 2011).

Although there may be multiple views for one system, probably the most key is the CONOPS or operational model. In order to develop content for the operational model, the strategic mission is decomposed into lower-level operational activities and resource flow (e.g., information) exchanges are identified (Beck, 2011). To start, use cases are developed to capture the mission objectives as they relate to the system stakeholders. For each use case, a set of well-posed mission scenarios that explore the range of possible operational conditions is developed. From this set, critical scenario objectives are down-selected and analyzed to identify the necessary system capabilities that will produce the effects that ultimately meet stakeholder objectives.

After determining the capabilities that will fulfill the mission objective, more in-depth analyses are used to develop an operational plan. These include identifying the "as is" operations in legacy systems, determining the mission tasks to analyze (based on the previously identified capabilities) in the form of a function structure, and identifying evaluation criteria (e.g., measures of effectiveness). As shown by Beck (2011), these analyses can all be captured using models.

Finally, validation test cases are defined based on the mission scenarios, mission tasks, and the evaluation criteria. The objectives of validation in this step are to ensure traceability between the documented stakeholder needs and the content contained in the *views*, resolve any conflicts or inconsistency within the requirements, ensure all validation activities are well-documented, and the validated set of requirements are under configuration management.

For an example set of diagrams that illustrate the activities in this step, please refer to Appendix B.

6.1.3.2 Evaluation of HSI Methods Against MBSE

6.1.3.2.1 Interviewing

As mentioned in the preceding section, having a “good” set of requirements will help reduce risk for the project. An important part of developing such requirements is to ensure that the customer needs are effectively translated, interpreted, and understood. This is especially important when collecting requirements related to human factors because they may be more difficult to articulate. Utilizing the one-on-one interview or Focus Groups (Stanton, et al., 2012, Ch. 11) approaches would provide the opportunity for the systems engineer to conduct in-depth queries to help stakeholders define their needs and clear up any ambiguous information. An effective method for requirements elicitation, *Interviewing* is already captured within the first step of the MBSE method. However, it would be beneficial to emphasize the importance of human factors during this activity to ensure relevant HSI concerns are vetted. The use of *views* and *viewpoints* would be especially useful for capturing human factors such as usability, ergonomics, safety, etc. For example, a *viewpoint* may include information about the operator and related concerns (e.g., safety during operation, usability of controls). A “human” *view* could then be generated from this viewpoint and would include all aspects of the system that interfaces and interacts with the operator. These views and viewpoints could then be iteratively developed with stakeholder input through individual meetings or group reviews where the models would serve as the focus of discussion.

6.1.3.2.2 Critical Incident Study

Although similar to *Interviewing*, this HSI method is only partially integrated within MBSE in that the approach is captured, but the objective is not. Specifically, there is a risk reducing activity within MBSE by rewriting requirements *after* they have been collected. The difference is that *Critical Incident Study* focuses on reducing risk *during* requirements elicitation by conducting interviews with the intent of eliminating or mitigating sources of operational failure. Of course, this assumes that a legacy or existing system exists and their operators and maintainers are available for interview. It is recommended that this method be integrated in the MBSE method in order to more thoroughly under the circumstances that lead to human-system difficulties in order to mitigate these risk factors during new system development. Some of the advantages of integrating *Critical Incident Study* into the first MBSE step include:

1. Sources of human-system difficulties can be used as strategic guidance to develop a system scope. Specifically, critical incident situations reveal gaps in capability and can be used to develop a problem statement.
2. Sources of human-system difficulties can be used to define the “as is” operations and highlight mission tasks that require in-depth analysis.
3. Suggested solutions to the critical incident situations can be used to support identification of necessary capabilities to overcome system deficiencies.
4. Suggested solutions to the critical incident situations can be used to initiate risk management actions early in the development lifecycle.

6.1.3.2.3 Reviewing

Like *Critical Incident Study*, this HSI method assumes that an existing system is available for study. When compared to MBSE, *Reviewing* does not really offer anything unique. The method objective of defining the system objective (i.e., describe system scope), identify a list of potential users (i.e., identify stakeholders), and define system functions (i.e., determine the capabilities that will fulfill the mission objective) all fall in line with the MBSE activities.

However, one key difference is that while MBSE suggests using existing strategic guidance to define a system scope and gathering requirements from stakeholders, it does not include observation or review of a legacy system. Important human factors information (e.g., human-system interactions) could be gleaned by simply observing the system in operation. This information could then be used to supplement stakeholder-provided needs and constraints. At the very least, documented observations could be used to define task descriptions to develop a function structure when determining which mission tasks require further scrutiny.

One other minor distinction is that while *Reviewing* indicates that system performance specifications (i.e., MOEs) should be gathered in order to inform mission analysis, the MBSE method explains that MOEs are derived out of mission analysis. This difference probably owes to the fact that *Reviewing* is based on observation of an existing system, which most likely already has established system performance specifications. MBSE does not necessarily preclude MOEs from informing mission analysis and, if a legacy system exists, this information should be used when developing mission scenarios, even though it is likely that the evaluation criteria for the new system will be redefined.

6.1.3.2.4 Mission Analysis

The objectives and activities of this HSI method closely resemble the *Define Operational Model* activity within MBSE. Specifically, both seek to determine the mission objectives and basic functions of the system by developing mission scenarios. Both describe multiple acceptable approaches (e.g., narratives, graphic models, storyboarding, HTA), although they may use varying terminology. MBSE especially calls the development of mission scenarios a “critical” step (Beck, 2011).

Although MBSE does not preclude textual descriptions, it is expected that models be leveraged to conduct mission analysis activities due to their clarity, organization, traceability, and their ability to capture relationships and constraints (e.g., parametric diagrams. Again, emphasis should be placed on elements that involve the human user and human-system interfaces). Appendix B offers a set of example models that illustrates the potential of MBSE to effectively execute the *Stakeholder Requirements Definition* step.

6.1.4 Models-Based Systems Engineering Support Summary

For the most part, the HSI methods evaluated are captured within MBSE, although the terminology used may vary based on the source. A few differences introduced by the HSI methods that would enhance MBSE include:

1. Observe/review of existing/legacy systems (when available)
2. Integrate *Critical Incident Study* to support:
    a. Development of system scope;
    b. Identification of “as is” operations;

c. Identification of solutions to capability gaps; and
d. Initiation of risk management actions.

Despite the fact that MBSE already incorporates most of these HSI methods in *Stakeholder Requirements Definition*, the chief point is to consider human factors both during stakeholder elicitation and mission analysis.

6.2. Requirements Analysis

The purpose of the *Requirements Analysis Process* is to transform the stakeholder, requirement-driven view of desired services into a technical view of a required product that could deliver those services (ISO/IEC 15288:2008). The objective in this second step of the Vee-model is to build a technical representation (i.e., defined requirements) of the system derived from elicited stakeholder needs, which will serve as the basis for architectural design, integration, and verification. As such, a major output of this step is a set of system functions, which must remain abstract enough such that no particular implementation is implied. In addition, performance requirements, non-functional requirements, and architectural constraints must be defined, as these will affect the emergent behaviors of the integrated system. As shown in Figure 7, other common outputs include a specification tree, a hierarchical representation of the set of specification for the system, and a system specification, a formal document of the approved system requirements.

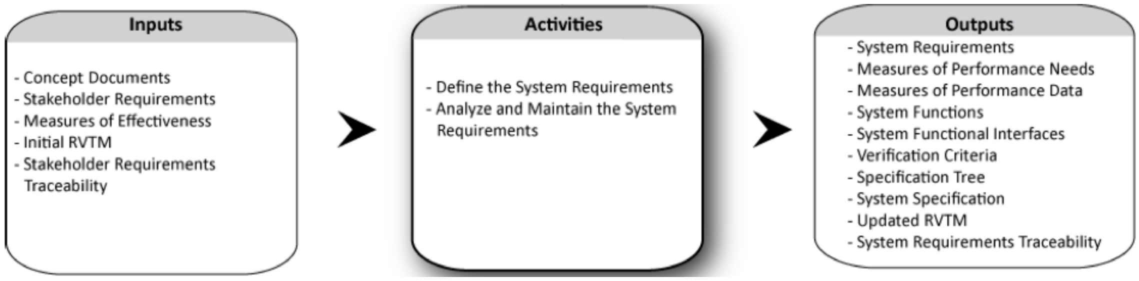

Inputs
- Concept Documents
- Stakeholder Requirements
- Measures of Effectiveness
- Initial RVTM
- Stakeholder Requirements Traceability
Activities
- Define the System Requirements
- Analyze and Maintain the System Requirements
Outputs
- System Requirements
- Measures of Performance Needs
- Measures of Performance Data
- System Functions
- System Functional Interfaces
- Verification Criteria
- Specification Tree
- System Specification
- Updated RVTM
- System Requirements Traceability

Figure 7: Context Diagram for Requirements Analysis Process (INCOSE SE Handbook, v 3.2.2)

Due to the iterative[1] and recursive[2] nature of this process, requirements may change based on new information gleaned from later process systems. Caution is recommended, however, as changes in requirements later in the development cycle can have a significant cost impact on the project, possibly resulting in cancellation (INCOSE SE Handbook v 3.2.2, §4.2.1.2).

Based on the mapping analysis, six methods were identified as applicable *Requirements Analysis* (see Table 2for a comparison matrix).

[1] When the application of the same process or set of processes is repeated on the same level of the system, the application is referred to as iterative (ISO/IEC CD 29148, *Requirements Engineering*).
[2] When the same set of processes or the same set of process activities are applied to successive levels of system elements within the system structure, the application form is referred to as recursive (ISO/IEC CD 29148, *Requirements Engineering*).

Table 2: Vee-model Step Two: HSI Methods Comparison

HSI Methods	Requirements Analysis	User-Centered Requirements Analysis	STAGE 2: Definition of the System	Functional Flow Analysis	Decision-Action Analysis	Analysis of Similar Systems
Requirements Analysis		⊙	■	■	■	■
User-Centered Requirements Analysis			■	■	■	■
STAGE 2: Definition of the System				⊙	⊙	■
Functional Flow Analysis					⊙	■
Decision-Action Analysis						■
Analysis of Similar Systems						

Same	⊙
Different	■
McKneely, et al., 2006	🟦
Stanton, et al., 2012	🟨
Chapanis, 1996	⬜
Sanders and McCormick, 1993	🟥

After comparison, three unique methods were identified and categorized as follows: 1) Analysis of Similar Systems; 2) Requirements Analysis; 2A) Functional Flow Block Diagramming (FFBD) (a sub-method for identifying system functions). Similar to the previous step, there is a complimentary element between these methods. As shown in Figure 8, Analysis of Similar Systems provides input to Requirements Analysis and FFBD. However, depending on the type of project, there are multiple approaches to creating a FFBD that will be explored in more detail below.

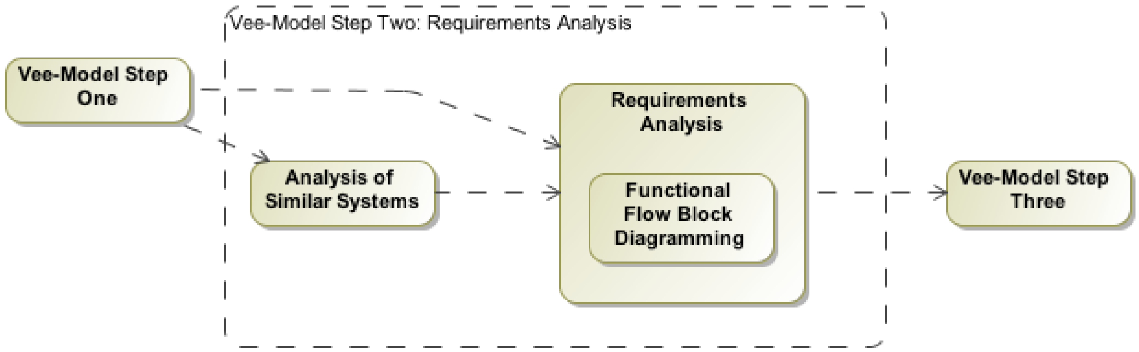

Vee-Model Step Two: Requirements Analysis
Vee-Model Step One
Analysis of Similar Systems
Requirements Analysis
Functional Flow Block Diagramming
Vee-Model Step Three

Figure 8: Vee-Model Step Two HSI Methods

6.2.1 Unique Methods

6.2.1.1 Analysis of Similar Systems

Method Objective: Identify salient features of systems that are similar to the one under consideration to provide human factors considerations input into *Requirements Analysis*.

Approach: Analyze data output from Vee-Model Step One HSI methods: *Interviewing*, *Critical Incident Study*, and *Reviewing*. Although information produced from these methods were used as inputs to *Mission Analysis*, the focus of this method is to extract information that provides the level of detail necessary for developing technical requirements and functions (e.g., skills, training, HF design problems, operability data, etc.) and is the reason for its inclusion with Vee-Model Step Two. Other potential data sources include questionnaires, activity analyses, or accident investigations (Chapanis 1996, Ch. 4).

Inputs: Productivity records, maintenance records, training records, accident or incident reports.

Outputs: Skills assessment, identification of relevant environmental factors, estimates of future staffing and manpower requirements, identification of operator and maintainer problems, preliminary assessments of workloads and stress levels, assessments of the desirability of and consequences of reallocating system functions.

6.2.1.2 Requirements Analysis

Method Objective: Specify the system characteristics necessary to meet the stakeholder requirements.

Approach: Develop system technical requirements and functions. Although both McKneely, et al. (2006) and Stanton, et al. (2012) suggest identifying intended system users and maintainers that the future proposed design with cater for, it is assumed that this activity would have been completed in the previous step if the Vee-model process is being used. Define MOEs (if not done previously) and Measures of Performance (MOPs) as they pertain to mission, human, and function requirements (McKneely, et al. 2006). Feasibility and internal compatibility of system requirements are assessed (McKneely, et al. 2006). One way to accomplish this is by the use of design scenarios or storybooks (Stanton, et al. 2012, Ch. 11). For users, tasks, and scenarios, Stanton, et al. (2012) recommend that each should be categorized as “absolutely must include”, “should include if possible”, and “exclude”. The role of the human, manning, training and cost guidelines are also developed (McKneely, et al. 2006).

Inputs: Mission objectives, system functions (i.e., high-level, non-technical), concept documentation, stakeholder requirements, and outputs from *Analysis of Similar Systems*, such as skills, staffing, manpower, environment, and human factors assessments.

Outputs: System requirements, system functions (i.e., high-level, technical), MOEs, MOPs, feasibility/compatibility assessment results (including any scenarios/storybooks), and human factors guidelines.

6.2.1.2.1 Functional Flow Block Diagramming

Method Objective: Specify the functions that must be performed by the system to meet the stakeholder requirements.

Approach: Develop functional flow block diagrams (a.k.a., functional flow diagrams, functional block diagram, or functional flows) that provide a graphical, sequential ordering of functions. These diagrams depict the interrelationships among the system functions, with each box representing one function (Sanders and McCormick 1993, Ch. 22). These functions are also numbered in a way that clarifies their relationship to one another and permits traceability of functions through the whole system (Chapanis 1996, Ch. 4). It should be noted that although FFBDs can be decomposed into multiple levels of detail, only “zero-order” and “first-order” are typically necessary to capture high-level functions. During the next step in the Vee-model, *Functional Architecting*, FFBDs may be used to decompose higher-level functions into detailed functions, as necessary. Figure 9 illustrates a zero-order FFBD, and Figure 10 illustrates a first-order FFBD, which elaborates on one of the zero-order functions.

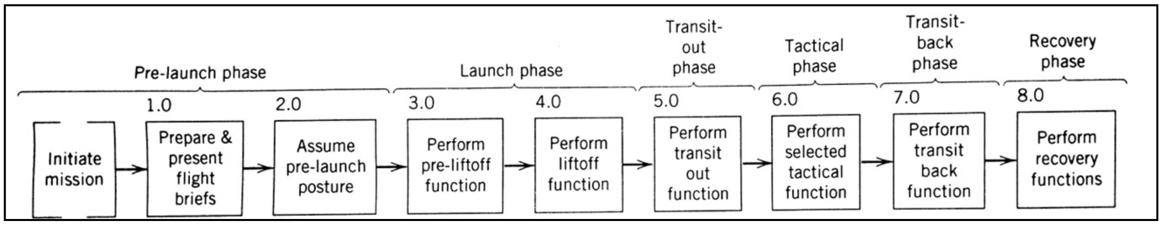

Pre-launch phase
Launch phase
Transit-out phase
Tactical phase
Transit-back phase
Recovery phase
Initiate mission
1.0
Prepare & present flight briefs
2.0
Assume pre-launch posture
3.0
Perform pre-liftoff function
4.0
Perform liftoff function
5.0
Perform transit out function
6.0
Perform selected tactical function
7.0
Perform transit back function
8.0
Perform recovery functions

Figure 9: A zero-order functional flow block diagram (Chapanis 1996, Ch. 4).

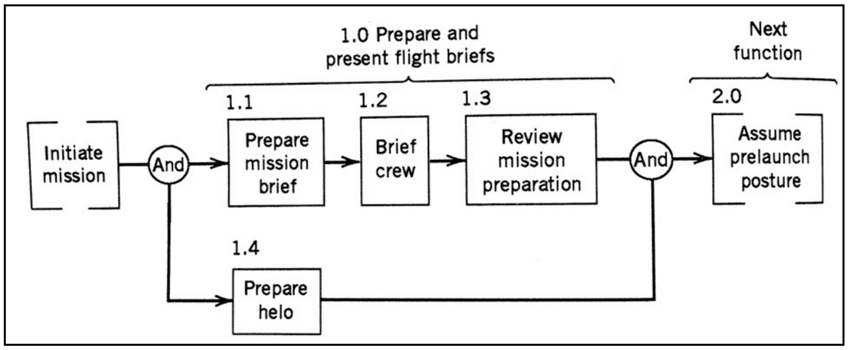


Figure 10: A first-order functional flow block diagram (Chapanis 1996, Ch. 4).

FFBDs may also be used to identify and depict functions in a system in which binary decisions have to be made (e.g., software-oriented projects). Chapanis (1996) terms this variation "Decision-Action Analysis."

Human factors specialists are tasked to ensure that the functions identified match the needs of the intended users (Sanders and McCormick 1993, Ch. 22).

Inputs: Mission objectives, system functions (i.e., high-level, non-technical), concept documentation, and stakeholder requirements.

Outputs: FFBDs, system functions (i.e., high-level, technical).

6.2.2 Unique Methods Summary

Generally, the activities described by these HSI methods do not introduce anything novel to the SE Process. However, within the context of *Requirements Analysis*, they offer valuable enhancements. For example, INCOSE identifies FFBDs as a "Cross-Cutting Technical Method" that may be used across the system lifecycle (INCOSE SE Handbook v 3.2.2, §4.12), but does not elaborate its application to this process step. The application of FFBDs to the *Requirements Analysis* process as discussed in these HSI methods, however, highlights the advantages of its use, such as enabling the justification of requirements (i.e., ensures identified functions match the needs of intended users by serving as a detailed "checklist" during verification), avoiding over/under specification (higher level functions are identified and described iteratively), and serving as a key input to subsequent process steps (i.e., allocation determinations and trade studies).

Analysis of Similar Systems, like the *Reviewing* and *Interviewing* HSI methods, feeds important human factors data within the SE Process. Since there is a more detailed (i.e., technical) focus on human factors within this analysis (e.g., skills, training, manpower), its outputs are more appropriately integrated within *Requirements Analysis*. Although the SE Process touches on this[1], it does not go into same level of depth as this HSI method.

[1] ISO/IEC 15288:2002, §5.5.3.3 e) *Specify system requirements and functions…that relate to critical qualities, such as health, safety, security, reliability, availability and supportability.*

Although these HSI methods discuss at a high-level the need for traceable and verifiable requirements, they do not consider the role that configuration control plays in maintaining continuity, a critical activity to ensure that system requirements meet at least one stakeholder requirement (ISO/IEC 15288:2002, §5.5.3.3). Rationale, decisions and assumptions are also maintained along with system requirements in a data repository, which will feed *Architectural Design*. The human factors data collected from *Analysis of Similar Systems*, for example, should be maintained in such a repository to help capture the details that support the HSI-related requirements.

While the HSI methods mapped to the *Requirements Analysis* process may not make major contributions to this step, they help instill a user-centered approach in defining requirements. In a way, these methods reuse traditional *Requirements Analysis* concepts and/or tools, but through the lens of HSI. With the human factors elements integrated into system requirements, the functional and physical architectures should inherently reflect the needs of the user.

6.2.3 Models-Based Systems Engineering Support

6.2.3.1 MBSE System Technical Requirements Summary

The purpose of MBSE Step Two is to translate customer needs (identified in MBSE Step One) into a set of "black box" system requirements (Beck, 2011). The activities closely resemble those defined in the SE Process above, yet at a more detailed level.

In order to begin transforming the stakeholder requirements in system (technical) requirements, the first step is to develop an initial Elaborated Context Diagram (ECD). An ECD captures the system black box requirements (e.g., functions, interfaces, controls, performance, non-behavioral) and represents a static, composite view of the system input/output (I/O) flows across multiple scenarios (Beck, 2011). In SysML, this is typically represented using an internal block diagram (ibd) as shown in Figure 11.

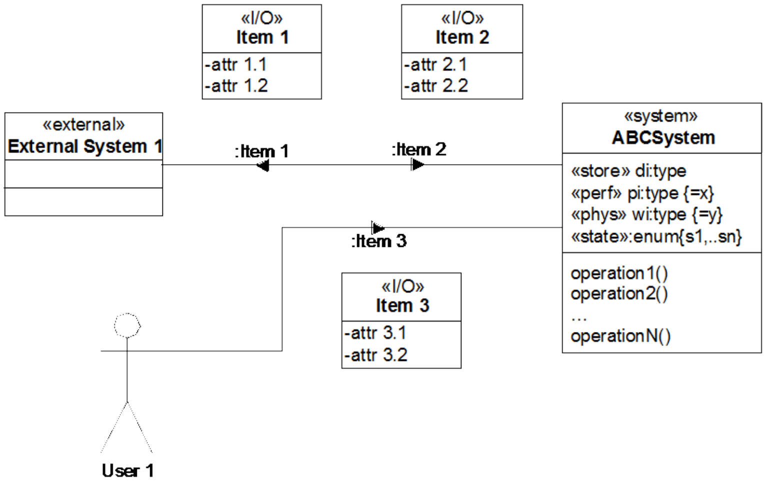


Figure 11: Generic ECD (INCOSE, 2006).

Captured within the ECD are the system-of-interest, external systems, users, and I/O items (flows).

After constructing an ECD, the next step is to develop system scenarios. Use cases and scenarios defined in the previous MBSE step are further developed using behavioral diagrams to produce system scenarios (Beck, 2011). The intent of creating system scenarios is to elaborate the requirements for an element or set of elements contained in the ECD. In SysML, these elements are typically known as *operations* or *attributes*. Figure 12illustrates a scenario using a subset of these elements from the ECD above using sequence diagram (sd).

