

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

LEVERAGING MODEL-BASED SYSTEMS ENGINEERING TO DEVELOP A REUSABLE VALIDATION, VERIFICATION, AND ACCREDITATION FRAMEWORK AND PROCESS FOR MODELING AND SIMULATION PRODUCTS

by

Gerardo Rodriguez Melo

September 2023

Thesis Advisor: Douglas L. Van Bossuyt
Co-Advisor: Mark M. Rhoades

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
LEVERAGING MODEL-BASED SYSTEMS ENGINEERING TO DEVELOP A REUSABLE VALIDATION, VERIFICATION, AND ACCREDITATION FRAMEWORK AND PROCESS FOR MODELING AND SIMULATION PRODUCTS

5. FUNDING NUMBERS

6. AUTHOR(S) Gerardo Rodriguez Melo

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

The Secretary of the Navy (SECNAV) Instruction 5200.46 requires the implementation of Validation, Verification and Accreditation (VV&A) to Modeling and Simulation (M&S) products and states that implementation of VV&A shall be commensurate with the level of risk, importance, and impact of M&S results. This thesis answers the primary research question: Can Model Based Systems Engineering (MBSE) be leveraged to develop a re-usable framework and process for supporting VV&A of M&S products? This thesis presents an MBSE based framework and methodology that can be applied to any M&S product to produce an MBSE model of the M&S product. The MagicGrid framework is used to guide the modeling processes which capture elements of the M&S product and other external entities that interact with the M&S model. The resulting MBSE model can be used for supporting VV&A responsibilities, activities, and products, such as the V&V plan. This thesis demonstrates the MBSE framework and methodology by applying it to the Cost Readiness Impact Model (CRIM), a Naval Air Systems Command (NAVAIR) M&S product. Application of this MBSE framework and process to CRIM resulted in an MBSE model of CRIM that defines CRIM's system architecture and associated Validation and Verification (V&V) elements. The CRIM MBSE model was evaluated to be effective in supporting VV&A elements. The author recommends this MBSE framework and methodology to anyone that must perform M&S VV&A.

14. SUBJECT TERMS	**15. NUMBER OF PAGES**
Model Based Systems Engineering, MBSE, Digital Engineering, DE, Digital Transformation, DT, Modeling and Simulation, M&S, Verification Validation and Accreditation, VV&A, Validation, Verification, Accreditation, Cost Readiness Impact Model, CRIM, MagicGrid, Presentation Framework, Architecture Framework, Validation	337
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

LEVERAGING MODEL-BASED SYSTEMS ENGINEERING TO DEVELOP A REUSABLE VALIDATION, VERIFICATION, AND ACCREDITATION FRAMEWORK AND PROCESS FOR MODELING AND SIMULATION PRODUCTS

Gerardo Rodriguez Melo
Civilian, Department of the Navy
BS, University of Texas, El Paso, 2013
MS, University of Texas, El Paso, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2023

Approved by: Douglas L. Van Bossuyt
Advisor

Mark M. Rhoades
Co-Advisor

Oleg A. Yakimenko
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The Secretary of the Navy (SECNAV) Instruction 5200.46 requires the implementation of Validation, Verification and Accreditation (VV&A) to Modeling and Simulation (M&S) products and states that implementation of VV&A shall be commensurate with the level of risk, importance, and impact of M&S results. This thesis answers the primary research question: Can Model Based Systems Engineering (MBSE) be leveraged to develop a re-usable framework and process for supporting VV&A of M&S products? This thesis presents an MBSE based framework and methodology that can be applied to any M&S product to produce an MBSE model of the M&S product. The MagicGrid framework is used to guide the modeling processes which capture elements of the M&S product and other external entities that interact with the M&S model. The resulting MBSE model can be used for supporting VV&A responsibilities, activities, and products, such as the V&V plan. This thesis demonstrates the MBSE framework and methodology by applying it to the Cost Readiness Impact Model (CRIM), a Naval Air Systems Command (NAVAIR) M&S product. Application of this MBSE framework and process to CRIM resulted in an MBSE model of CRIM that defines CRIM's system architecture and associated Validation and Verification (V&V) elements. The CRIM MBSE model was evaluated to be effective in supporting VV&A elements. The author recommends this MBSE framework and methodology to anyone that must perform M&S VV&A.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1