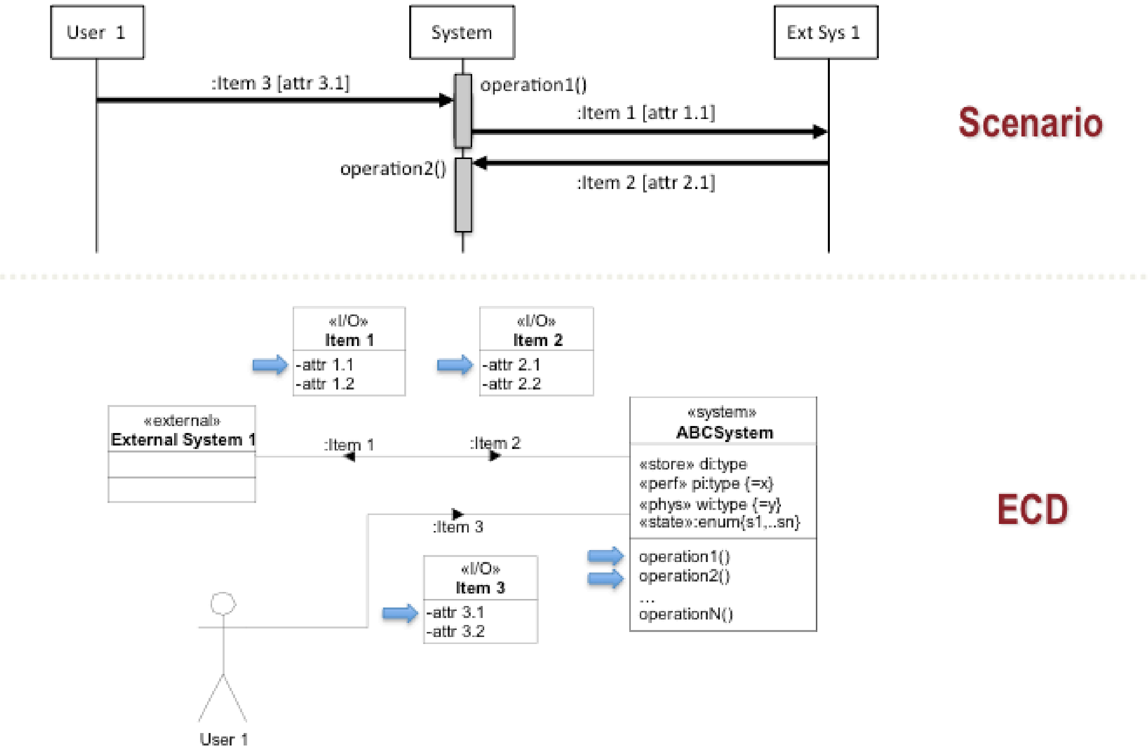


Figure 12: System Scenarios vs. ECD (INCOSE, 2006).

System scenarios should address all high-probability mission scenarios identified in the previous step, external system interactions with the system-of-interest, stressing scenarios, failure conditions, and support scenarios (e.g., system backups, configuration management, etc.).

Using the ECD and supporting modeling artifacts, block box requirements are further refined. For example, highly detailed functions may need the use of algorithms and mathematical relationships in order to be appropriately specified. Likewise, critical MOPs that impact mission MOEs may need to be identified (e.g., timelines in critical system scenarios that affect mission critical timelines) (Beck, 2011).

Design constraints also serve as an important input in developing technical requirements[1], and as such, must be assembled from the outputs of the previous step (e.g., concept documents) and

[1] INCOSE SE Handbook v 3.2.2, §4.2.1.5

linked to the ECD. These constraints are realized as design constraints within the SysML language, as illustrated in Figure 13.

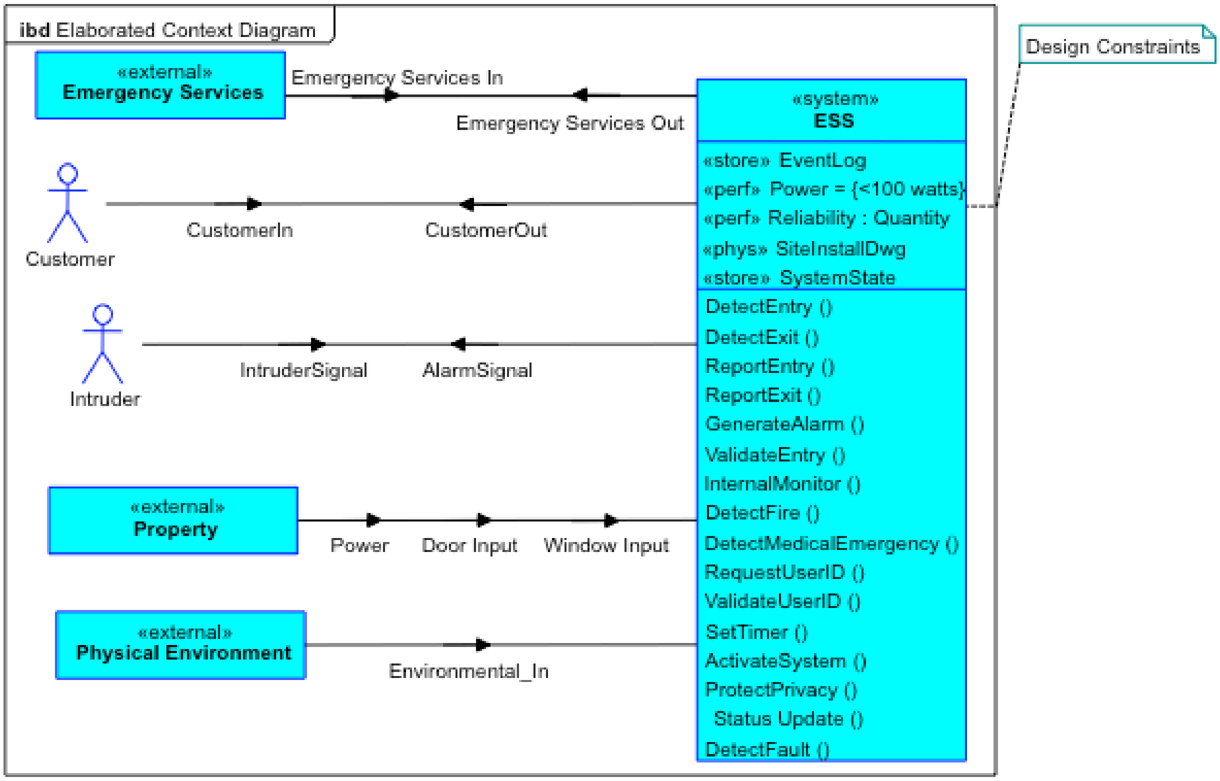


Figure 13: Elaborated Context Diagram (ECD) with design constraints (INCOSE, 2006).

The next step is to perform system requirements analyses, such as requirements variation, trade off, effectiveness, and risk analyses. Based on the results, system requirements and constraints are updated. Finally, the system requirements are verified against established requirements criteria[1] and validated against stakeholder requirements to ensure they will fulfill the mission objectives. The main output of this process step should be a set of requirements that describe the required system functions and associated I/O, the required external interfaces, and MOPs.

6.2.3.2 Evaluation of HSI Methods Against MBSE

6.2.3.2.1 Analysis of Similar Systems

Typically, the inputs to the *Requirements Analysis* step are collected from the outputs of the previous step, *Stakeholder Requirements Definition*. As described in the previous evaluation of the MBSE Step One, *Interviewing* and *Reviewing* can provide valuable human factors information for defining mission objectives and stakeholder requirements. However, this information will tend to be more high level and does not capture the necessary details (e.g., skills, training, HF design problems, etc.) to sufficiently capture HSI considerations within the

[1] INCOSE SE Handbook v. 3.2.2, §4.2.2.2 *Characteristics of Good Requirements*

system technical requirements. *Analysis of Similar Systems* could enhance the MBSE *System Technical Requirements* step by refining this information and providing it as a secondary input.

Specifically, outputs from this method could supplement the development of an ECD and related scenarios. For example, operability data from a legacy system could be used as a starting point for either identifying operations of the system-of-interest, or (if operations are already identified) creating related scenarios. Environmental factors could help identify/elaborate I/O items and skills assessment information could help detail the human-system interactions (i.e., depending on the required user skills, the system-of-interest may need to automate certain operations). Given the potential advantages of incorporating the outputs of this method into MBSE, it is recommended that it be integrated into this step.

6.2.3.2.2 Requirements Analysis

As stated previously, this method does not offer major contributions to the established SE Process. Nonetheless, given this method's recommendation to use design scenarios or storybooks (Stanton, et al. 2012, Ch. 11), it would potentially integrate better with MBSE. The intent of using scenarios in this HSI method is to assess the feasibility and internal compatibility of system requirements, which can be viewed as part of the MBSE goal to elaborate requirements for elements within the ECD. Both methods recognize the need for prioritization of scenarios, although the HSI method provides a more defined categorization guideline. Overall, MBSE would benefit by integrating this method in order to better capture the role of the human within the technical requirements.

6.2.3.2.3 Functional Flow Block Diagramming

Since the objective of this method is to specify system functions, the potential enhancements to MBSE are potentially two-fold. With regards to developing an ECD, the act of creating FFBDs would help the systems engineer determine the necessary functions (operations) and related data (attributes) that must be defined for the system-of-interest. Since FFBDs can be decomposed into increasing levels of detail (e.g., zero-order, first-order, etc.), they would serve as excellent tools for deriving technical requirements from stakeholder requirements. Additionally, if an ECD has already been created and has defined operations, FFBDs could be used to elaborate these operations in the form of scenarios. By diagramming scenarios with FFBDs, functions that involve human-system interaction may become more apparent, facilitating efficient verification to ensure that the functions match the needs of the intended user.

6.2.4 Models-Based Systems Engineering Support Summary

Although similar to the *System Technical Requirements* step, these HSI methods offer some enhancements that are not addressed in this step that would be advantageous to MBSE:

1. Apply relevant HSI information from *Analysis of Similar Systems* to the development of the ECD and related scenarios
2. Focus on HSI considerations when building scenarios
3. Prioritize scenarios using the following categorization: "absolutely must include", "should include if possible", and "exclude"

4. Utilize FFBDs to:
    a. Derive technical requirements from stakeholder requirements;
    b. Determine operations and attributes within the ECD;
    c. Elaborate operations in the form of scenarios; and
    d. Assist in verification by elucidating the functions that involve human-system interactions.

Although there is not a strong correlation between the HSI methods to MBSE, they have enough similarities to effectively integrate into this step while offering a renewed emphasis on human factors.

6.3. Architectural Design: Functional Architecting

As explained in *Mapping Analysis* (§5) above, *Architectural Design* has been decomposed into two sub-steps: *Functional Architecting* and *Physical Architecting* in order to facilitate a more accurate mapping of HSI methods to the SE Process. However, in keeping with the structure of this paper, the following is a brief description of this SE process step as defined by ISO/IEC 15288 and the INCOSE SE Handbook. This description should be viewed as an overall summary for both sub-steps, with more detailed descriptions in the subsequent MBSE sections.

The purpose of *Architectural Design* is to synthesize a solution that satisfies system requirements (ISO/IEC 15288:2008). Multiple possible implementations are usually developed to explore the trade space in order to reach an optimal architecture. The project baseline, as documented in the previous two steps, serves as the primary input along with lifecycle constraints (e.g., operability, maintainability, disposability). System-level functions are decomposed down the lowest logical element, requirements are partitioned and allocated to these elements, and interface requirements are documented. A major output of this step is the system architecture description, which is typically represented using diagrams, including justifications for the selected concept. As shown in Figure 14, Technical Performance Measures (TPMs) and system element requirements traceability are also among the outputs of this step.

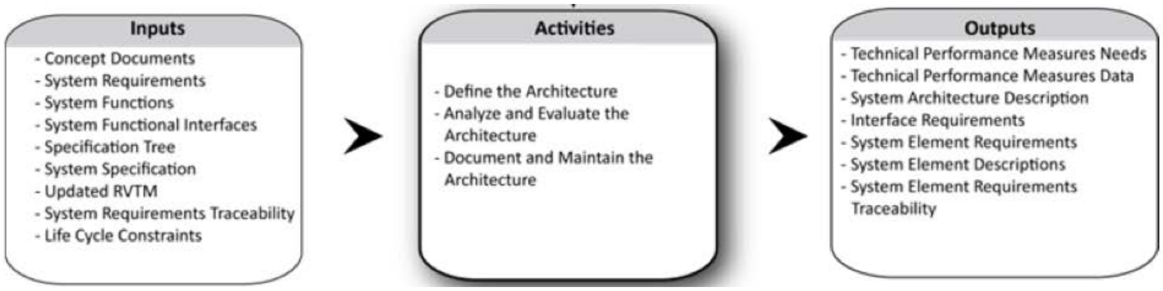

Inputs
- Concept Documents
- System Requirements
- System Functions
- System Functional Interfaces
- Specification Tree
- System Specification
- Updated RVTM
- System Requirements Traceability
- Life Cycle Constraints
Activities
- Define the Architecture
- Analyze and Evaluate the Architecture
- Document and Maintain the Architecture
Outputs
- Technical Performance Measures Needs
- Technical Performance Measures Data
- System Architecture Description
- Interface Requirements
- System Element Requirements
- System Element Descriptions
- System Element Requirements Traceability

Figure 14: Context Diagram for Architectural Design Process (INCOSE SE Handbook, v 3.2.2)

In addition, the system element requirements derived in this step will serve as the basis for verifying the realized system and generating a verification strategy (ISO/IEC 15288:2008). It should also be noted that INCOSE (2011, v 3.2.2, §4.3.1.5) mentions capturing human elements when identifying interfaces and interactions as well as including HSI considerations in support of defining a system integration strategy.

Based on the mapping analysis, twenty-one methods were identified as applicable to *Functional Architecting* (see Table 3for a comparison matrix).

Table 3: Vee-model Step Three: HSI Methods Comparison

HSI Methods	Function Analysis	Design Concept Evaluation	Allocation of Function Analysis Steps 1 & 2	STAGE 2: Definition of the System	HierarchicalTask Analysis	Task Analysis for Knowledge Description	Task Action Grammar (TAG)	Job Process Charts	Functional Flow Analysis	Decision-Action Analysis	Action-Information Analysis	Timeline Analysis	Operational Sequence Analysis	Simulation	Operational Sequence Diagrams	Flow Process Charts	Decision/Action Diagrams	Timeline	Functional Flow Diagram	Integrated Definition for Functional Modeling (IDEF0)	Action/Information Requirements
Function Analysis		■	■	⊙	■	■	■	⊙	⊙	⊙	■	■	■	■	■	⊙	⊙	■	⊙	■	■
Design Concept Evaluation			■	■	■	■	■	■	■	■	⊙	■	■	■	■	■	■	■	■	■	⊙
Allocation of Function Analysis Steps 1 & 2				■	⊙	■	■	■	■	■	■	■	■	■	■	■	■	■	■	⊙	■
STAGE 2: Definition of the System					■	■	■	⊙	⊙	⊙	■	■	■	■	■	⊙	⊙	■	⊙	■	■
Hierarchical Task Analysis						■	■	■	■	■	■	■	■	■	■	■	■	■	■	⊙	■
Task Analysis for Knowledge Description							⊙	■	■	■	■	■	■	■	■	■	■	■	■	■	■
Task Action Grammar (TAG)								■	■	■	■	■	■	■	■	■	■	■	■	■	■
Job Process Charts									⊙	⊙	■	■	■	■	■	⊙	⊙	■	⊙	■	■
Functional Flow Analysis										⊙	■	■	■	■	■	⊙	⊙	■	⊙	■	■
Decision-Action Analysis											■	■	■	■	■	⊙	⊙	■	⊙	■	■
Action-Information Analysis												■	■	■	■	■	■	■	■	■	⊙
Timeline Analysis													■	■	■	■	■	⊙	■	■	■
Operational Sequence Analysis														■	⊙	■	■	■	■	■	■
Simulation															■	■	■	■	■	■	■
Operational Sequence Diagrams																■	■	■	■	■	■
Flow Process Charts																	⊙	■	⊙	■	■
Decision/Action Diagrams																		■	⊙	■	■
Timeline																			■	■	■
Functional Flow Diagram																				■	■
Integrated Definition for Functional Modeling (IDEF0)																					■
Action/Information Requirements																					

Same	⊙
Different	■
McKneely, et al., 2006	🟦
Stanton, et al., 2012	🟨
Stammers, et. al, 1990	🟪
Chapanis, 1996	□
Sanders and McCormick, 1993	🟥
DoD, 1999	🟧

The results of the comparison helped to identify six unique methods: 1) Task Analysis for Knowledge Description (TAKD) 2) Functional Analysis (includes Functional Flow Block Diagramming, Flow Process Charts, and Hierarchical Task Analysis (HTA); 3) Timeline Analysis; 4) Simulation; 5) Action/Information Analysis; and 6) Operational Sequence Diagrams (OSDs).

While there are complimentary interactions between these methods, most of them can be used as separate methods and some may even be redundant (e.g., when decision/action diagrams are used, OSDs are not (DoD 1999, §8.3.6.4)). As shown in Figure 15, TAKD is a preliminary step to Functional Analysis, which has multiple "sub-methods". Functional Analysis then outputs to Action/Information Analysis and/or OSDs. Timeline Analysis and Simulation are separate methods, however they may inform other HSI methods as shown in the diagram.

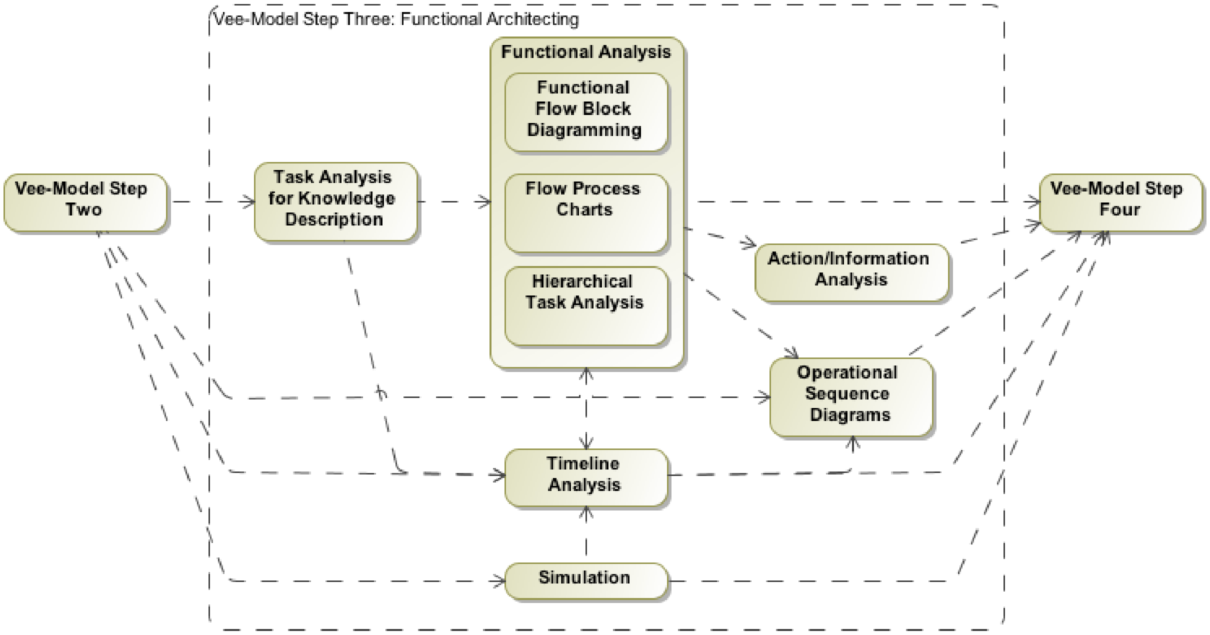

Vee-Model Step Three: Functional Architecting
Functional Analysis
Functional Flow Block Diagramming
Flow Process Charts
Hierarchical Task Analysis
Vee-Model Step Two
Task Analysis for Knowledge Description
Vee-Model Step Four
Action/Information Analysis
Operational Sequence Diagrams
Timeline Analysis
Simulation

Figure 15: Vee-Model Step Three HSI Methods

6.3.1 Unique Methods

6.3.1.1 Task Analysis for Knowledge Description (TAKD)

Method Objective: Define a set of abstracted functions (tasks) that take into account human cognitive components to help facilitate functional architecture development.

Approach: Compile a list of system functions (usually already identified in the previous Vee-model steps) into two separate lists: objects and actions. Abstract these terms into generic actions and objects that are implementation-agnostic (i.e., not specific to any particular task environment). Recast each original function using the abstracted terminology. Stammers, et al. (1990) recommend created a "knowledge representation grammar sentence" for each function, which is comprised of one generic action (function) and up to three generic objects, upon which this action can act. A similar method, Task Action Grammar (TAG), may be used to help develop a "dictionary" of simple functions including their components or features, helping to describe interfaces between functions (Stammers, et al., 1990).

Inputs: System (high-level) functions, functional interfaces, system requirements, concept documents.

Outputs: Abstracted functions, grammar sentences that describe functions (verbs) and the objects (nouns) that they may act upon.

6.3.1.2 Functional Analysis

Method Objective: Develop the system's functional architecture and evaluate architecture alternatives.

Approach: Decompose high-level system functions into progressively lower level functions. Functions are described by verb-noun phrases and may be instantaneous, prolonged, simple or complex (Sanders and McCormick 1993, Ch. 22). While there are many different variations of this method, the following three approaches, Functional Flow Block Diagrams, Flow Process Charts, and Hierarchical Task Analysis, were the main methods identified from the HSI sources and will be described in further detail below.

Functional Flow Block Diagrams, see §6.2.1.2.1 above for a basic description, are iteratively carried out to additional levels of detail as may be necessary to at least determine significant performance requirements. Although the system may be broken into functions, tasks and subtasks to be performed, they are not allocated to any particular system component (e.g., hardware, software, humanware) (McKneely, et al. 2006). A detailed level FFBD is depicted in Figure 16.

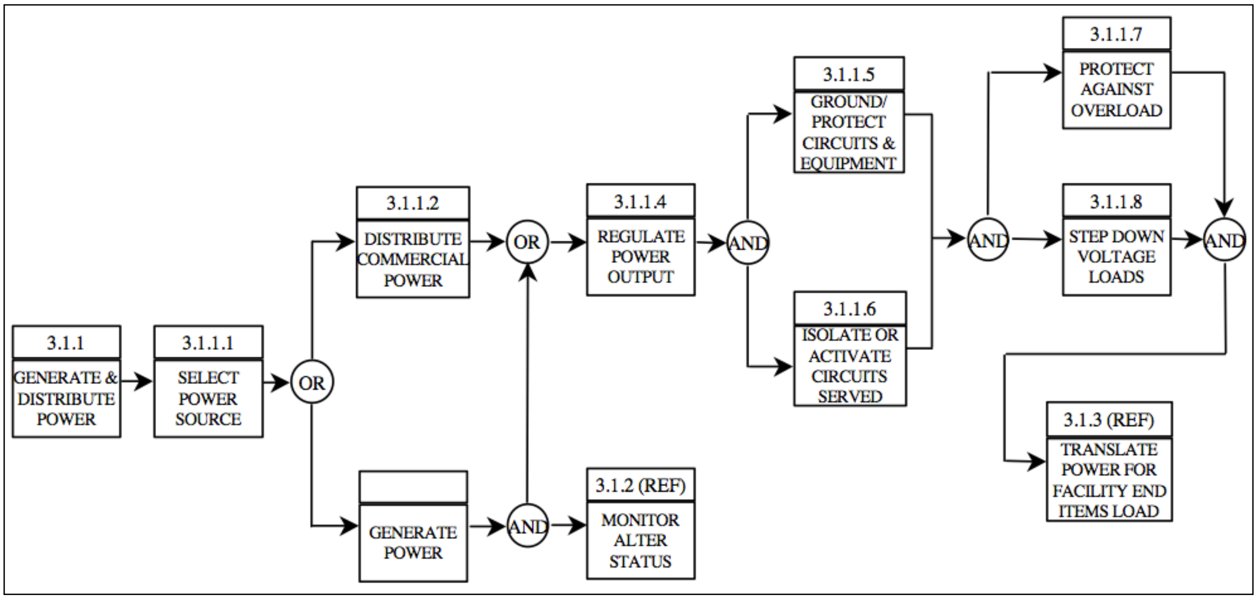


Figure 16: Third-level Functional Flow Block Diagram (DoD 1999, §8.3.5.2.3).

FFBDs may be modified for software-intensive systems where binary decisions are prevalent, in which case, decision-action diagrams are typically used (DoD 1999, §8.3.8 and Chapanis 1996, Ch. 4).

Flow Process Charts, also known as Job Process Charts, are basically plots of a sequence of activities usually with a corresponding time scale (DoD 1999, §8.3.7). Flow chart type symbols are used to represent the different task elements of a particular function (Stammers, et. al, 1990). Flow Process Charts provide a means to represent and analyze a function at any hierarchical level with added detail. Stammers, et al. and the DoD describe the advantages of using Flow Process Charts to model human-computer interactions (e.g., operator station). Figure 17 shows a segment of a Flow Process Chart.

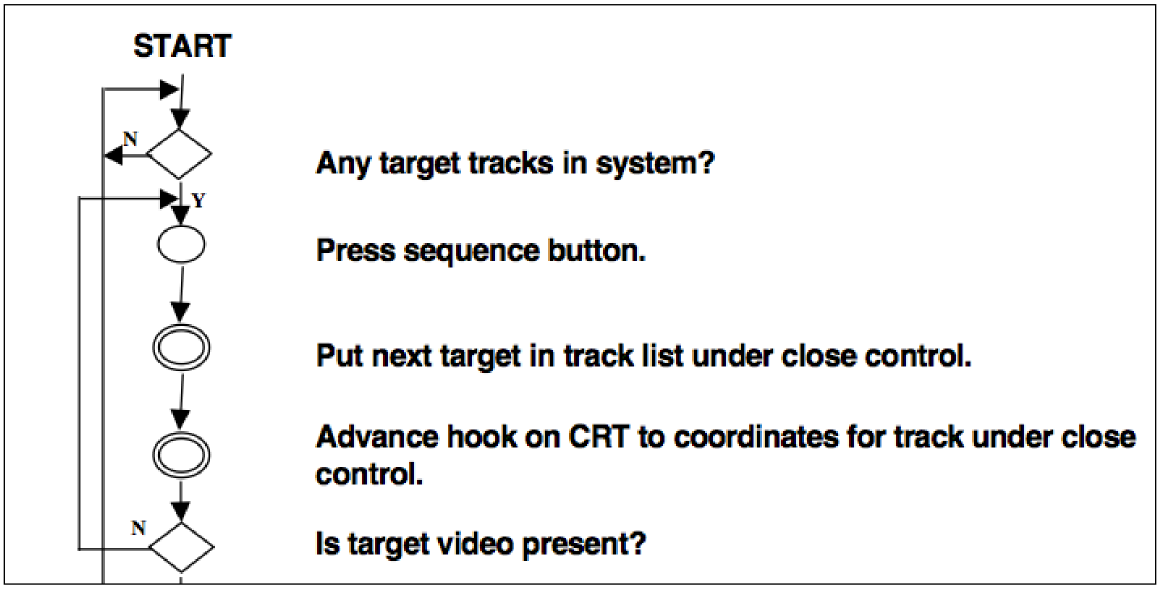

START
N
Any target tracks in system?
Y
Press sequence button.
Put next target in track list under close control.
Advance hook on CRT to coordinates for track under close control.
N
Is target video present?

Figure 17: Flow Process Chart (sample) (DoD 1999, §8.3.7.1).

Although Hierarchical Task Analysis shares the same objective as FFBDs and Flow Process Charts, the main difference is that, unlike other charting methods, it does not necessarily show the order in which functions are carried out. Rather, it portrays the relationships between functions (DoD 1999, §8.3.11.1.4). In a hierarchical approach, a function is analyzed by breaking it into sub-functions, which become increasingly detailed as the hierarchy progresses (Stammers, et. al, 1990). Verb-noun phrases are used to describe each function and interfaces between functions are identified and may be captured in what Stammers, et. al call “plans”, as shown in Figure 18.

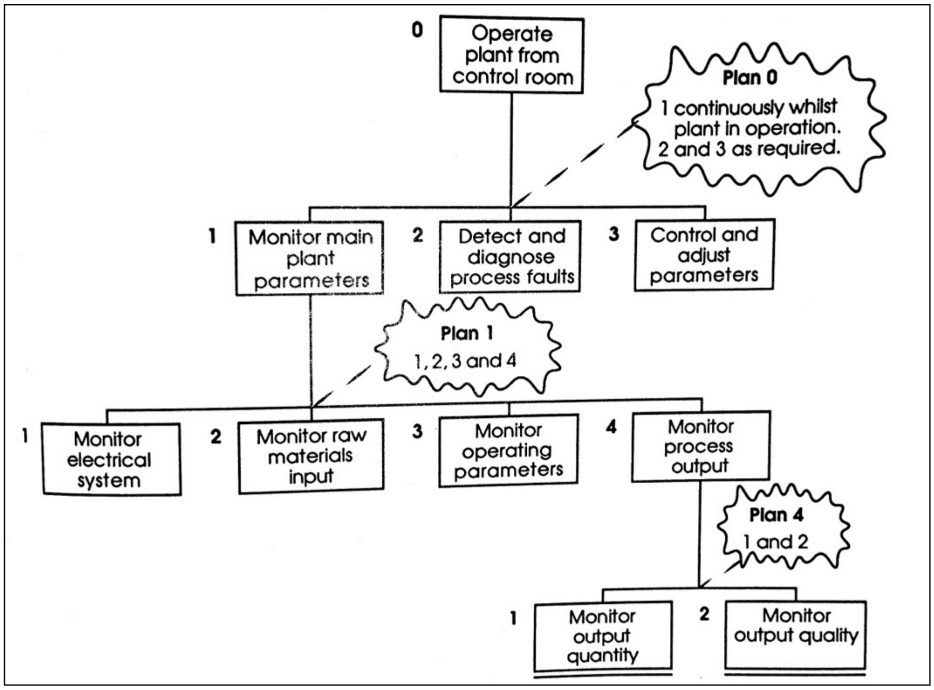

0
Operate plant from control room
Plan 0
1 continuously whilst plant in operation.
2 and 3 as required.
1
Monitor main plant parameters
2
Detect and diagnose process faults
3
Control and adjust parameters
Plan 1
1, 2, 3 and 4
1
Monitor electrical system
2
Monitor raw materials input
3
Monitor operating parameters
4
Monitor process output
Plan 4
1 and 2
1
Monitor output quantity
2
Monitor output quality

Figure 18: Example from a hierarchical task analysis (Stammers, et. al, 1990).

A more detailed version of this method has been formally defined as Integrated Definition for Functional Modeling (IDEF$_0$) (DoD 1999, §8.3.11.1). IDEF$_0$ is a formally defined language for modeling systems by creating a series of interrelated parent-child diagrams that describe the functions and the relationships (e.g., inputs, outputs, controls, and mechanisms) between functions. Figure 19 illustrates this functional decomposition through parent-child diagrams.

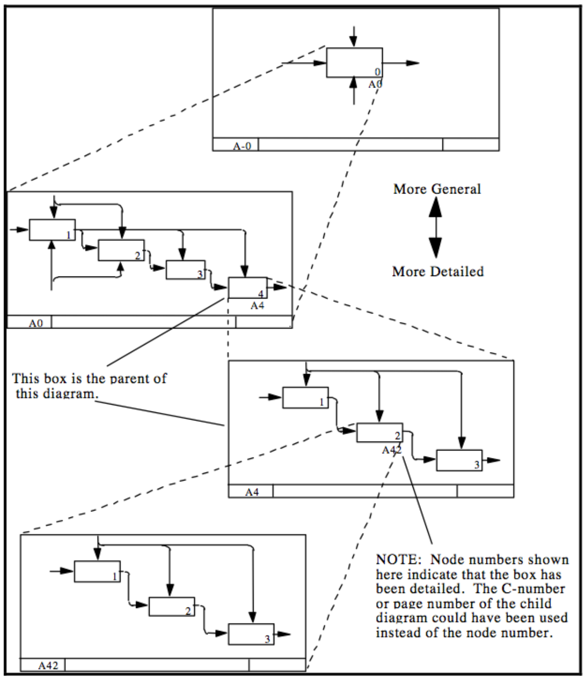

0
A0
A-0
More General
More Detailed
1
2
3
4
A4
A0
This box is the parent of this diagram.
1
2
A42
3
A4
1
2
3
A42
NOTE: Node numbers shown here indicate that the box has been detailed. The C-number or page number of the child diagram could have been used instead of the node number.

Figure 19: System decomposition using multiple IDEF$_0$ graphs (DoD 1999, §8.3.11.1.4).

Inputs: System (high-level) functions, functional interfaces, system requirements, concept documents, functions analyzed by TAKD, knowledge description grammar sentences.

Outputs: Functional architecture, derived/decomposed functions, system element (detailed) requirements (including performance specifications), interface requirements.

6.3.1.3 Timeline Analysis

Method Objective: Verify that temporal relationships among functions are compatible, facilitate workload evaluation, and provide early personnel estimates.

Approach: Plot sequences of functions and their respective durations on a timeline. The DoD states that if time data from a previous system is not available, then using a predetermined time standard is recommended (DoD 1999, §8.3.10.2). Charts and graphs are used to visually identify potential sequencing problems or show overlapping activities (functions) (Chapanis 1996, Ch. 4). *Timeline analysis* may also serve as a complement to *Functional Analysis* by mapping time durations to functions, and *Functional Analysis* may also provide information on potential conflicts as they relate to sequencing of functions (DoD 1999, §8.3.10.4). Figure 20 shows a sample timeline plot.

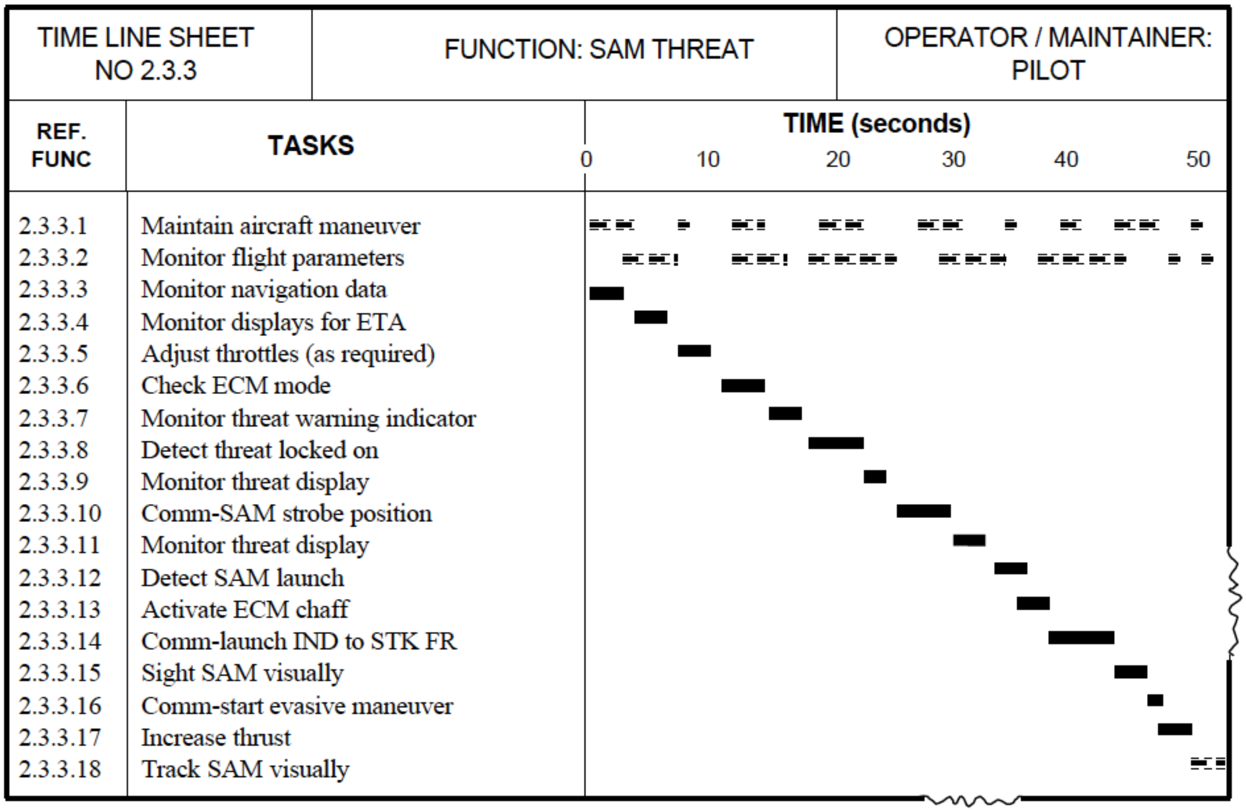


Figure 20: Timeline plot (sample) (DoD 1999, §8.3.10.2).

Inputs: Data from task analyses including *Functional Analysis* (e.g., function durations, sequencing of functions).

Outputs: Timeline plots, temporal relationships among functions, sequencing incompatibilities.

6.3.1.4 Simulation

Method Objective: Predict the performance of the system-of-interest, evaluate alternative architectures, and generate requirements for system "-ilities".

Approach: Prepare models or mockups that evaluate system functions and multiple functional architectures against performance measures. Chapanis (1996) explains that during preliminary design (architectural design), simulations may also be used to identify design requirements for ease of maintenance, develop preliminary specifications (functions) for equipment operability and maintainability, or allow users to experience and receive training on systems that are complex, dangerous, or expensive. Simulations may also provide more detailed and accurate estimates of task overlap or times needed for the execution of long sequences with multiple subtasks, which may serve as inputs to *Timeline Analysis* (DoD 1999, 8.3.10.4).

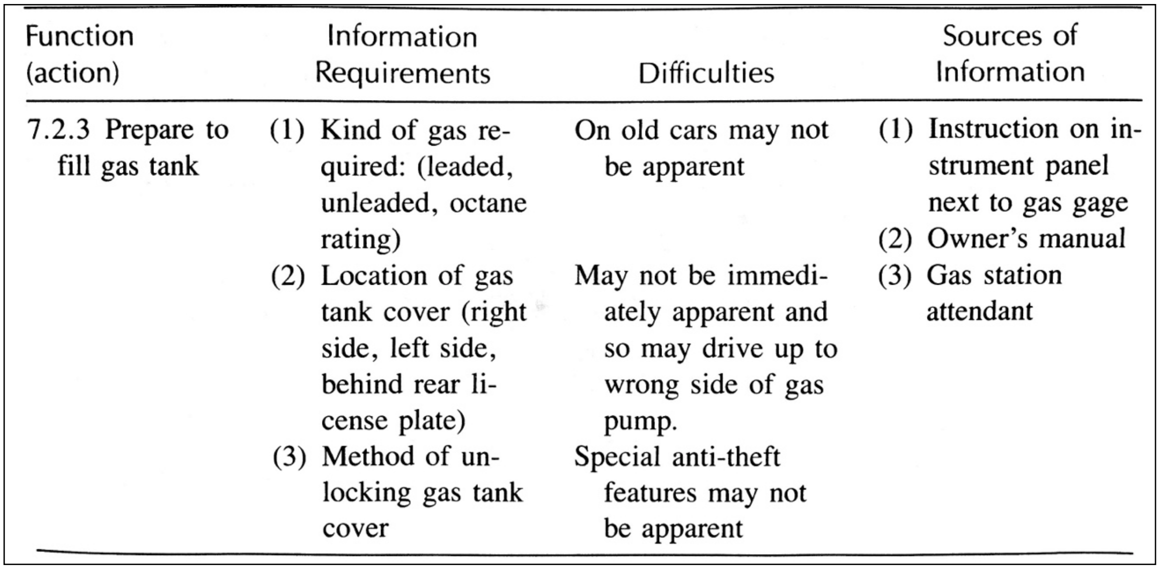
Inputs: Functions, operating procedures, hardware and software to run simulations.

Outputs: Simulated predictions about performance, evaluations of alternative configurations, and evaluations of operating procedures.

6.3.1.5 Action/Information Analysis

Method Objective: Elaborate the system functions to help facilitate functional allocation and allocation trade studies.

Approach: Both Chapanis (1996) and the DoD (1999) indicate that this method should be implemented after *Functional Analysis*, but prior to the allocation of functions, which occurs during *Physical Architecting* (Vee-model Step Four). Using a tabular format, create a list of functions and for each function identify and describe the specific actions necessary to perform that function, information requirements, sources of information, potential problems, error-inducing features and any other relevant commentary. Figure 21 depicts a sample table based on Action/Information Analysis.



Function (action)	Information Requirements	Difficulties	Sources of Information
7.2.3 Prepare to fill gas tank	(1) Kind of gas required: (leaded, unleaded, octane rating)	On old cars may not be apparent	(1) Instruction on instrument panel next to gas gage (2) Owner's manual (3) Gas station attendant
	(2) Location of gas tank cover (right side, left side, behind rear license plate)	May not be immediately apparent and so may drive up to wrong side of gas pump.	
	(3) Method of unlocking gas tank cover	Special anti-theft features may not be apparent	

Figure 21: Example of an Action-Information Analysis: Filling an Automobile Gas Tank (Chapanis, 1996, Ch. 4).

Additional columns may be added if more detail is desired for the preparation of allocation trades (DoD 1999, §8.3.9.2). In particular, *Action/Information Analysis* may help highlight requirements for operator-system interfaces, necessary personnel provisions, and support requirements (Chapanis 1996, Ch. 4).

Inputs: Functions and data from *Functional Analysis*, comments and data from knowledgeable experts.

Outputs: Detailed list of action and information requirements, personnel and support requirements, potential problems, and probable solutions.

6.3.1.6 Operational Sequence Diagram (OSD)

Method Objective: Determine the functional relationships and interactions among system elements (e.g., flow of materials/information, I/O, potential sources of human-system difficulties).

Approach: Develop a graphic representation of operator tasks as they relate sequentially to both equipment and other operators. According to Chapanis (1996), an OSD is a time-based chart with special symbology that combines events, information, actions, decisions, and data. OSDs retain the same basic attributes as Flow Process Charts, but additionally show the flow of information/material through a system and the interactions among stakeholders and/or subsystems that facilitate this flow (DoD 1999, §8.3.6.1). The DoD (1999, §8.3.6.4) also references a similar method, Functional Sequence Diagram, that may be easier to construct, but does not provide as much useful information as the OSD. Figure 22 depicts a sample OSD.

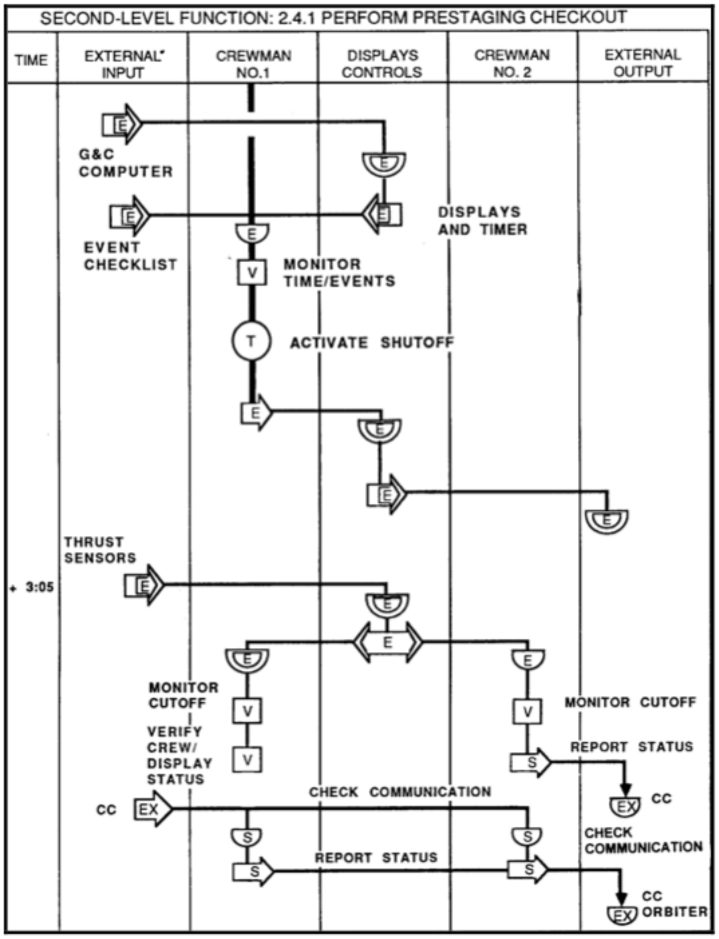


Figure 22: Operational Sequence Diagram (sample) (DoD 1999, §8.3.6.2).

Inputs: System (high-level) functions, functional interfaces, scenarios, timelines, and data from task analyses including *Functional Analysis* (e.g., FFBDs, decision/action diagrams).

Outputs: Time-based chart (OSD) showing functional relationships, flow of materials/information, physical and sequential distribution of operations, I/O, consequences of alternative design configurations, and potential sources of human-system difficulties (Chapanis 1996, Ch. 4).

6.3.2 Unique Methods Summary

Although some aspects of these HSI methods are already identified in the SE process (e.g., FFBD, see §6.2.2), most of them have unique human factors considerations that should be integrated within *Architectural Design (Functional Architecting)* step.

As Stammers, et al. (1990) point out, since systems have become increasingly complex and more highly automated, the role of the human has moved away from the physical ‘hands on’ approach, involving more and more complex cognitive processing. As a result, newer systems must take into account the abstracted functions in order to adequately document the cognitive components of tasks. *Task Analysis for Knowledge Description* provides a method for accomplishing this and enables the systems engineer to build an architecture that is implementation-agnostic, or as INCOSE terms it, a consistent logical architecture.[1]

Timeline Analysis does not appear to be captured within the SE process, but may useful in supplementing *Functional Analysis.* Specifically, time durations could be added to functions and through a timeline plot, potential sequencing problems may be identified.

Although according to the SE process each system architecture option should include a description of the salient features and parameter values of the system elements, it does not go into the level of detail described in the *Action/Information Analysis* method. As a subsequent activity to *Functional Analysis*, this method could prove beneficial in allocating functions to a physical architecture (Vee-model Step 4), by identifying the specific actions and related information elements for functions.

The general principles and methods of *Functional Analysis* (as defined by these HSI methods) are already captured by the SE process. For instance, INCOSE recognizes system hierarchy (i.e., HTA), FFBDs, and IDEF0 diagrams as relevant tools for constructing and evaluating system architectures[2] and identifies the Object-Oriented Systems Engineering Method (OOSEM) and SysML (which may include forms of HTA, FFBDs, and Flow Process Charts) as useful techniques for deriving a logical architecture. [3] However, there are some instances where the author calls out how a method is useful in integrating human factors (e.g., Flow Process Charts are used to model human-computer interactions). Similarly, *Simulation* is accounted for in the SE process and is discussed at length in the INCOSE SE Handbook[4], although it is highlighted that it facilitates life-like training for users and helps to determine functions for equipment operability and maintainability. *Operational Sequence Diagrams* is another example of a method generally accounted for in the SE process (i.e., its use in OOSEM and SysML), but emphasized through the lens of HSI.

The HSI methods mapped to this step in the Vee-model clearly have more potential impact for enhancing the SE process, at least more so than the previous two steps, which may also indicate an increased level of effort expended in this step. If this is the case, this may show that HSI involvement and the corresponding effort to integrate human factors within the systems development begins to increase as the project transitions from the Concept to Development within the system lifecycle.

6.3.3 Models-Based Systems Engineering Support

6.3.3.1 MBSE Functional Architectures Summary

A key step in the system design process, *functional* (and, for object-oriented methods, *logical)* architecting serve to define a “white box” view of the system (Beck, 2011). Specifically,

[1] INCOSE SE Handbook v 3.2.2, §4.3.1.5
[2] INCOSE SE Handbook v 3.2.2, §4.3.2.4
[3] INCOSE SE Handbook v 3.2.2, §4.3.1.5
[4] INCOSE SE Handbook v 3.2.2, §4.12.1

structured and object-oriented analyses are performed to identify and describe system functions, how they relate to each other, and under what circumstances they must be performed, both operational and environmental. Beck (2011) suggests that the best functional architectures are technology agnostic (i.e., they do not address how functions will be performed), thereby leaving physical architecture trade spaces unbiased.

Based on the elaborated context diagram, system scenarios, and system technical requirements identified in the previous step, a functional hierarchy can be developed. In order to capture all relevant functions, this activity is two-fold: 1) decompose system functions (a top-down activity); and 2) compose system functions from sub-system scenarios (a bottom-up activity).

Decomposition typically begins by partitioning the top-level functions from the system ECD into second-level functions, and repeated recursively to develop functions for the third-level, fourth-level, etc. The level of granularity will depend on the amount of detail available from the scenarios. In SysML, a functional hierarchy is captured using a block definition diagram (bdd) in the form of an inverted "tree" structure. An example bdd are other SysML diagrams related to this Vee-model step are provided in Appendix B and C and will be referenced when appropriate throughout this section.

Beck (2011) notes that while the decomposition procedure is somewhat unguided, a partitioning scheme[1] can be applied as outline below. Functions should be identified to:

- interface with each external system
- receive (e.g., format) each system input
- produce each system output
- control the system
- provide required system support services (e.g., manage data, communications, faults)

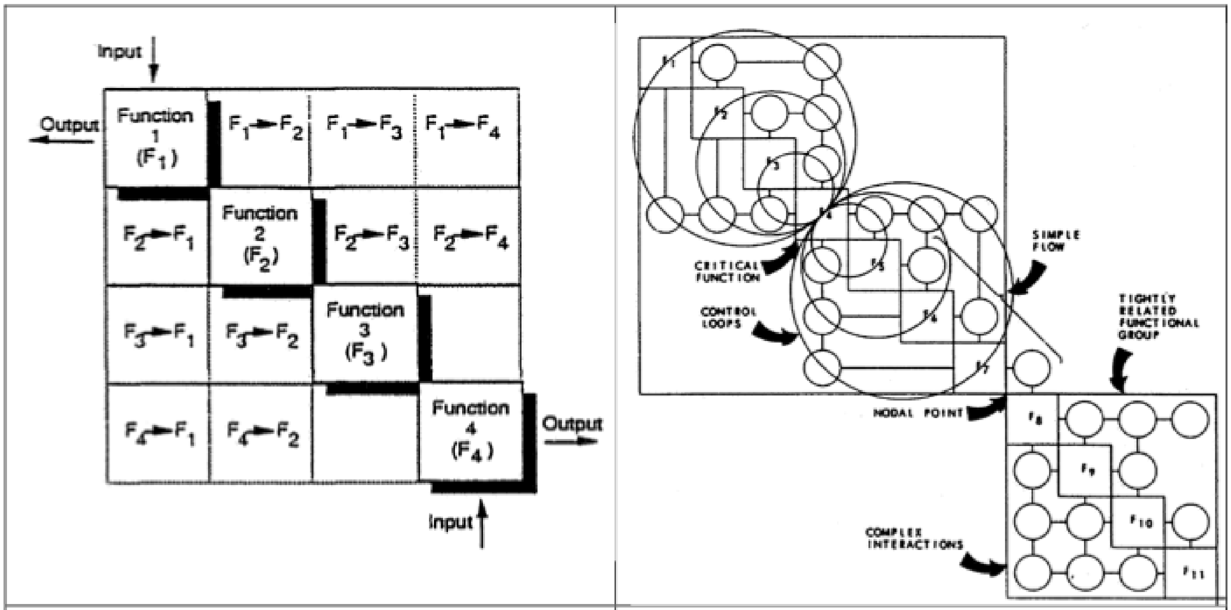
It should also be noted that each function is identified with verb-noun pair and should be at the same level of abstraction as its parent function.

Composition (bottom-up structuring) uses detailed scenarios (usually one that elaborates an activity from the system scenarios) to identify functions. The traditional way to construct these scenarios is through a network diagram, however current MBSE practices would capture these as SysML activity diagrams (act), as shown in Appendix B. These functions are then organized into higher-level functions based on their attributes and this process is repeated successively until a hierarchy is formed from bottom to top. Beck (2011) strongly recommends composition when the system is unprecedented or represents a radical departure from an existing system, although it may be used to augment decomposition as a means to help assure completeness.

Once a functional hierarchy has been developed, the sequential relationships between functions are identified, typically using a method such as FFBD or IDEF0. In SysML, functional flow diagrams may also be represented in activity diagrams, as shown in Appendix B. After defining the sequencing of functions, interfaces are then identified. N^2 diagrams, similar to those depicted

[1] Adapted from Hatley, D. J., and I. A. Pirbhai, *Strategies for Real-Time System Specification,* Dorset House, New York, 1988.

in Figure 23, are used to record these interfaces and highlight where potential regroupings may facilitate more effective system partitioning.



Basic N² chart with feed-forward and feedback interfaces identified.

Key system structural features can be identified based on a completed N² Chart.

Figure 23: N^2 diagram features (NASA, 1995).

Although the OMG SysML specification does not define any table or matrix modeling construct similar to the N^2 diagram, Beck (2011) recommends mimicking similar pair-wise queries on an activity diagram, as shown in Figure 24.

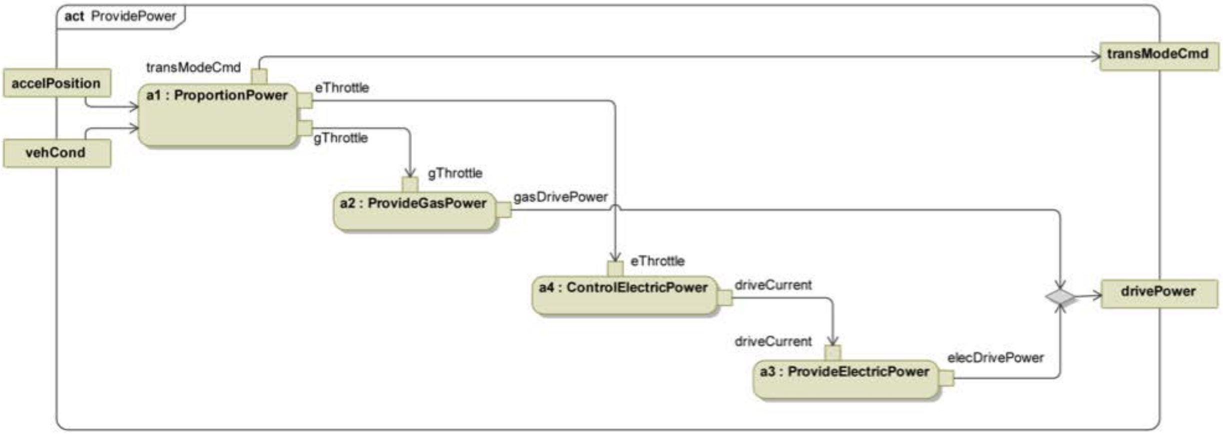


Figure 24: Example SysML diagram comparable to N^2 diagram (adapted from OMG, 2006).

After the functions and interfaces are identified, system technical requirements (e.g., functional, performance, I/O) are allocated to the associated functions. Requirements allocated to the first level of the hierarchy are decomposed and allocated to the appropriate sub-functions until all functions are linked to required behavior (Beck, 2011). Once the system requirements have been mapped to functions, they are analyzed through the use of various modeling or simulation techniques. Beck (2011) suggests that at a minimum it is generally considered necessary to define a functional timeline for the system in order to, for example, identify time-critical design requirements. This type of information could also be captured in a SysML sequence diagram (sd). Other types of evaluation methods may include the Petri net mathematical modeling language (for systems with complex interactions), Business Process Modeling Notation (BPMN), or Event-driven Process Chain (EPC). Most importantly, the results of these analyses must be used to update the system requirements.

Although the functional architecture has been defined, there are some limitations to the structured analysis described above. Specifically, it does not address the management of system parameters (e.g., storage and passing or flow of data), nor does it account for the allocation of non-functional requirements. Beck (2011) states that while this may not be an issue for systems with a smaller functional hierarchy and simple flows, the problem can become unwieldy as functional structure size and complexity grows – especially in understanding and managing the impact of system requirements or design changes.

To alleviate this problem, object-oriented analysis (OOA) is used to develop a *logical* architecture, which combines the system operations and attributes (i.e., parameter statements) into “objects”, or *logical* components. Essentially, functions, flows, and controls are repartitioned into a logical architecture that remains technology agnostic and therefore maintains an appropriate level of abstraction. An example logical hierarchy is depicted in Figure 25.

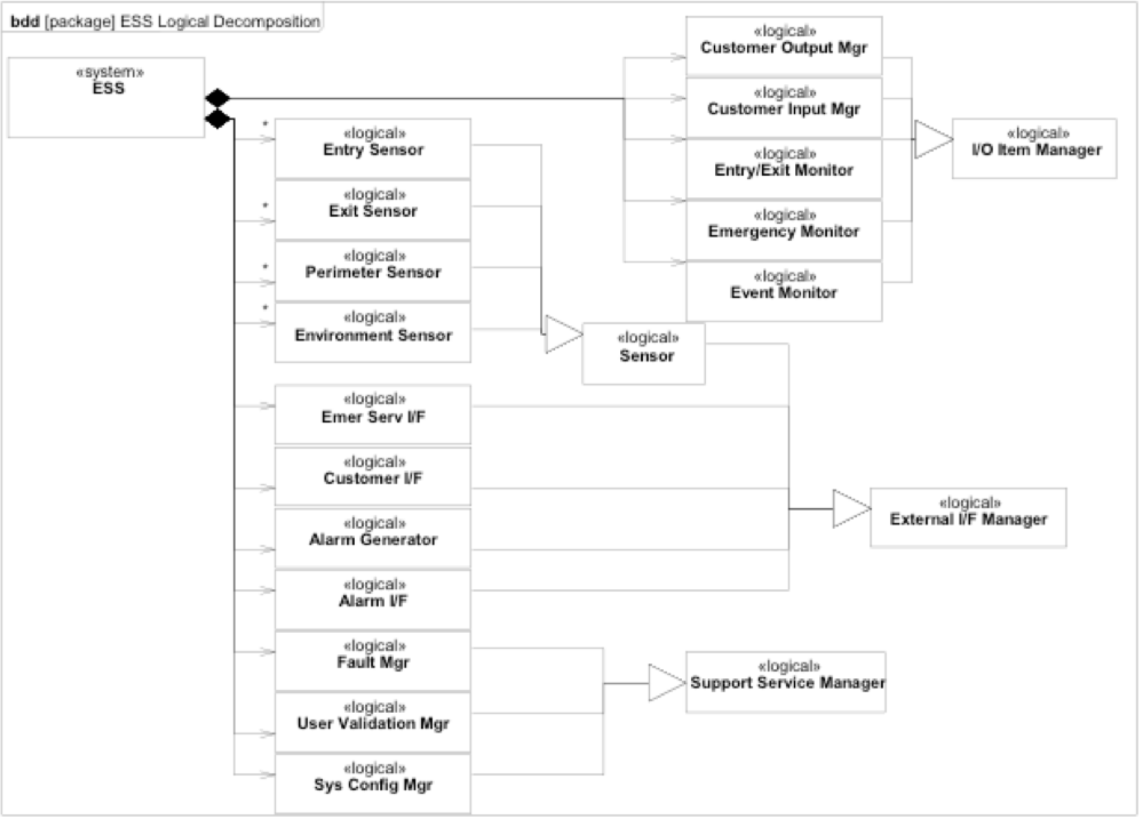

bdd [package] ESS Logical Decomposition
«system» ESS
«logical» Entry Sensor
«logical» Exit Sensor
«logical» Perimeter Sensor
«logical» Environment Sensor
«logical» Emer Serv I/F
«logical» Customer I/F
«logical» Alarm Generator
«logical» Alarm I/F
«logical» Fault Mgr
«logical» User Validation Mgr
«logical» Sys Config Mgr
«logical» Customer Output Mgr
«logical» Customer Input Mgr
«logical» Entry/Exit Monitor
«logical» Emergency Monitor
«logical» Event Monitor
«logical» I/O Item Manager
«logical» Sensor
«logical» External I/F Manager
«logical» Support Service Manager
*
*
*
*

Figure 25: Example logical hierarchy (INCOSE, 2006).

Estefan (2008, §3.2.1) states that developing a logical architecture mitigates requirements changes on the system design and helps to manage technology changes. Another advantage to using a logical architecture is that it helps to reduce the set of criteria to consider when evaluating the physical solution space (e.g., physical architecture) by ruling out potential physical architectural alternatives based on basic partitioning criteria applied to the logical architecture.

Partitioning criteria used to help group functions together at some level in a logical hierarchy include modularity (i.e., maximize cohesion and minimize coupling), similarity (i.e., related functions), changeability (i.e., functions that have a high likelihood of changing), and constraints (i.e., functions that share common constraints). Similar to the structured analysis, the logical hierarchy may be developed via decomposition (i.e., allocating functions to logical components, working from top down) and/or composition (i.e., assign a one-for-one logical element to each leaf-level function, then group them based on the partitioning criteria). Interface diagrams are updated by adding "swim lanes" that allocate functions to logical components, as shown in Figure 26.

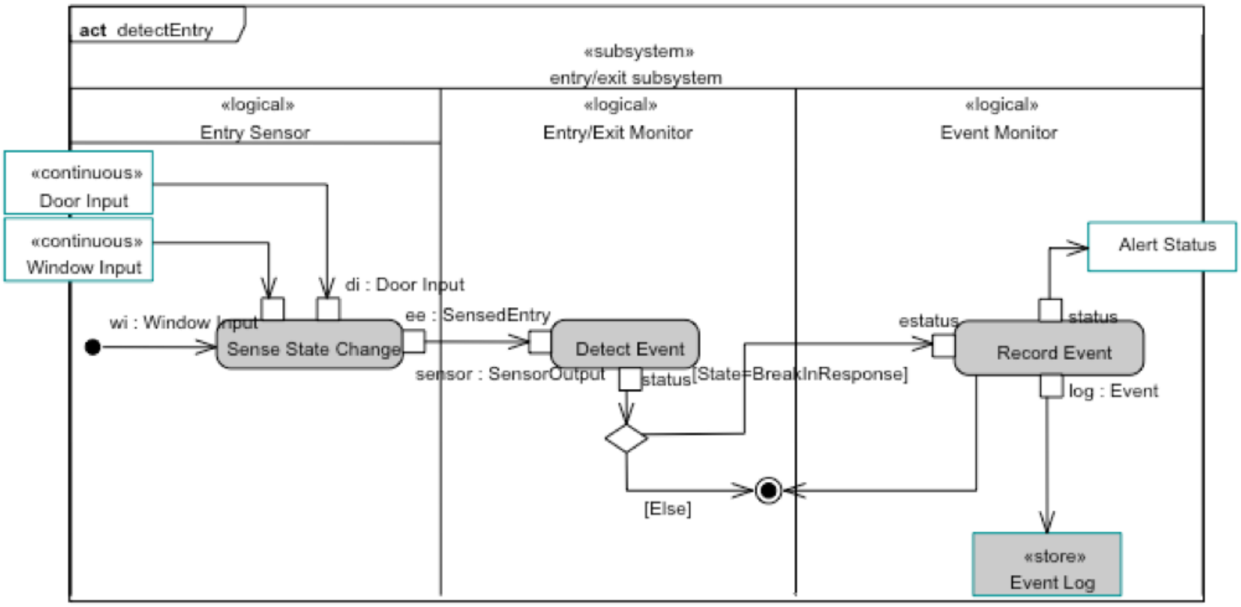


Figure 26: Example logical interface diagram (INCOSE, 2006).

Once these logical diagrams have been developed, requirements allocated to the functional architecture are now allocated to the appropriate (function performing) logical components and should include the following types: functional, performance, I/O, control, store, and design constraints. (Beck, 2011). Logical flows are then analyzed through timeline analyses via sequence diagrams and additional modeling and simulation may be used to evaluate system parameters. The results of these analyses will then be used to update the system requirements.

6.3.3.2 Evaluation of HSI Methods Against MBSE

6.3.3.2.1 Task Analysis for Knowledge Description

Given that the architectures developed in this step, both functional and logical, should be technology agnostic and therefore have a certain level of abstraction, TAKD would serve as an excellent method for defining functions and/or logical components. The structure of the knowledge representation grammar sentence closely resembles object-oriented analysis in that functions (operations) are grouped with objects (attributes) upon which they can act. In fact, these grammar sentences may even serve to define the components in the logical architecture. Moreover, Task Action Grammar would be a useful starting point to N^2 diagramming by providing a list of functions and their associated features, helping to describe potential interfaces.

6.3.3.2.2 Functional Analysis

As stated previously, this method does not offer anything new to the SE process, nor does it identify a method that is not already captured by MBSE. However, the strong similarities between the activities within this HSI method and those described in this MBSE step indicate that a models-based approach to systems development may more effectively take into account human factors than the "classical" SE process. For example, Hierarchical Task Analysis, FFBDs/IDEF0 diagrams, and Flow Process Charts are key activities within MBSE and can be modeled using SysML diagrams (e.g., block definition diagram for HTA, activity diagram for FFBDs/IDEF0 diagrams and Flow Process Charts). With these methods already incorporated to MBSE, successful integration of human factors within this activity merely becomes a matter of the level of HSI emphasis within the project.

6.3.3.2.3 Timeline Analysis

Although this method was not accounted for in the SE process, it is identified as one of the most important analyses in the MBSE Functional Architectures Step. In addition to identifying time-critical design requirements, *Timeline Analysis* helps to ensure that functional sequencing is correct and, with regards to human factors, facilitates workload evaluation and provides early personnel estimates.

6.3.3.2.4 Simulation

Like *Functional Analysis*, this method is already described within MBSE. For example, simulation, through various tools, may be used to evaluate system parameters, which may help identify or update performance requirements. Given its existing integration within MBSE, simulations focused on operability and maintainability should be emphasized.

6.3.3.2.5 Action/Information Analysis

This method would be probably most useful in identifying how the functional architecture should allocate to the logical architecture. Specifically, actions and information requirements identified through this HSI method could be incorporated into the partitioning criteria used to group functions within the logical hierarchy.

6.3.3.2.6 Operational Sequence Diagram

Although this method is described in this MBSE step as useful for diagramming timeline analyses, its application in determining the functional relationships and interactions among system elements is not adequately covered. Additional OSD-based analyses that include detailed information about events, actions, and decisions related to operator-equipment or operator-operator interactions would be beneficial in identifying functional relationships and interactions among logical components. OSDs may also help define partitioning criteria, clarifying how functions should be mapped to logical components.

6.3.4 Models-Based Systems Engineering Support Summary

While some of these HSI methods have been accounted for within MBSE, the majority offer important enhancements:

1. Use *TAKD* to:
   a. Define (abstracted) functions;
   b. Use knowledge representation grammar sentences to define logical components;
   c. Develop a list of functions via the TAG method to identify potential interfaces within the N^2 diagram.
2. Given the existing integration of *Functional Analysis* within MBSE, place additional emphasis on human factors considerations.
3. Use Timeline Analysis to:
   a. Identify time-critical design requirements;
   b. Verify that the temporal relationships among functions are compatible;
   c. Facilitate workload evaluation and provide early personnel estimates.
4. Focus on operability, maintainability, and other human factors non-functional requirements during simulations.
5. Use *Action/Information Analysis* to help define partitioning criteria while allocating functions to the logical architecture.
6. Use *Operational Sequence Diagrams* to:
   a. Identify functional relationships and interactions among logical components;
   b. Define partitioning criteria (i.e., how functions should be mapped to logical components).

Based on the fact that the HSI methods identified use several of the modeling tools used in MBSE, human factors can be more effectively integrated into systems development when MBSE is implemented when compared to the “classical” SE process.

6.4. Architectural Design: Physical Architecting

As stated in the previous section, although *Architectural Design* has been decomposed into *Functional Architecting* and *Physical Architecting* in this paper, it is summarized as one step to maintain consistency with ISO/IEC 15288 and the INCOSE SE Handbook. For a summary of the *Architectural Design* process, please refer to §6.3. A detailed description of *Physical Architecting* is provided in the subsequent MBSE section.

Based on the mapping analysis, twenty-one methods were identified as applicable to *Physical Architecting* (see Table 4 for a comparison matrix).

Table 4: Vee-model Step Four: HSI Methods Comparison

HSI Methods	Function Allocation	Task Design and Analysis	Perf, Workload, & Manning Lvl Est.	Conduct SH Analysis for Allocation of Functions	Consider Human and Computer Capabilities	Assess Impact of Allocation of Function on Task Perf and Job Satisfaction	Allocation of Functions	Human Perf Requirements	Task Description & Analysis	Simulation	Functional Allocation	Activity Analysis	Task Analysis	Fault Tree Analysis	Failure, Modes, and Effects Analysis (FMEA)	Controlled Experimentation	Workload Assessment	Cognitive Task Analysis	Function Allocation Trades	Workload Analysis	Task Description/Analysis
Function Allocation		■	■	⊙	⊙	⊙	⊙	■	■	■	⊙	■	■	■	■	■	■	■	⊙	■	■
Task Design and Analysis			■	■	■	■	■	■	⊙	■	■	■	⊙	■	■	■	■	⊙	■	■	⊙
Perf, Workload, & Manning Lvl Est.				■	■	■	■	⊙	■	⊙	■	⊙	■	■	■	⊙	⊙	■	■	⊙	■
Conduct SH Analysis for Allocation of Functions					⊙	⊙	⊙	■	■	■	⊙	■	■	■	■	■	■	■	⊙	■	■
Consider Human and Computer Capabilities						⊙	⊙	■	■	■	⊙	■	■	■	■	■	■	■	⊙	■	■
Assess Impact of Allocation of Function on Task Perf and Job Satisfaction							⊙	■	■	■	⊙	■	■	■	■	■	■	■	⊙	■	■
Allocation of Functions								■	■	■	⊙	■	■	■	■	■	■	■	⊙	■	■
Human Perf Requirements									■	⊙	■	⊙	■	■	■	⊙	⊙	■	■	⊙	■
Task Description & Analysis										■	■	■	⊙	■	■	■	■	⊙	■	■	⊙
Simulation											■	⊙	■	■	■	⊙	⊙	■	■	⊙	■
Functional Allocation												■	■	■	■	■	■	■	⊙	■	■
Activity Analysis													■	■	■	⊙	⊙	■	■	⊙	■
Task Analysis														■	■	■	■	⊙	■	■	⊙
Fault Tree Analysis															⊙	■	■	■	■	■	■
Failure, Modes, and Effects Analysis (FMEA)																■	■	■	■	■	■
Controlled Experimentation																	⊙	■	■	⊙	■
Workload Assessment																		■	■	⊙	■
Cognitive Task Analysis																			■	■	⊙
Function Allocation Trades																				■	■
Workload Analysis																					■
Task Description/Analysis																					

Same	⊙
Different	■
McKneely, et al., 2006	🟦
Stanton, et al., 2012	🟨

Chapanis, 1996	□
Sanders and McCormick, 1993	🟥
DoD, 1999	🟧

After comparative analysis, four unique methods were identified: 1) Function Allocation 2) Task Description and Analysis; 3) Performance, Workload and Manning Level Estimation; and 3A) Failure Analysis (a sub-method for determining performance impacts).

The methods in this step are sequential, beginning with the Function Allocation, then Task Description and Analysis, and finally Performance, Workload, and Manning Level Estimation, which includes Failure Analysis. As shown in Figure 27, these methods are also iterative (Sanders and McCormick 1993, Ch.22) to the extent that the results of the latter three methods necessitate a change in the allocation of functions.

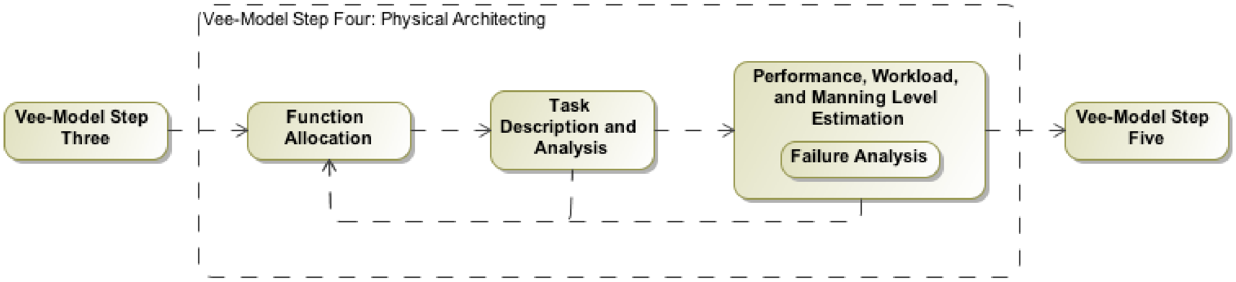

Vee-Model Step Four: Physical Architecting
Vee-Model Step Three
Function Allocation
Task Description and Analysis
Performance, Workload, and Manning Level Estimation
Failure Analysis
Vee-Model Step Five

Figure 27: Vee-Model Step Four HSI Methods

6.4.1 Unique Methods

6.4.1.1 Function Allocation

Method Objective: Assign each system function, action, and decision to hardware, software, or humanware such that the resulting physical configuration maximizes total system performance and effectiveness.

Approach: Allocate each system function to a physical component (e.g., hardware, software, humanware). Although allocation decisions may be predetermined by constraints established during Mission Analysis and Requirements Analysis (McNeely, et. al 2006), also known as *Mandatory Allocation* (Sanders and McCormick 1993, Ch. 22) they are typically determined by comparing alterative configurations in terms of their effectiveness in performing a given function (Chapanis 1996, Ch. 4). There are many techniques for making allocation decisions. Sanders and McCormick list a few (1993, Ch. 22, *A Strategy for Allocating Functions)*, one of which defines a "decision space" as an aid in making allocation decisions (see Figure 28).

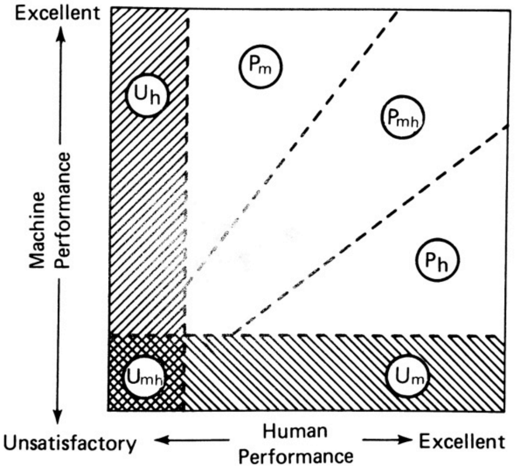


Figure 28: Human-Machine Allocation Decision Space (Sanders and McCormick 1993, Ch. 22)

The DoD identifies other techniques including trial-and-error, Fitts lists, and design evaluation matrix (DoD 1999, §8.3.12.1). Chapanis (1996, Ch. 4) recommends establishing weighting criteria for comparing physical configurations and using a simple rating scheme for making allocation decisions. While these techniques are helpful, they can be dated and must be used judiciously (McNeely, et. al 2006). Sanders and McCormick (1993, Ch. 22) point out a number of limitations in using such comparisons, such as the rapid evolution of machines relative to humans, the inability to account for costs related to allocation decisions, and the lack of consideration of social, cultural, or political issues. Dynamic allocation may also be used, in which allocation decisions are made real-time by the operator (e.g., autopilot on aircraft and cruise control in automobiles) (Sanders and McCormick 1993, Ch. 22). Finally, the impact of allocation decisions on total system performance is assessed. For any allocations that have a significant negative effect on performance, an alternative allocation should be determined (Stanton, et. al, 2012). Likewise, results from the three subsequent methods of this step are used to reallocate functions as necessary.

Inputs: Functional architecture, functional flow analyses, technical basis reports, known human capabilities and limitations, task analysis results, and performance, workload and manning level estimations.

Outputs: Physical architecture, and allocation justifications including evaluation matrices, weighted comparisons, Fitts lists, etc.

6.4.1.2 Task Description and Analysis

Method Objective: Record and analyze how the human interacts with the system.

Approach: List and describe all tasks, subdividing tasks into subtasks, and record related supplementary information. At a minimum details should be provided for: information requirements; evaluations and decisions that must be made; task times; operator actions; and environmental conditions (Chapanis 1996, Ch. 4). McNeely, et al. (2006) provide a similar list: task cues, user decision/action, information required to support the decision/action, and mechanisms to implement the results of the decision/action. The DoD (1999, §8.3.2.3) recommends analyzing tasks that are: potentially hazardous; time consuming; difficult; show potential for improvement; or required by the procuring activity. Human-machine interactions are articulated (McNeely, et al., 2006) and are often reflected in an operational sequence diagram[1] (Sander and McCormick 1993, Ch. 22).

Inputs: Physical architecture, allocation justifications, and expert opinion from users of similar systems.

Outputs: Ordered list of human-specific tasks including supplementary information.

6.4.1.3 Performance, Workload and Manning Level Estimation

Method Objective: Assess workloads and related manning and training requirements in order to predict and optimize system performance.

Approach: Determine human performance requirements for each function/task if not previously identified. These may include required accuracy, speed, or time necessary to develop performance proficiency and user satisfaction (Sanders and McCormick 1993, Ch. 22). Next, appraise operator task loadings. These are typically calculated by estimating the time required to perform a task, divided by the time available or allotted to perform it (Chapanis 1996, Ch. 4). A workload profile, as depicted in Figure 29, may be used to highlight unbalanced workload distributions (DoD 1999, §8.3.13.1).

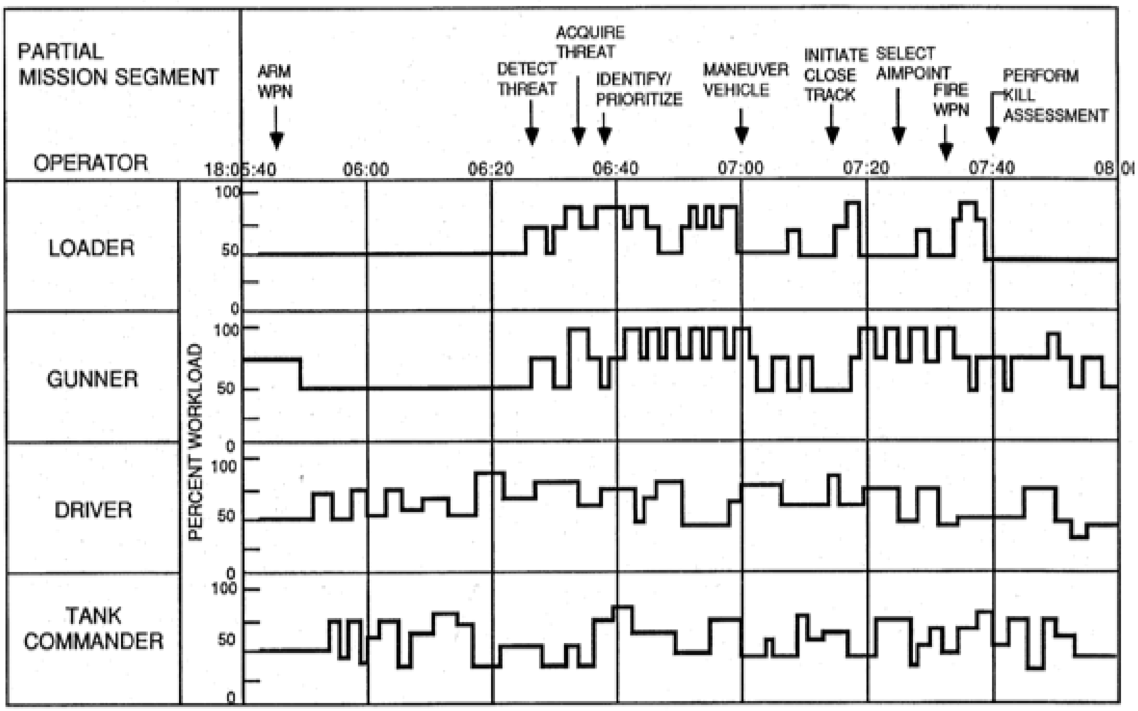


Figure 29: Workload analysis profile (sample) (DoD 1999, §8.3.13.1).

[1] See §6.3.1.6 for an overview of Operational Sequence Diagram (OSD).

Estimates of manning (personnel) and training requirements are also determined, typically by direct observation when a precedent system exists. Chapanis (1996, Ch. 4) refers to this as *Activity Analysis*, which may yield other important information such as assessments of stress levels and indications where changes in procedure would improve performance. Simulation and controlled experimentation provide mechanisms for assessment of system performance with the defined human role (McNeely, et al., 2006). Simulations may also help evaluate alternative configurations, evaluate operating procedures, provide training, and identify mismatches between personnel and equipment (Chapanis 1996, Ch. 4). Controlled experimentations help define relationships or correlations between variables as well as differences between alternative configurations, procedures, or environments (Chapanis 1996, Ch. 4). These estimates are then evaluated against the performance requirements. If workload and manning estimates are within acceptable performance limits, hardware and software detailed design may begin. If workload overloads/underloads exist or if manning levels are disproportionate, reallocation of functions is necessary to meet required performance levels.

Inputs: MOPs, task time/frequency/precision data, and expert opinion from users of similar systems.

Outputs: Workload/manning/training estimates, workload profiles, performance evaluation data, and areas that require reallocation.

6.4.1.3.1 Failure Analysis

Method Objective: Predict human errors and determine the resulting impacts to system performance.

Approach: This method is comprised of two ‘sub-methods’, *Fault Tree Analysis* and *Failure Modes and Effects Analysis (FMEA)*. Although different in their respective approaches, they both focus on anticipating operator/maintainer mistakes with the aim of designing against those mistakes for the system-of-interest. Specifically, *Fault Tree Analysis* begins with an undesirable event or failure and attempts to determine those combinations of events and circumstances that could lead to it (Chapanis 1996, Ch. 4). As shown in Figure 30, probabilities of various undesirable events and the sequences that would produce those undesirable events are depicted in a tree-like structure.

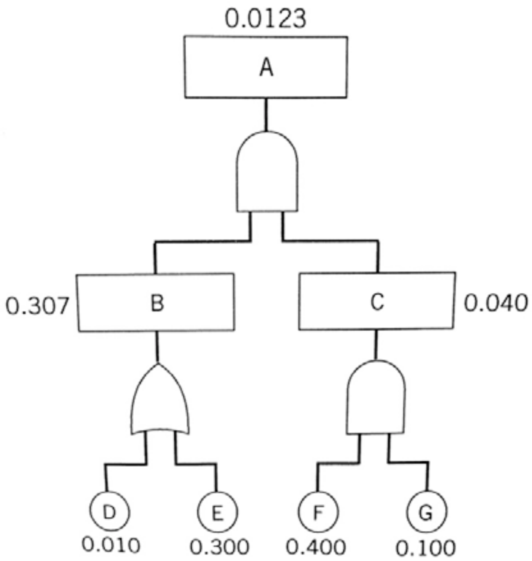


Figure 30: Fault tree (sample) (Chapanis 1996, Ch. 4).

Conversely, *FMEA* begins with components (e.g., human operator) and deduces the consequences of a failure in one or more of those components (Chapanis 1996, Ch. 4). A list of human failures that have major impacts on system performance is produced along with the probabilities of failure occurrence. Based on this information reallocation decisions are made to reduce the probability of serious system failures.

Inputs: Results from task analyses, functional flow analyses, action/information analyses, and human reliability data.

Outputs: Probabilities of undesirable events and system failures due to human errors and areas that require reallocation.

6.4.2 Unique Methods Summary

While the general concepts of these HSI methods are touched upon within the SE process, it does not go into the same level of detail necessary for integrating human factors within the systems development.

Although the SE process describes allocating functions to a physical architecture and notes the importance of taking human factors into account.[1], it does not provide the same level of detail as the *Function Allocation* method. Since allocation decisions made in this step will determine how the human interacts with the system, this HSI method should be integrated into the SE process. The HSI sources related to this method offer various approaches and tools that can help the systems engineer make informed allocation decisions while providing supporting documentation (e.g., decision space chart, evaluation matrix, ‘Fitts’ list) that may serve as a basis for selection justification. These artifacts are also crucial in the feedback loops between these HSI methods,

[1] ISO/IEC 15288:2002, §5.5.4.3 d)

which the SE process describes as necessary in order to ensure proper allocation, requirements satisfaction, and manufacturing compliance (INCOSE SE Handbook v.3.2.2, §4.3.2.8).

The activities related to *Task Description and Analysis* are identified within the SE process, although described more generally.[1] The advantage of this HSI method is that it focuses on human tasks and analyzes how the human interacts with the system. However, it should be noted that INCOSE makes reference to task analyses and their importance in understanding human capabilities.[2] That being said, specific techniques, such as those described in *Task Description and Analysis* provide needed detail about how to conduct these analyses.

While the SE process identifies[3] key activities within *Performance, Workload and Manning Level Estimation*, such as establishing performance requirements and evaluating design solutions against these requirements, it does not offer a specific method for doing this. In addition, the HSI method provides multiple ways for assessing humanware allocations to ensure workload and manning levels are not disproportionate, whereas the SE process does not address this.

The sub-methods described in *Failure Analysis* are typically used by safety engineers for evaluating system errors and therefore are described within the SE process. For example, INCOSE dedicates a section to elaborating *Failure Modes and Effects Analysis (FMEA).*[4] However, since the *Failure Analysis* method as described herein adapts these methods to deal specifically with human errors, it would be beneficial to consider these sub-methods as defined by HSI during system development.

The HSI methods mapped to *Physical Architecting* focus on one of the most important aspects to HSI: defining and evaluating how the human interacts with the system. Similar to the previous step, *Functional Architecting*, there is an increased level of effort in analyzing and evaluating human factors within the system, especially due to the iterative nature of reallocating functions. As the life cycle continues to progress through development, it is crucial to ensure that sufficient human factors analyses are conducted and incorporated into the physical architecture prior to *Implementation* activities in order to avoid failures across human-machine interfaces, resulting in costly changes.

6.4.3 Models-Based Systems Engineering Support

6.4.3.1 MBSE Physical Architectures Summary

After structured and object-oriented analyses have been conducted to produce both functional and logical architectures, the next step is to define one or more physical architectures to which functions or logical components will be allocated. Based on technical screening criteria, physical architecture alternatives are down-selected for further analysis. These alternatives should also explore various component technologies, which will be used to conduct trade studies. Physics-

[1] INCOSE SE Handbook v.3.2.2, §4.3.2.3
[2] INCOSE SE Handbook v.3.2.2, §9.12.2.1
[3] ISO/IEC 15288:2002, §5.5.4.3 f)
[4] INCOSE SE Handbook v.3.2.2, §9.1.2.1

based models are then used to evaluate the remaining architectures and serve as a means to conduct performance-based evaluations to help make a final selection.

Prior to allocating functions to the physical architecture, partitioning criteria, from which functions will be grouped, must be defined (Beck, 2011). These may include:

- COTS, reuse, and other design constraints
- Physical or environmental
- Safety and security
- Subcontractor or development responsibility.

Once the portioning criteria have been established, a generic physical hierarchy (GPA) is developed. Beck (2011) intends *generic* to mean that the partitioning is made without any specification of the performance characteristics of the physical resources that comprise each element. For every logical (or functional) architecture, a generic physical architecture is defined, usually by decomposition or composition. Figure 31 shows an example GPA.

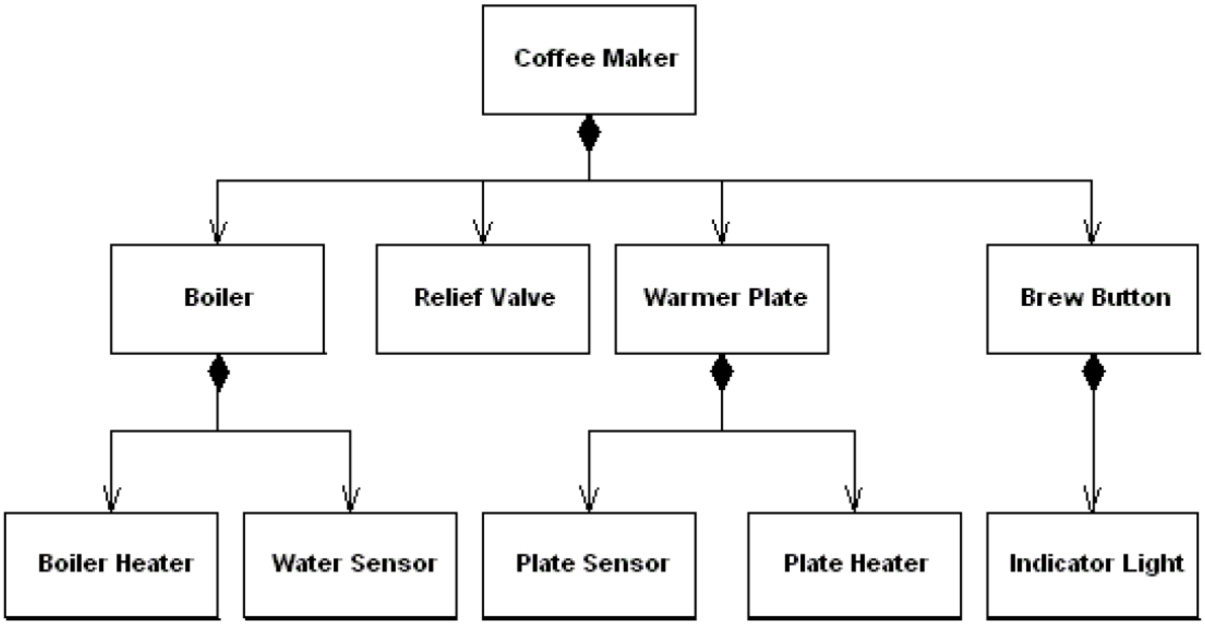


Figure 31: Generic physical architecture (sample) (Weirich, 1999).

With the GPA defined, all functions or logical components, including system design constraints, are allocated to each element (node) within the GPA, producing one-to-one or one-to-many relationships. For each leaf node within the hierarchy, different solutions that are likely to satisfy the allocated requirements are generated. Beck (2011) notes that by identifying a relatively large, creative number of options, there is a greater chance that the best alternatives will be considered in the final analysis. These options are documented in *Technical Basis Reports* (TBRs).

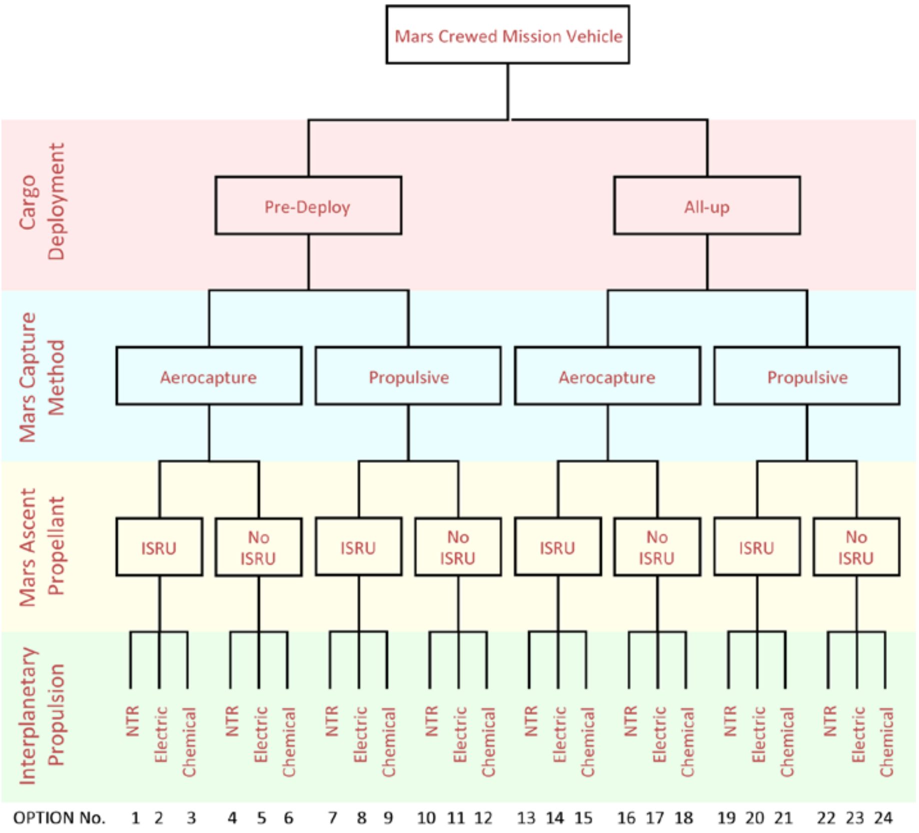
The list of component alternatives and the GPA are then combined to form what Beck (2011) calls *instantiated* physical architectures. These architecture instantiations identify how specific technologies are used to implement the system, and are frequently represented using two techniques: the *morphological box* and the *trade tree*.

A morphological analysis (MA) divides a problem into segments and identified at least two solutions for each segment (Beck, 2011). This is typically portrayed as a table with the columns representing the problem segments (e.g., components of the GPA) and the rows filled with the alternate specific instantiations for each component, as shown in Table 5. The total number of alternatives is given by multiplying the number of options in each column (e.g., using the table below, there are 2x5x4x4x2=320 possible hammers defined).

Table 5: Morphological Box for Hammer (Beck, 2011).

Handle Size	Handle Material	Striking Element	Weight of Hammer Head	Nail Removal Element
8 inches	Fiberglass with rubber grip	1-inch-diameter flat steel	12 oz.	Steel claw at nearly a straight angle
22 inches	Graphite with rubber grip	1-inch-diameter grooved steel	16 oz.	Steel claw at a 60-degree angle with handle
	Steel with rubber grip	1.25-inch-diameter flat steel	20 oz.	
	Steel I-beam encased in plastic with rubber grip	1.25-inch-diameter grooved steel	24 oz.	
	Wood			

A trade tree uses a hierarchy structure, in which each branch level represents a problem segment and each node on that branch represents a proposed segment solution (Beck, 2011). Each line through the root rode to leaf node represents an alternative, with the total number of alternatives given by the number of leaf nodes, as shown in Figure 32.



NTR- Nuclear Thermal Rocket
Electric= Solar or Nuclear Electric Propulsion

Figure 32: Trade tree for NASA Manned Mars Mission (adapted from Guerra, 2008).

Once a set of alternatives has been identified, the next step is to begin the down-selection process. Eliminating infeasible alternatives (e.g., those that have incompatible technologies) should be the first activity in this process. Beck (2011) recommends conducting pairwise comparisons between all component alternatives similar to those used in an upper triangular matrix or a Quality Function Deployment correlation matrix. Within a trade tree, branches may be “pruned” to eliminate non-workable solutions. In addition, a preliminary screening should be conducted to help narrow down the set of alternatives even further before developing models, which require extensive resources to generate and analyze. Examples of technical screening criteria include: technical maturity, similarity of alternatives, flexibility, reliability, etc.

The output of these activities should be a small set of instantiated physical architectures. Each alternative is formally documented in the form of either a *Technical Description Document* (TDD) or model-based concept description, either of which should be used as living documents. Beck (2011) explains that TDDs will usually include graphical representations of the physical architectures they describe, as shown in Figure 33. In SysML, these diagrams may be captured in block definition diagrams (bdd) that depict system hierarchy, internal block diagrams (ibd) that depict interfaces, and sequence diagrams (sd) to depict scenarios. At this point, component-level requirements should be documented and trace to the system-level requirements.

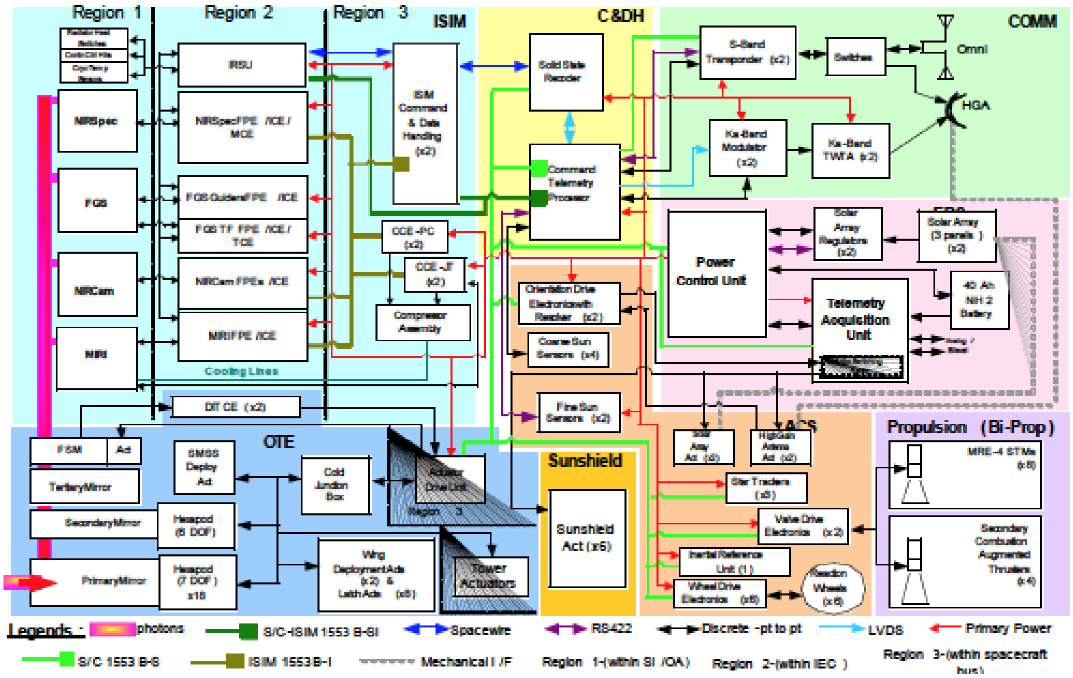


Figure 33: Physical architecture represented in a block diagram (Guerra, 2008).

The final step in developing physical architectures is to generate physics-based models that measure the effectiveness of the alternatives. Beck (2011) asserts that these models form the “backbone” for assessing each physical architectures performance against system requirements after suitable component-level models are developed (completed in later steps of the Vee-model).

6.4.3.2 Evaluation of HSI Methods Against MBSE

6.4.3.2.1 Function Allocation

Although this MBSE step describes allocation as a necessary activity in developing physical architectures, it does not provide the level of detail elaborated in the *Function Allocation* method. Specifically, by integrating this HSI method into MBSE, the systems engineer would have multiple approaches available for allocating functions or logical components to the physical architecture. For example, Fitts lists, design evaluation matrices, and decision space diagrams may be used to supplement partitioning criteria in making allocation decisions. Dynamic allocation may also be used, which increases architecture flexibility although may introduce added complexity. Moreover, since *Function Allocation* is clearly focused on ensuring that functions are appropriately allocated to the human element, integrating this HSI method into MBSE helps to resolve allocation decisions where, for example, a function could be performed by either a human or by a machine necessitating further analysis to make the proper determination.

6.4.3.2.2 Task Description and Analysis

A key activity in the MBSE method is the identification of component alternatives from which instantiated physical architectures are developed. However, it is important to first understand, in detail, the allocated functions prior to exploring differing solutions that will execute those functions. *Task Description and Analysis* provides this level of detail for human-related tasks, which can help guide the identification of technologies with which the human must interact. This information should also be used to identify infeasible physical architecture alternatives (i.e., those that use technologies incompatible or difficult to use by humans).

6.4.3.2.3 Performance, Workload and Manning Level Estimation

With the potential of generating millions of possible combinations[1], it becomes necessary to screen out physical architectures based on a set of screening criteria. *Performance, Workload and Manning Level Estimation* can be used to help define criteria based on the human-specific performance requirements (e.g., accuracy, speed) and constraints (e.g., task loadings, personnel levels), thereby ensuring that only those physical architectures "optimized" for human interaction make it through the screening process. The performance evaluation data generated by this HSI method should also be used to inform the system physical models in evaluating alternatives.

6.4.3.2.4 Failure Analysis

This HSI method is particularly useful when one or more physical architectures under consideration are based on a precedent system. Since *Failure Analysis* is concerned with failure occurrence in current physical systems, the results should be used to throughout this MBSE step. When identifying component alternatives, technologies known to have a high probability of failure could be precluded from list. Likewise, physical configurations that lead to circumstances or events that cause errors can be eliminated as an infeasible alternative. Finally, screening criteria should take into account reliability data (i.e., probability of failure) generated by this HSI method.

6.4.4 Models-Based Systems Engineering Support Summary

While there are some similarities between the HSI methods described in this step and MBSE, they offer distinct enhancements that should be taken into account:

1. Use *Function Allocation* to:
    a. Supplement partitioning criteria to make allocation decisions;
    b. Conduct human-machine comparisons (e.g., Fitts list, decision space diagram) to determine proper allocations;
    c. Consider dynamic allocation, which increases architecture flexibility.
2. Use *Task Description and Analysis* to:
    a. Provide details for human-related tasks, which can help identify component alternatives;

[1] "For real systems there are usually millions of possible combinations, evaluation of which is likely intractable, making a preliminary screening necessary" (Beck, 2011).

    b. Identify which physical architectures are infeasible due human-machine incompatibilities.
3. Use *Performance, Workload and Manning Level Estimation* to:
    a. Define criteria to "screen out" physical architectures that do not meet human-specific performance requirements and constraints;
    b. Generate performance evaluation data that should inform physics-based models.
4. If one or more candidate physical architectures are based on a precedent system, use *Failure Analysis* data when:
    a. Identifying component technologies;
    b. Assessing infeasible alternatives;
    c. Generating preliminary screening criteria.

Generating, synthesizing, and down-selecting physical architectures could potentially be onerous depending on the number of alternatives, saying nothing of attempting to incorporate human factors. In such cases, these HSI methods provide valuable information to help inform the activities of this MBSE step and ensure optimal integration of the human within the system.

7. CONCLUSIONS

The incorporation of human factors within systems development continues to face several challenges within the systems engineering domain: 1) the human factors body of knowledge itself is currently fragmented; 2) there is a lack of formal integration of human factors methods within the SE process as defined by established standards; and 3) as MBSE continues to become more pervasive within the SE discipline, human factors are at risk of being left behind if MBSE support for human factors is not identified and implemented.

Human Systems Integration aims to address these problems. To accomplish this goal, the human factors domain must first be organized and consolidated in preparation for effective analysis and evaluation within the context of systems engineering. In order for human factors to be successfully incorporated within the systems development life cycle, HSI methods must be evaluated against the SE process to define what potential enhancements they offer and determine how they could be integrated to maximize the impact of those enhancements. Moreover, the same kind of evaluation must be conducted for MBSE to identify how models could be leveraged to represent the human element and, likewise, how HSI methods could be used to enhance the MBSE methodology.

The goal of this paper is to accomplish these tasks by providing a framework within which HSI methods could be assimilated into the SE process and identify how these methods could enhance the MBSE methodology. The results show that, while there are some HSI methods that do not introduce anything unique to the SE process or MBSE, there are some that, when integrated, enhance both. In general, most of these enhancements pertain to architectural activities of the SE process: *Functional Architecting* and *Physical Architecting*. Overall, whether defining stakeholder requirements or defining the physical architecture, the chief point is that when applying any method, there should be a focused effort to incorporate human factors within the development activities. Using the results from this paper as a starting point, the systems engineer can identify which HSI methods should be applied to a project, whether using an established SE process only or in conjunction with the MBSE methodology. In that regard, there are strong similarities between the HSI methods and the MBSE methodology, such as commonality of approaches and tools, which would potentially lead to more effective integration.

While this research may provide a stepping-stone to reaching a more complete integration of human factors within systems engineering and MBSE, there are several possible research opportunities going forward. Due to the limited scope of this paper, evaluation of the HSI methods mapped to the remaining Vee-model steps, *Implementation* through *Validation*, is needed to offer a more comprehensive approach to the integration of human factors within systems development. In addition, although many HSI methods were identified for this paper, there are still a significant number of sources within the human factors domain that remain untapped. Identifying the methods within these sources, organizing, and consolidating them will be key to enable a more effective integration within the systems engineering process. Finally, MBSE offers significant advantages through the use of modeling and simulation methods and related tools. The correlation between the human factors domain and MBSE requires further investigation to determine additional enhancements that HSI methods may provide to the MBSE methodology, with the intent of more accurately capturing the human element within systems.

8. REFERENCES

Booher, Harold R. 2003. *Handbook of Human Systems Integration*. John Wiley & Sons.

Boy, Guy A. 1998. “Cognitive Function Analysis for Human-Centered Automation of Safety-Critical Systems.”

Bruseberg, Anne. 2008. *The HFI Case Concept: Guidance on Specifying, Tracking and Documenting Human Factors Integration Requirements, Acceptance Criteria and Evidence*. Human Factors Integration Defence Technology Centre.

Human Factors Integration Defence Technology Centre. 2009. *Cost-Benefit Analysis for Human Factors Integration: A Practical Guide*.

Beck, David F. 2013. "Models-based Systems Engineering Process Methods" (in preparation), Sandia National Laboratories, Albuquerque, NM.

Burns, John, and Jerry Gordon. 2005. “Human Systems Integration” June 9, Orlando, FL.

Burns, John, Jerry Gordon, Matt Wilson, Milt Stretton, and Dan Bowdler. 2005. “A Framework for Applying HSI Tools in Systems Acquisition.” Paper No. 2281:11. Orlando, FL.

Chapanis, Alphonse. 1996. *Human Factors in Systems Engineering*. Wiley.

Cunio, Phillip, and M.L. Cummings. 2009. *A Framework for an HSI Downselection Tool*. Technical. Massachusetts Institute of Technology.

Defense Science Board. 2005. *Report of the Defense Science Board Task Force on Patriot System Performance*. OUSD for Acquisition, Technology,and Logistics,Washington,DC,20301-3140.

Directorate of Human Performance Integration | Human Performance Optimization Division. 2009. *Air Force Human Systems Integration Handbook: Planning and Execution of Human Systems Integration*. http://www.wpafb.af.mil/shared/media/document/AFD-090121-054.pdf.

Endsley, Mica. 2000. “Situation Models: An Avenue to the Modeling of Mental Models.” In *Proceedings of 14th Triennial Congress of the International Ergonomics Association and He 44th Annual Meeting of the Human Factors and Ergonomics Society*.

Estefan, Jeff A. 2008. *Survey of Model-Based Systems Engineering (MBSE) Methodologies*. International Council on Systems Engineering (INCOSE).

Flickinger, Don. 1957. “Man - The Essential Factor in Systems.” In *Proceedings of the National Symposium on Human Factors in Systems Engineering*. Philadelphia, PA: Human Factors Society of America and Institute of Radio Engineers and IRE Professional Group on Military Electronics.

Human Factors Society of America. 1957. “Man - The Essential Factor in Systems.” In *Proceedings of the National Symposium on Human Factors in Systems Engineering*. Philadelphia, PA: Human Factors Society of America and Institute of Radio Engineers and IRE Professional Group on Military Electronics.

Friedenthal, Sanford. 2012. *A Practical Guide to SysML: The Systems Modeling Language*. 2nd ed. Waltham, MA: Morgan Kaufmann.

Gabbar, Hossam. 2007. “Design of Virtual Plant Environment for Future Generation Green Production Systems.” *Systems Engineering* 10 (2) (February 14): 155–166. doi:10.1002/sys.20068.

Giachetti, Ronald, Veronica Marcelli, Jose Cifuentes, and Jose Rojas. 2012. “An Agent-Based Simulation Model of Human-Robot Team Performance in Military Environments.” *Systems Engineering* 16 (1) (February 15): 15–28. doi:10.1002/sys.21216.

Grady, Jeffrey O. 2010. *System Requirements Analysis*. Academic Press.

Guerra, Lisa. 2008. *Space Systems Engineering.* NASA, Exploration Systems Mission Directorate. http://spacese.spacegrant.org .

Handley, Holly. 2011. "Incorporating the NATO Human View in the DoDAF 2.0 Meta Model." *Systems Engineering* 15 (1) (June 1): 108–117. doi:10.1002/sys.20206.

Hardman, Nicholas, and John Colombi. 2011. "An Empirical Methodology for Human Integration in the SE Technical Processes." *Systems Engineering* 15 (2) (July 1): 172–190. doi:10.1002/sys.20201.

Harris, Steven, and Jennifer Narkevicious. 2013. "First Principles in the Analysis of Human-System Dynamics" presented at the INCOSE 2013 International Workshop, January 27, Jacksonville, FL.

Hause, Matthew, and Francis Thom. 2007. "HCI Aspects of SysML and Architectural Frameworks." In *Systems Engineering: Key to Intelligence Enterprises*. INCOSE.

Hebeler, Emily, Jennifer McKneely, and Sarah Rigsbee. 2012. "The Application of Human-Systems Integration: Designing the Next Generation of Military Global Positioning System Handheld Devices." *Johns Hopkins APL Technical Digest* 31 (1): 66–75.

The Institute of Electrical and Electronics Engineers, Inc. 2005. "Adoption of ISO/IEC 15288:2002 Systems Engineering—System Life Cycle Processes".

The Institute of Electrical and Electronics Engineers, Inc. 2005. "IEEE Standard for Application and Management of the Systems Engineering Process (IEEE Std 1220-2005)".

International Council on Systems Engineering (INCOSE). 2006. Version 02.42.00. *Object-Oriented System Engineering Method (OOSEM)*, April.

International Council on Systems Engineering. 2007. Version 3.2.2. *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*.

International Council on Systems Engineering. 2008. "Integrating the Human in Every System: Special Feature." *INSIGHT: Publication of the International Council on Systems Engineering* 11 (2) (April): 1–68.

Karwowski, Waldemar. 2006. *International Encyclopedia Of Ergonomics And Human Factors*. CRC Press.

Kossiakoff, Alexander, and William N Sweet. 2003. *Systems Engineering Principles and Practice*. Wiley Series in Systems Engineering and Management. Hoboken, N.J.: J. Wiley.

Lane, Jo Ann, and Tim Bohn. 2012. "Using SysML Modeling To Understand and Evolve Systems of Systems." *Systems Engineering* 16 (1) (November 28): 87–98. doi:10.1002/sys.21221.

Linsell, Mark, and Chris Vance. 2008. *Modelling Human Factors Using the Systems Modelling Language*. Human Factors Integration Defence Technology Centre.

Madni, Azad. 2009. "Integrating Humans with Software and Systems: Technical Challenges and a Research Agenda." *Wiley InterScience* (April 17). doi:10.1002/sys.20145.

Madni, Azad. 2010. "Towards a Generalizable Aiding-Training Continuum for Human Performance Enhancement." *System Engineering* 14 (2) (April 22): 129–140. doi:10.1002/sys.20166.

Madni, Azad, and Michael Sievers. 2012. "Systems Integration: Key Perspectives, Experiences, and Challenges." *Systems Engineering* (September 29): 1–15. doi:10.1002/sys.21249.

McGovern Narkevicius, Jennifer, John Winters, and Nicholas Hardman. 2008. "Talking the Talk: Cross-Discipline Terminology Challenges." *Incose Insight* 11 (2) (April): 25–27.

McKneely, J.A.B., D.F. Wallace, A.A. Perry, and J.J. Winters. 2006. “Human Systems Engineering Process and Methods.” In *International Encyclopedia of Ergonomics and Human Factors*. 2nd ed. Boca Raton, FL: Taylor & Francis Group, LLC.

Militello, Laura G., and Robert J. B. Hutton. “Applied Cognitive Task Analysis (ACTA): a Practitioner’s Toolkit for Under Standing Cognitive Task Demands.” *Ergonomics* 41 (11): 1618–1641.

Muralidhar, Ajoy. 2008. “How Human Systems Integration and Systems Engineering Can Work Together.” *Incose Insight* 11 (2) (April): 11–14.

National Aeronautics and Space Administration (NASA). 1995. *NASA Systems Engineering Handbook*, SP-610S, June.

Narkevicius, Jen. “Human Systems Integration: Railway Systems: Applications HSI Beyond DoD.”

Nemeth, Christopher P. 2004. *Human Factors Methods for Design: Making Systems Human-Centered*. Taylor & Francis.

Newman, Richard. 1999. “Issues in Defining Human Roles and Interactions in Systems.” *Systems Engineering* 2 (3) (June 15): 143–155. doi:10.1002/(SICI)1520.

Object Management Group (OMG). 2006. *OMG SysML Specification*, May.

Royce, Winston W. 1970. “Managing the Development of Large Software Systems.” In , 9. The Institute of Electrical and Electronics Engineers.

Rushby, John. 2001. “Modeling the Human in Human Factors.” In , 2187:86–91. Budapest, Hungary: Springer-Verlag.

Salvendy, Gavriel. 2012. *Handbook of Human Factors and Ergonomics*. John Wiley & Sons.

Sanders, Mark S., and Ernest J. McCormick. 1993. *Human Factors in Engineering and Design*. McGraw-Hill.

Sanders, Mary, and Elisabeth Fitzhugh. 2005. *Cognitive Systems Engineering Tool Survey - A Subtask in Support of Commander’s Decision Aids for Predictive Battle-Space Awareness (CDA4PBA)*. Human Effectiveness Directorate: Warfighter Interface Division: Cognitive Systems Branch. http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA452155.

Stammers, Rob B., Michael S. Carey, and Jane A. Astley. 1990. “Task Analysis.” In *Evaluation of Human Work*. Bristol, PA: Taylor & Francis, Ltd.

Stanton, Neville A., Paul M. Salmon, Guy H. Walker, Chris Baber and Daniel P. Jenkins. 2012. *Human Factors Methods: A Practical Guide for Engineering and Design*. Ashgate Publishing, Ltd.

Stanton, Neville Anthony, Alan Hedge, Karel Brookhuis, Eduardo Salas, and Hal W. Hendrick. 2004. *Handbook of Human Factors and Ergonomics Methods*. CRC Press.

Thryft, Ann. 2013. “Crowdsourcing App Helps Space Agency Improve Robots.” *Design News*, May.

United States of America Department of Defense. 1999. “Human Engineering Program Process and Procedures (MIL-HDBK-46855A)”.

United States of America Department of Defense. 2012. “Human Engineering Design Criteria Standard (MIL-STD-1472G)”.

U.S. Coast Guard Research and Development Center. 2009. *Survey of Human Systems Integration (HSI) Tools for USCG Acquisitions*.

Usability Partners. “ISO Standards: Standards in Usability and User-centred Design.” *Usability Partners*. http://www.usabilitypartners.se/about-usability/iso-standards.

Weimer, Jon. 1995. *Research Techniques in Human Engineering*. Prentice Hall.

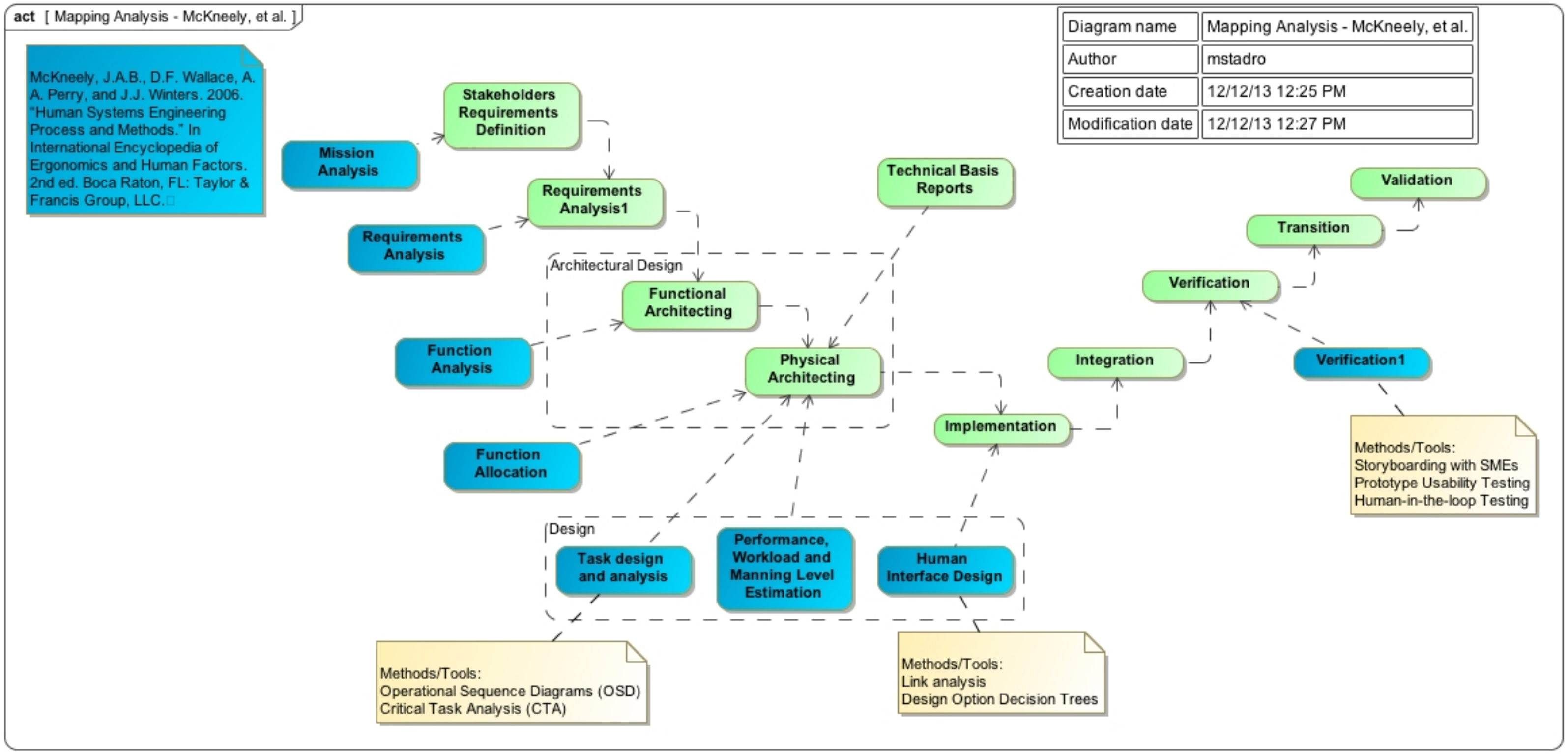
Weirich, Jim. 1999. *OOAD Design Problem: The Coffee Maker*.

Wickens, Christopher D., Anne S. Mavor, and James P. McGee. 1997. *Flight to the Future: Human Factors in Air Traffic Control*. National Academies Press.

Wilson, John R. 1990. "A Framework and a Context for Ergonomics Methodology." In *Evaluation of Human Work*. Bristol, PA: Taylor & Francis, Ltd.

Wilson, John R., and NIGEL CORLETT. 1990. *Evaluation of Human Work*. CRC Press.

APPENDIX A: MAPPING ANALYSIS DIAGRAMS



act [Mapping Analysis - McKneely, et al.]

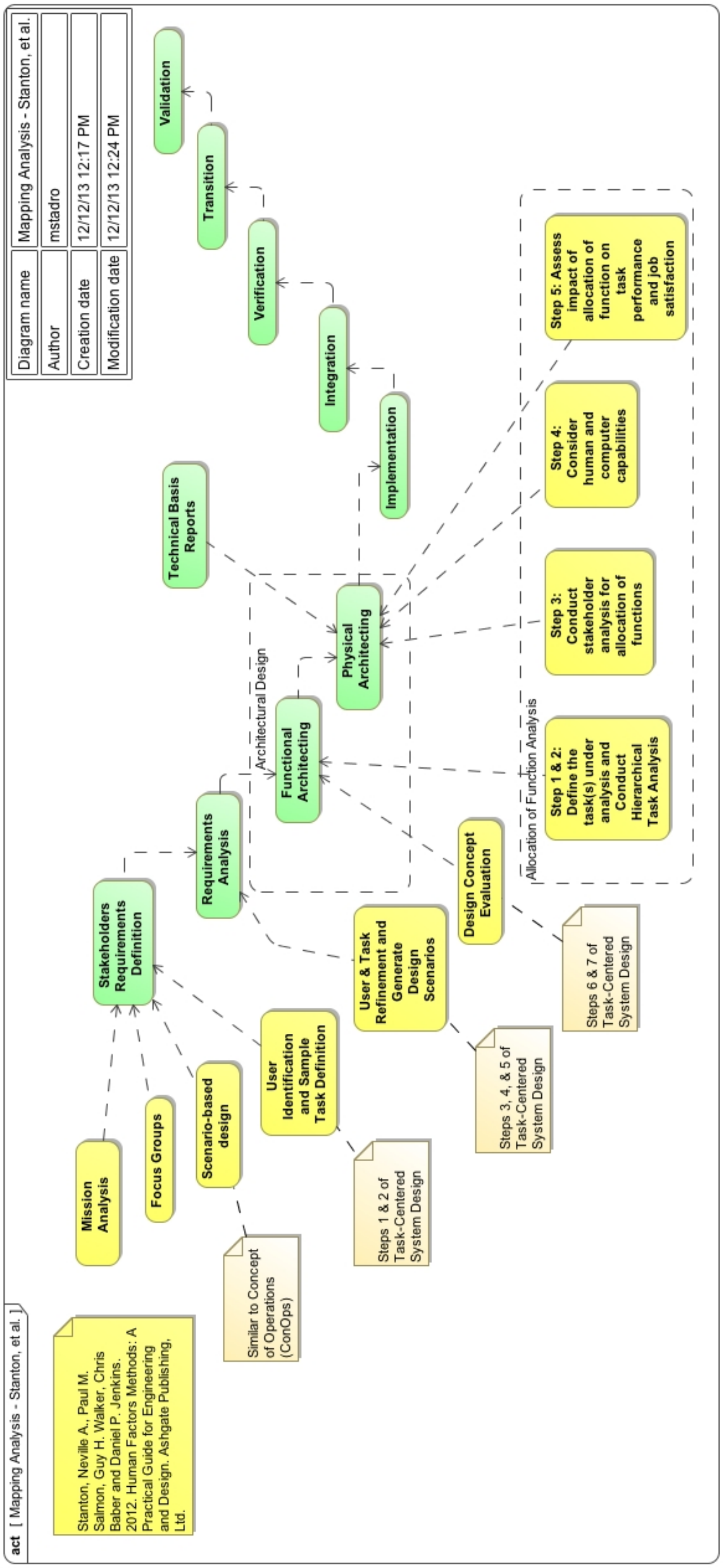
Diagram name	Mapping Analysis - McKneely, et al.
Author	mstadro
Creation date	12/12/13 12:25 PM
Modification date	12/12/13 12:27 PM

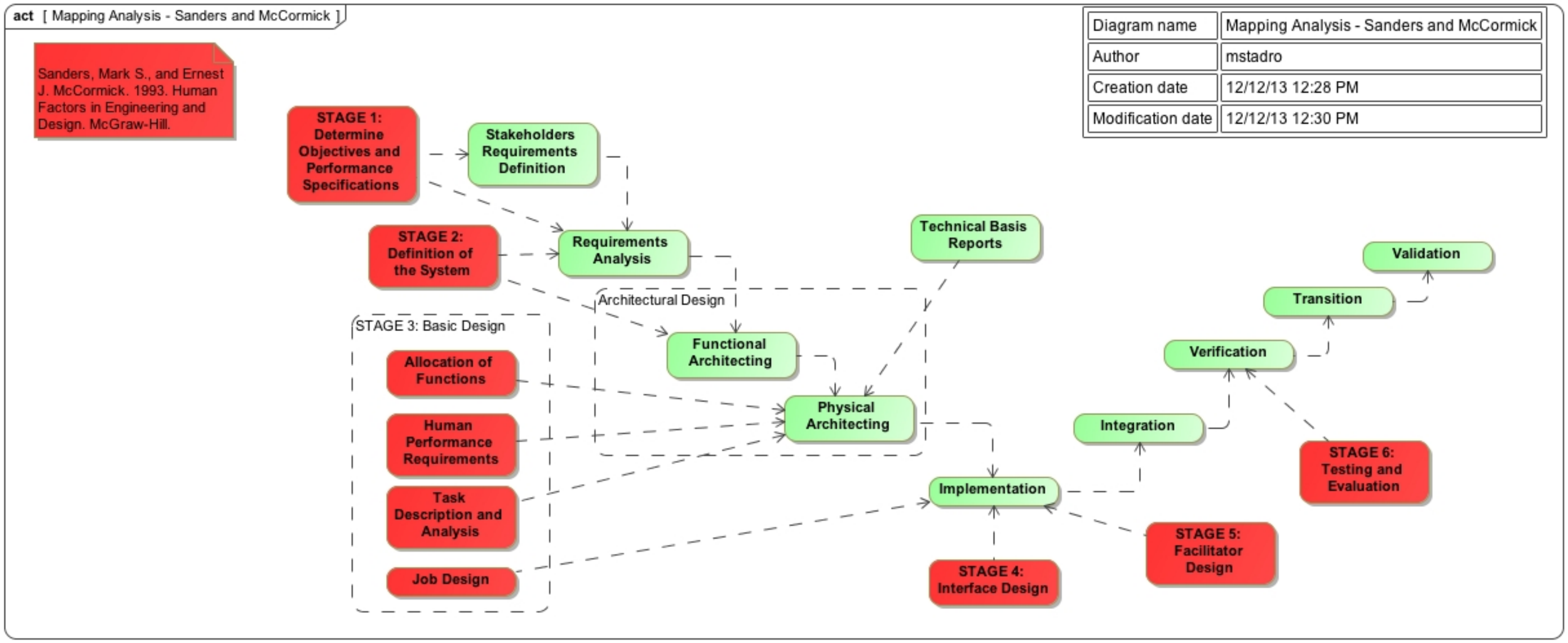
McKneely, J.A.B., D.F. Wallace, A. A. Perry, and J.J. Winters. 2006. "Human Systems Engineering Process and Methods." In International Encyclopedia of Ergonomics and Human Factors. 2nd ed. Boca Raton, FL: Taylor & Francis Group, LLC.

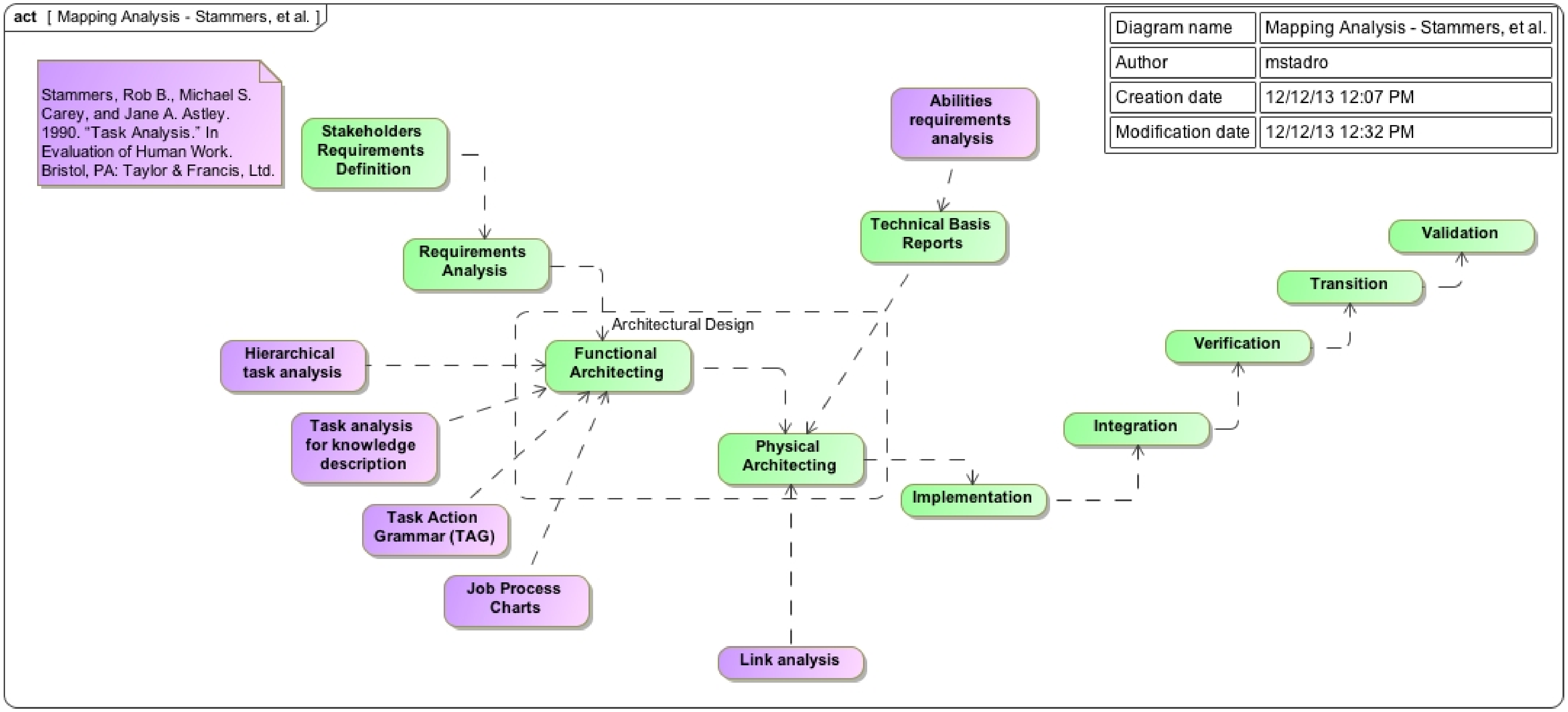
Methods/Tools:
Storyboarding with SMEs
Prototype Usability Testing
Human-in-the-loop Testing

Methods/Tools:
Operational Sequence Diagrams (OSD)
Critical Task Analysis (CTA)

Methods/Tools:
Link analysis
Design Option Decision Trees


act [Mapping Analysis - Stanton, et al.]
Diagram name
Mapping Analysis - Stanton, et al.
Author
mstadro
Creation date
12/12/13 12:17 PM
Modification date
12/12/13 12:24 PM
Stanton, Neville A., Paul M. Salmon, Guy H. Walker, Chris Baber and Daniel P. Jenkins. 2012. Human Factors Methods: A Practical Guide for Engineering and Design. Ashgate Publishing, Ltd.
Mission Analysis
Focus Groups
Scenario-based design
Similar to Concept of Operations (ConOps)
User Identification and Sample Task Definition
Steps 1 & 2 of Task-Centered System Design
User & Task Refinement and Generate Design Scenarios
Steps 3, 4, & 5 of Task-Centered System Design
Design Concept Evaluation
Steps 6 & 7 of Task-Centered System Design
Stakeholders Requirements Definition
Requirements Analysis
Architectural Design
Functional Architecting
Physical Architecting
Technical Basis Reports
Implementation
Integration
Verification
Transition
Validation
Allocation of Function Analysis
Step 1 & 2: Define the task(s) under analysis and Conduct Hierarchical Task Analysis
Step 3: Conduct stakeholder analysis for allocation of functions
Step 4: Consider human and computer capabilities
Step 5: Assess impact of allocation of function on task performance and job satisfaction

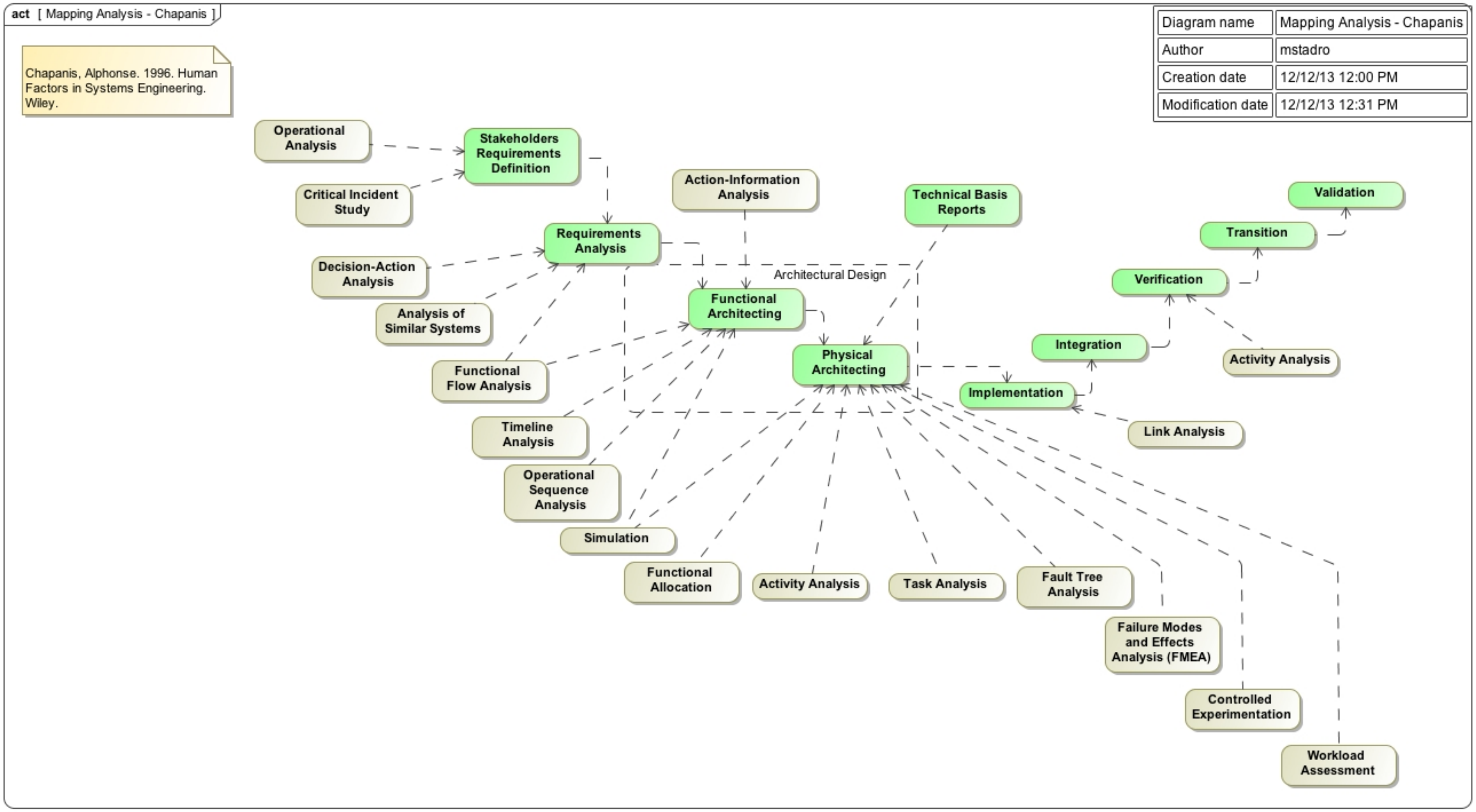

act [Mapping Analysis - Sanders and McCormick]
Sanders, Mark S., and Ernest J. McCormick. 1993. Human Factors in Engineering and Design. McGraw-Hill.
Diagram name
Mapping Analysis - Sanders and McCormick
Author
mstadro
Creation date
12/12/13 12:28 PM
Modification date
12/12/13 12:30 PM
STAGE 1: Determine Objectives and Performance Specifications
Stakeholders Requirements Definition
STAGE 2: Definition of the System
Requirements Analysis
Architectural Design
Functional Architecting
Physical Architecting
Technical Basis Reports
STAGE 3: Basic Design
Allocation of Functions
Human Performance Requirements
Task Description and Analysis
Job Design
Implementation
STAGE 4: Interface Design
STAGE 5: Facilitator Design
Integration
Verification
STAGE 6: Testing and Evaluation
Transition
Validation



act [Mapping Analysis - Stammers, et al.]
Stakeholders Requirements Definition
Requirements Analysis
Abilities requirements analysis
Technical Basis Reports
Architectural Design
Hierarchical task analysis
Functional Architecting
Task analysis for knowledge description
Task Action Grammar (TAG)
Job Process Charts
Physical Architecting
Link analysis
Implementation
Integration
Verification
Transition
Validation

Diagram name	Mapping Analysis - Stammers, et al.
Author	mstadro
Creation date	12/12/13 12:07 PM
Modification date	12/12/13 12:32 PM

Stammers, Rob B., Michael S. Carey, and Jane A. Astley. 1990. “Task Analysis.” In Evaluation of Human Work. Bristol, PA: Taylor & Francis, Ltd.


act [Mapping Analysis - Chapanis]
Diagram name
Mapping Analysis - Chapanis
Author
mstadro
Creation date
12/12/13 12:00 PM
Modification date
12/12/13 12:31 PM
Chapanis, Alphonse. 1996. Human Factors in Systems Engineering. Wiley.
Operational Analysis
Critical Incident Study
Stakeholders Requirements Definition
Requirements Analysis
Decision-Action Analysis
Analysis of Similar Systems
Functional Flow Analysis
Timeline Analysis
Operational Sequence Analysis
Simulation
Functional Allocation
Action-Information Analysis
Architectural Design
Functional Architecting
Physical Architecting
Technical Basis Reports
Activity Analysis
Task Analysis
Fault Tree Analysis
Failure Modes and Effects Analysis (FMEA)
Controlled Experimentation
Workload Assessment
Implementation
Link Analysis
Integration
Verification
Activity Analysis
Transition
Validation

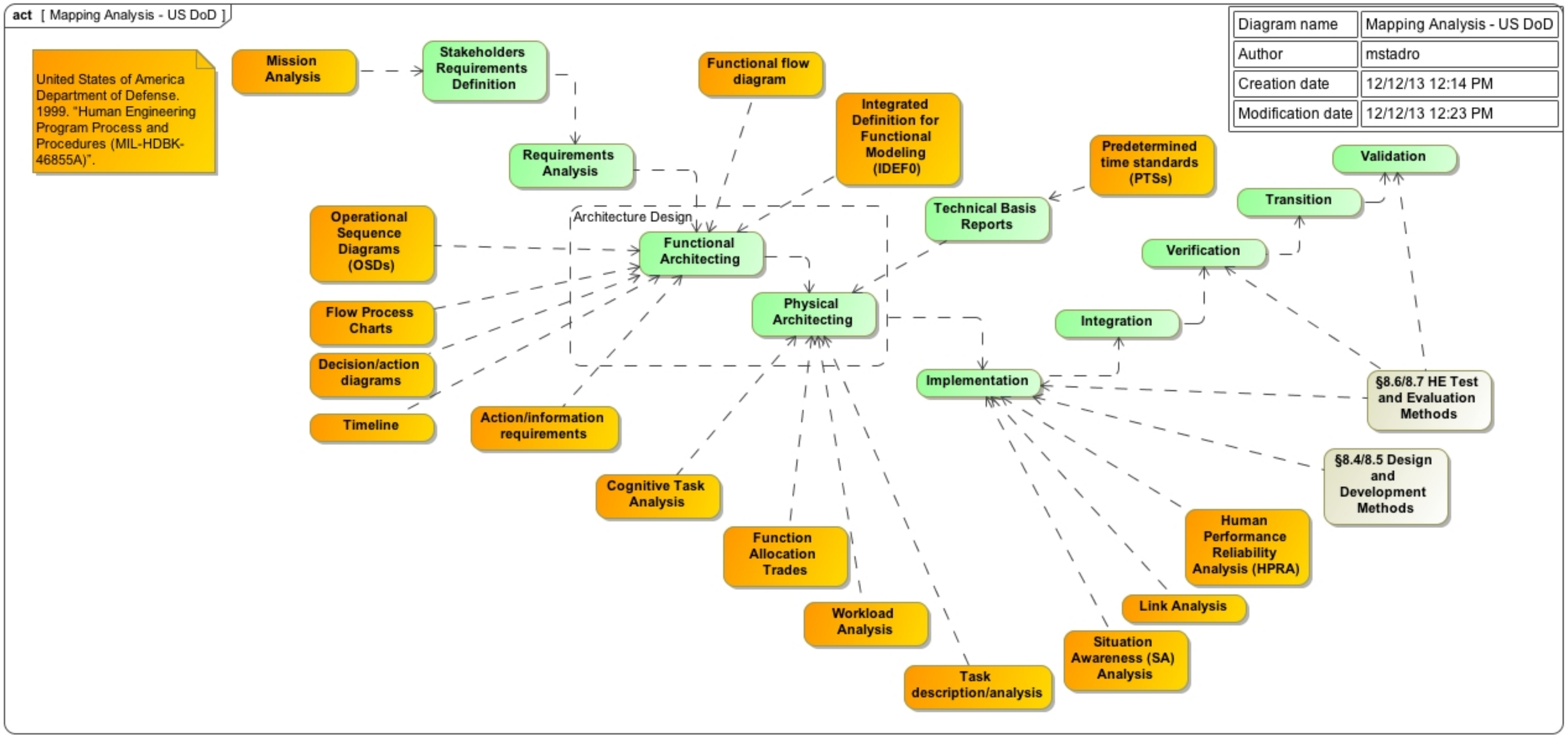

act [Mapping Analysis - US DoD]
Mission Analysis
Stakeholders Requirements Definition
Requirements Analysis
Functional flow diagram
Integrated Definition for Functional Modeling (IDEF0)
Predetermined time standards (PTSs)
Technical Basis Reports
Validation
Transition
Verification
Integration
Implementation
Architecture Design
Functional Architecting
Physical Architecting
Operational Sequence Diagrams (OSDs)
Flow Process Charts
Decision/action diagrams
Timeline
Action/information requirements
Cognitive Task Analysis
Function Allocation Trades
Workload Analysis
Task description/analysis
Situation Awareness (SA) Analysis
Link Analysis
Human Performance Reliability Analysis (HPRA)
§8.4/8.5 Design and Development Methods
§8.6/8.7 HE Test and Evaluation Methods

Diagram name	Mapping Analysis - US DoD
Author	mstadro
Creation date	12/12/13 12:14 PM
Modification date	12/12/13 12:23 PM

United States of America Department of Defense. 1999. "Human Engineering Program Process and Procedures (MIL-HDBK-46855A)".

APPENDIX B: MBSE STAKEHOLDER REQUIREMENTS DEFINITION

The following set of diagrams represents some of the activities from the first step in the MBSE method by using SysML to model an automobile system (Friedenthal, et al., Chapter 4, 2011).

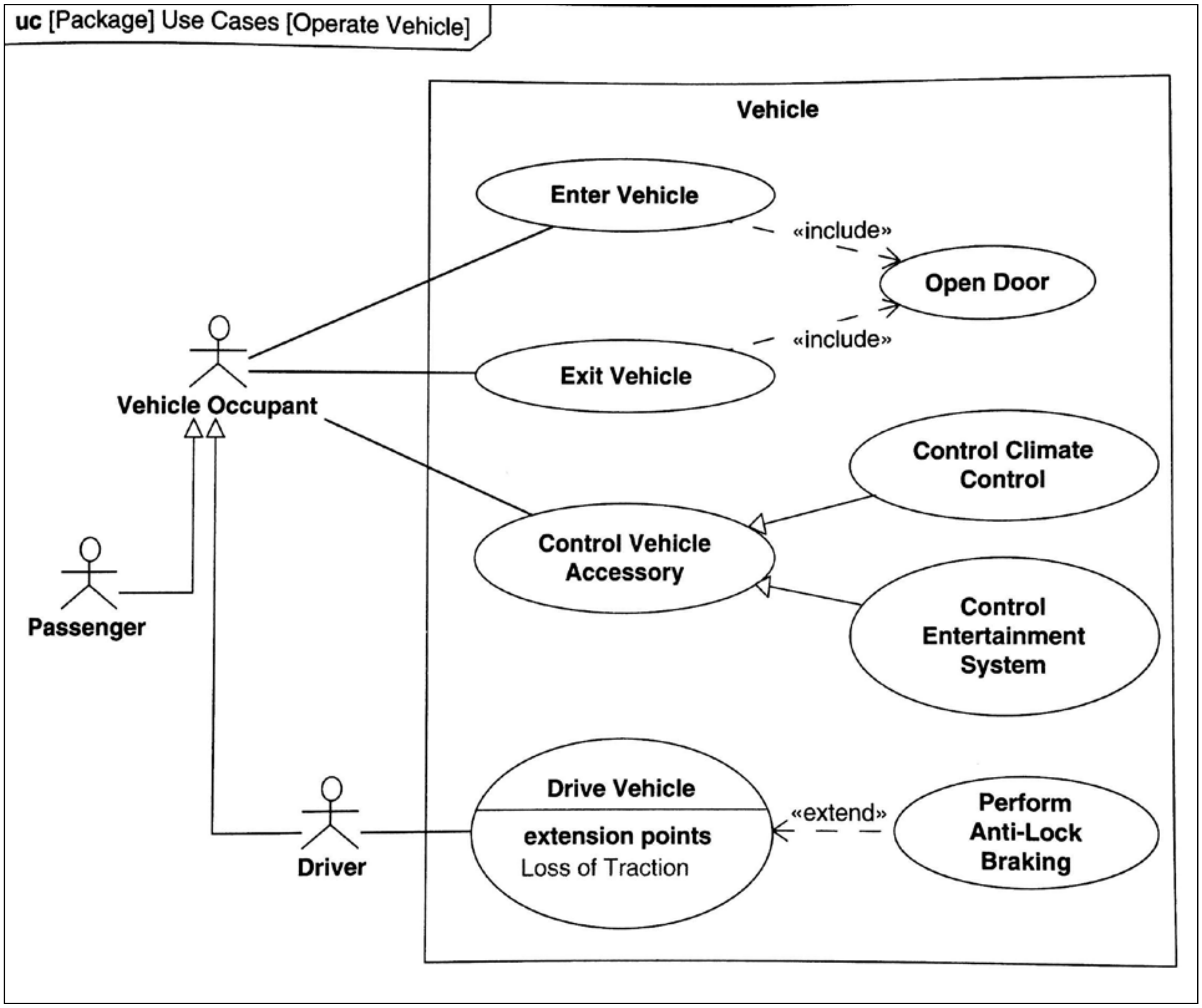


Figure 34: Use Case Diagram (uc) of a human operating a vehicle.

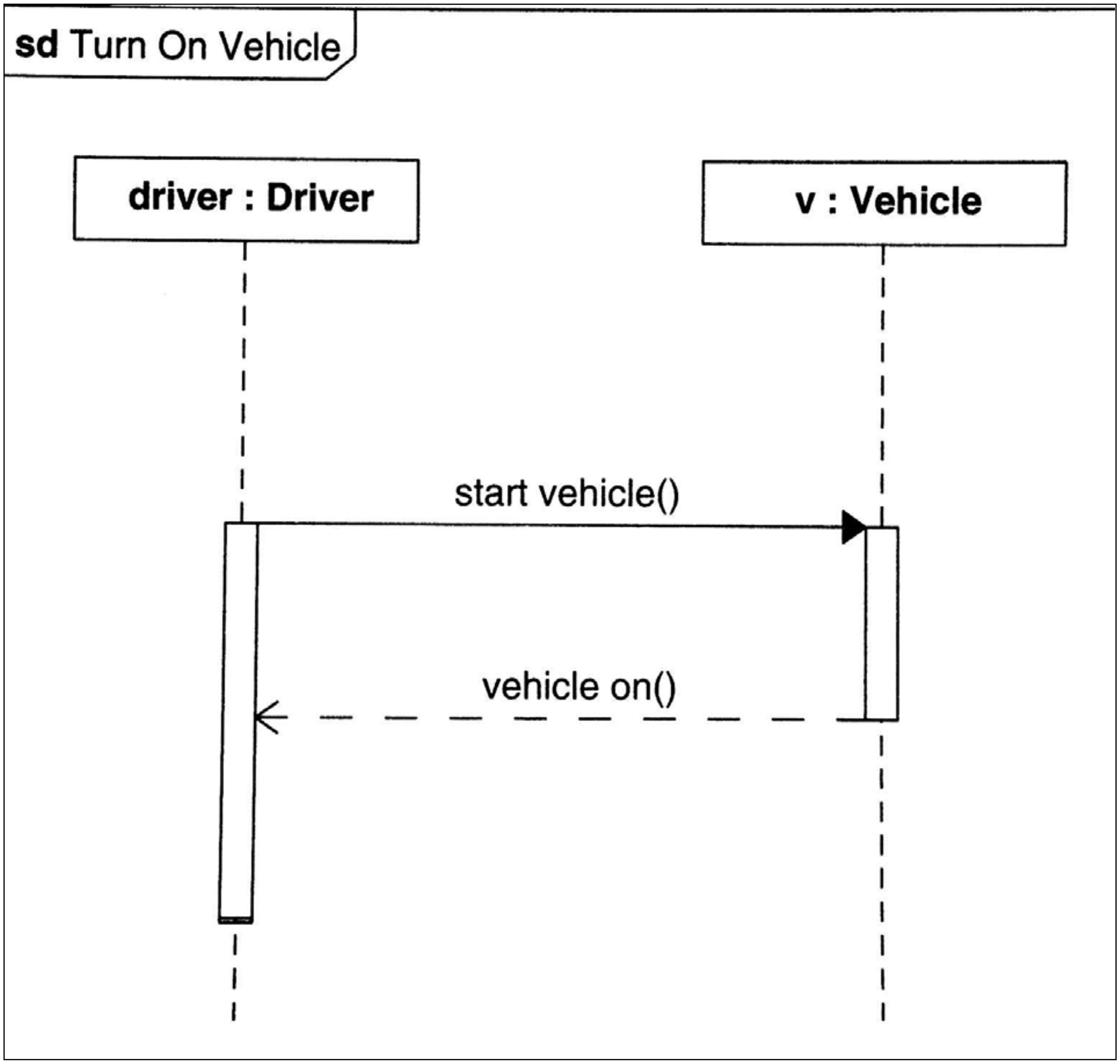


Figure 35: High-level human-system interaction captured using a Sequence Diagram (sd).

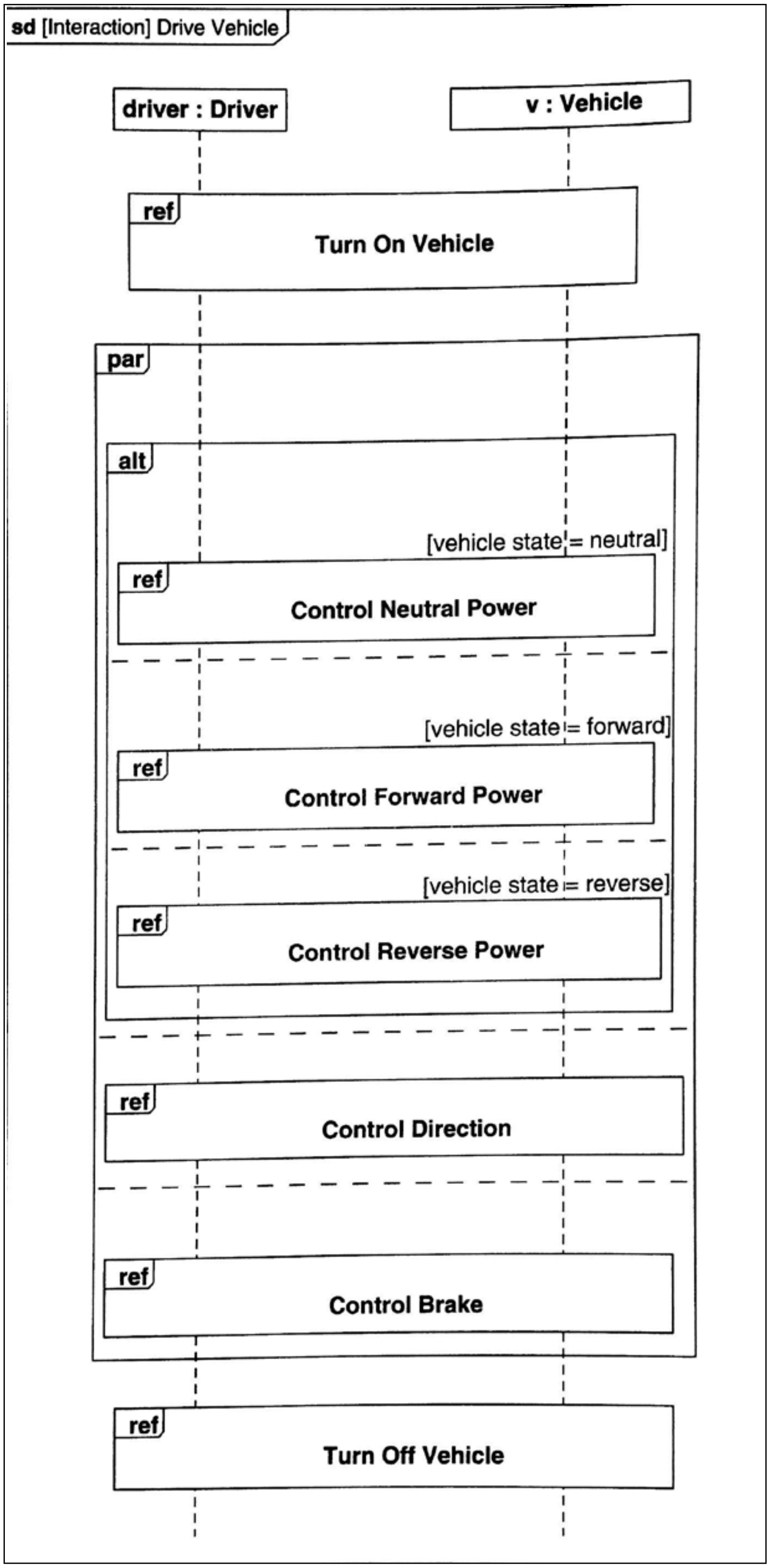


Figure 36: Detailed model of human-system interaction captured using a sequence diagram (sd).

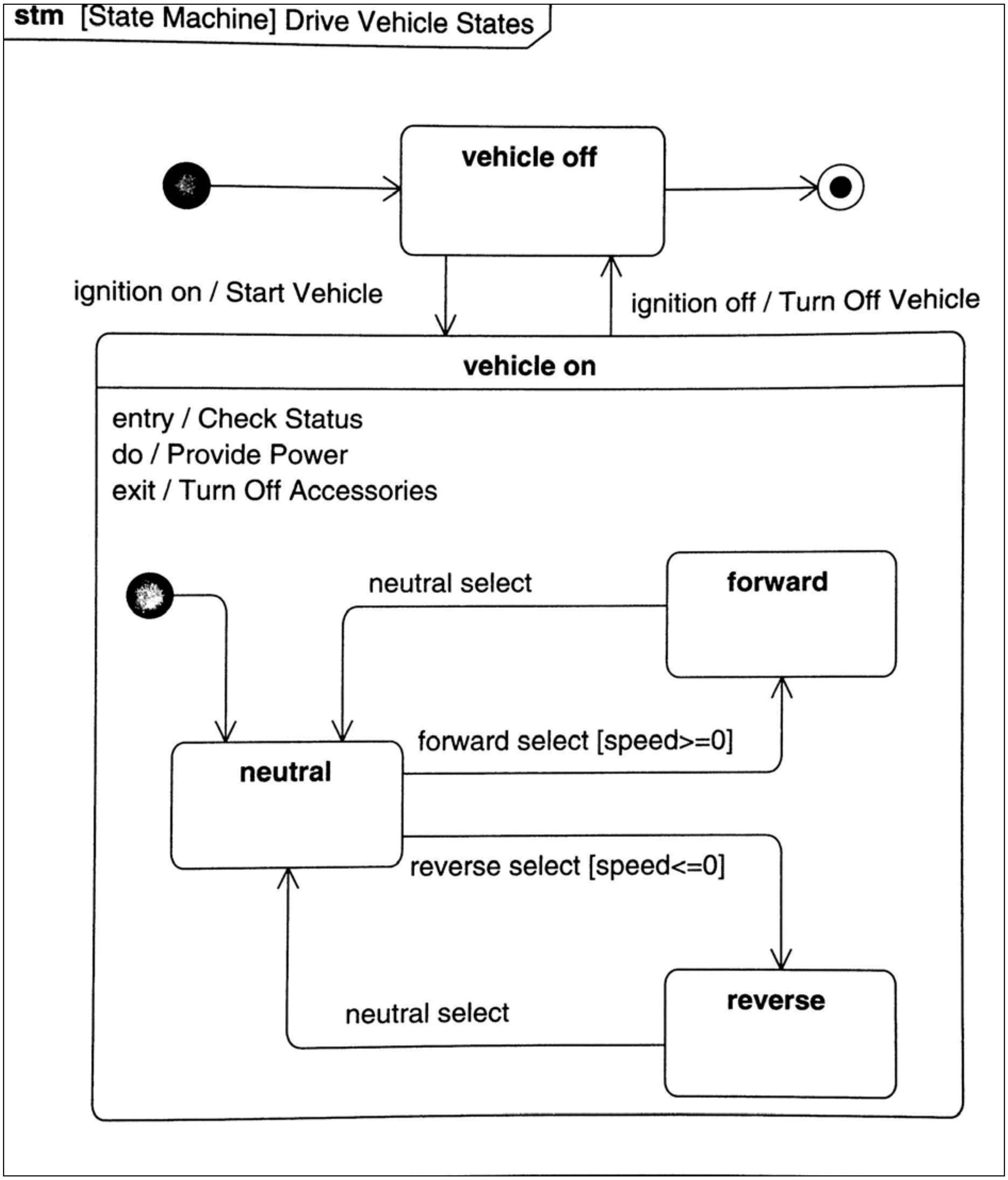

stm [State Machine] Drive Vehicle States
vehicle off
ignition on / Start Vehicle
ignition off / Turn Off Vehicle
vehicle on
entry / Check Status
do / Provide Power
exit / Turn Off Accessories
neutral select
forward
forward select [speed>=0]
neutral
reverse select [speed<=0]
neutral select
reverse

Figure 37: The *Drive Vehicle* use case elaborated using a State Machine (stm) diagram.

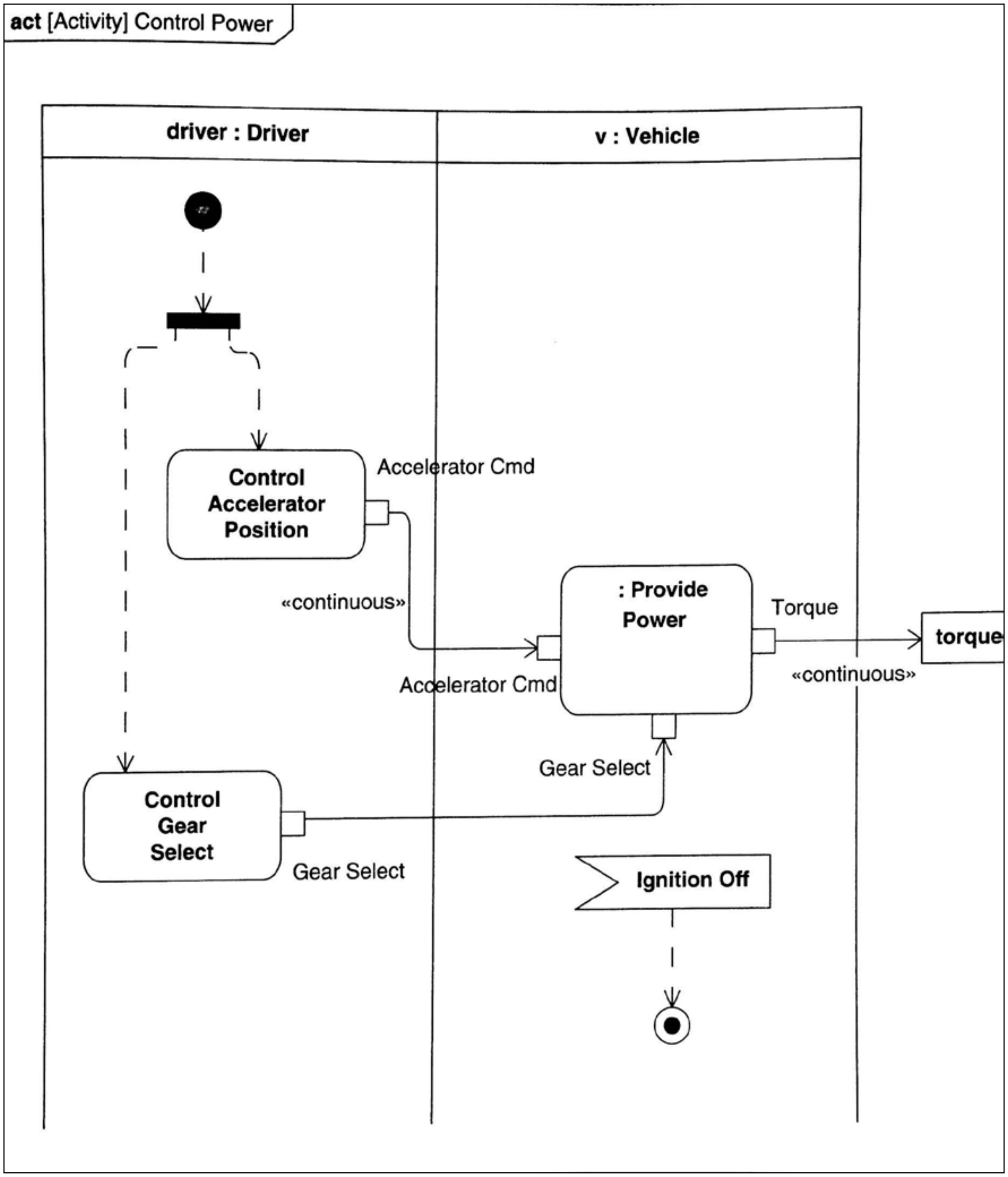


Figure 38: An Activity Diagram (act) with "swim lanes" distinguishes the actions performed by the human and those performed by the vehicle.

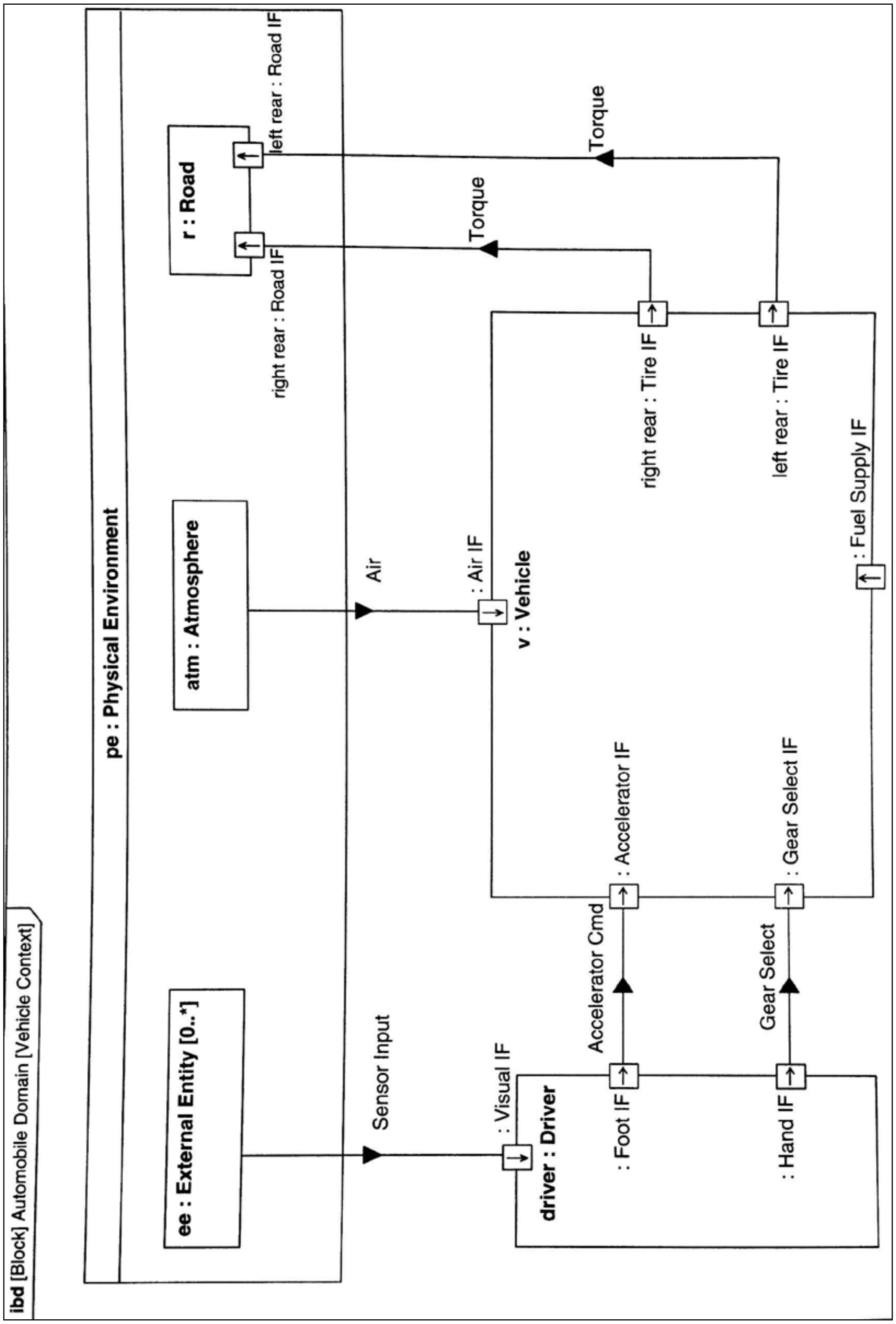


Figure 39: A contextual depiction of the vehicle's environment is captured using an Internal Block Definition Diagram (ibd).

APPENDIX C: MBSE FUNCTIONAL ARCHITECTURES

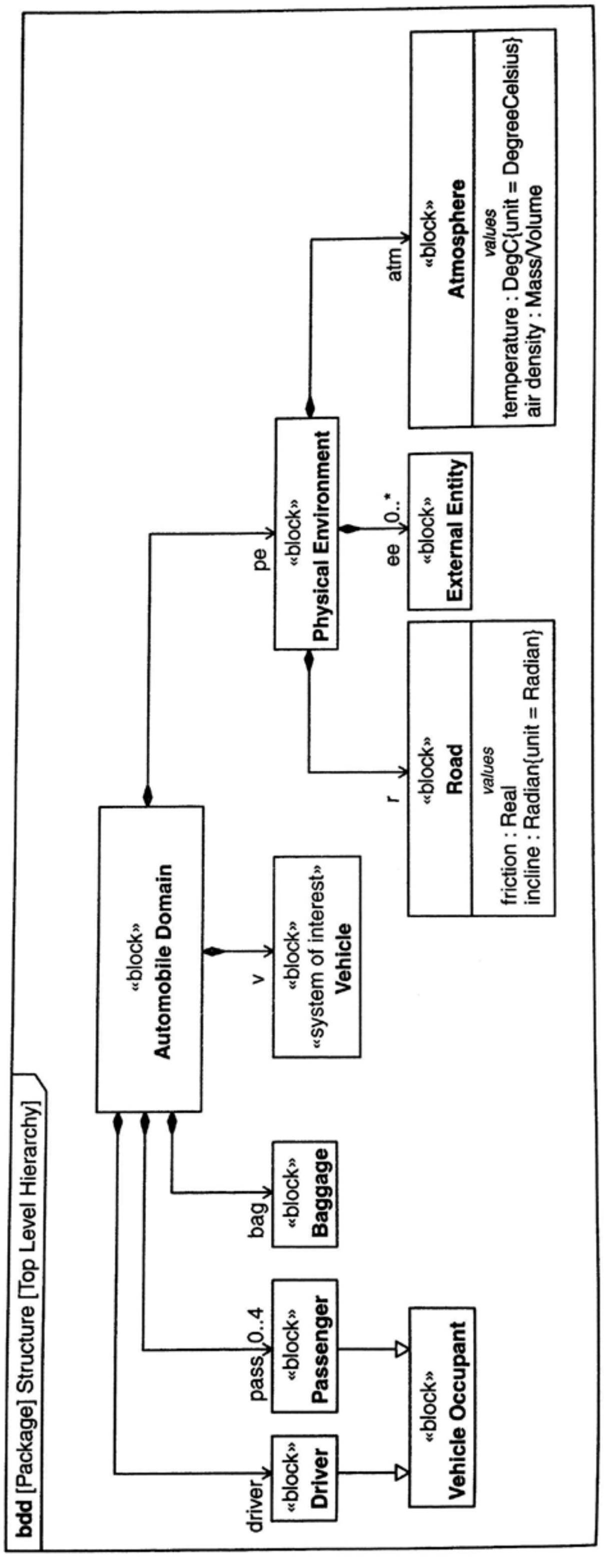


Figure 40: Block definition diagram (bdd) of the Automobile Domain (Friedenthal, et al., Chapter 4, 2011).

DISTRIBUTION

1	MS0933	Joselyne O. Gallegos	Org. 9500
1	MS0899	Technical Library	9536 (electronic copy)

Sandia National Laboratories