A. INSPIRATION 1

B. BACKGROUND ON THE DOD'S AND DON'S DIGITAL ENGINEERING TRANSFORMATION 2

1. On Digital Transformation 5

2. On Systems Engineering Transformation 7

C. BACKGROUND ON THE COST READINESS IMPACT MODEL 7

D. OBJECTIVES AND CONTRIBUTION 9

1. Primary Objective 9

2. Secondary Objective 9

3. Contribution 10

E. THE RESEARCH QUESTIONS 11

F. SCOPE, LIMITATIONS, AND ASSUMPTIONS 11

G. ORGANIZATION OF STUDY 12

II. LITERATURE REVIEW AND THEORETICAL FRAMEWORK 15

A. SYSTEMS THEORY 15

1. Systems Hierarchy, Decomposition, Black-Box And White-Box Abstraction, And General Logical Abstraction 16

2. The Machine Age's Mechanistic Understanding Of World: Reductionism And Determinism 24

3. The Systems Age: Expansionism, Teleology, Synthetic Thinking, Systems Approach 26

B. SYSTEMS ARCHITECTURE 31

1. Systems Architecture Overview 31

2. Systems Architecture Definitions 34

3. Modular vs. Integral Architecture. 35

4. The Physical Architecture 37

5. The Functional Architecture 40

6. The Allocated Architecture 50

7. Architecture Presentation Frameworks 53

C. SYSTEMS ENGINEERING 62

1. Systems Engineering Definitions 63

2. Technical Processes and Technical Management Processes 64

3. Systems Engineering Approaches 65

4. System Engineering Activities 68
5. Systems Engineering Technical Reviews, System Baselines and System Specifications..... 75
6. System Design Considerations..... 81
7. Measures of Effectiveness, Measures of Performance and Key Performance Parameters, and Key System Attributes..... 83
D. SYSTEMS ENGINEERING VALIDATION, VERIFICATION AND ACCREDITATION 88
E. MODEL-BASED SYSTEMS ENGINEERING 91
1. The Three Pillars of MBSE –by Delligatti..... 94
2. The MBSE Approach, "Same Same, but Different" 95
3. Background on the Systems Modeling Language..... 98
F. THE MAGICGRID FRAMEWORK..... 109
1. Why MagicGrid? 110
G. MODELING AND SIMULATION..... 113
1. Taxonomy of Model Types..... 114
2. Quantitative Modeling Uncertainty and Errors 118
3. Uncertainty Classification Example..... 120
H. MODELING AND SIMULATION VALIDATION, VERIFICATION AND ACCREDITATION 135
1. M&S VV&A Background and Definitions..... 135
2. Sargent's Simplified Model Development Process and V&V Process Relationships 139
3. M&S V&V Framed through Sargent's Real-World and Simulation-World Paradigms..... 148
4. Validation Techniques..... 153
5. Accreditation and Credibility..... 165
I. COST READINESS IMPACT MODEL PAST V&V RESEARCH 167
J. MULTI-ECHELON, MULTI-INDENTURE INVENTORY SYSTEMS..... 168
2. The System Approach..... 169
3. The Item Approach..... 171
4. The conceptual "Physics" behind the Multi-Echelon Inventory Problem..... 172
5. The Conceptual "Physics" behind the Multi-Echelon, Multi-Indenture Inventory Problem..... 177

III. METHODOLOGY AND PRESENTATION OF DATA AND RESULTS . 179
1. What the M&S VV&A Model Is, and What It Is Not!..... 179

2. Goals of this Methodology 181
3. When Is the MBSE Model Good Enough, Or What Is The Required Level Of Detail? 182
4. "Discrete" Descriptive Modeling 184
B. METHODOLOGY 186
1. Methodology Summary 186
2. Reverse Engineering the Cost Readiness Impact Model (CRIM) 189
3. Forward Engineering Future M&S Products and Their VV&A Elements 191
4. Modeling of Stakeholder Needs 193
5. Modeling CRIM Use Cases 197
6. Modeling CRIM's System Context 202
7. Modeling CRIM's Functional Analysis 206
8. Modeling CRIM's Measurements of Effectiveness 211
9. Modeling CRIM's Logical Subsystems Communication 213
10. Modeling (W4) Measures of effectiveness 217
11. Modeling System Structure 218
12. Modeling CRIM's System Behavior 222
13. Modeling CRIM's System Requirements 226
14. Modeling CRIM's System Parameters 229
15. Modeling Subsystem Behavior 232
16. Modeling Component Behavior 236
17. Modeling Subsystem Structure 241
18. Modeling Subsystem Parameters 245
19. Modeling Component Structure 247
20. Modeling Component Parameters 251
21. Other "Fit for Purpose Views" 257

IV. RESULTS ANALYSIS AND INTERPRETATION 261
A. RESULTS EVALUATION: CRITERIA 1 TO 4 262
1. Criteria 1. Evaluation of the CRIM MBSE Model System Boundaries 264
2. SCORE: Excellent 264
3. Criteria 2. Evaluation of the CRIM MBSE Model Data Flows/Interfaces 264
4. SCORE: Good 264
5. Criteria 3. Evaluation of the CRIM MBSE Model System Behavior 264
6. SCORE: Good 264

7. Criteria 3. Evaluation of the CRIM MBSE Model V&V entities 265
8. SCORE: Good 265
B. RESULTS EVALUATION: CRITERIA 5 TO 11 265
1. Criteria 5. Evaluation of the CRIM MBSE Model to support the V&V Plan's Problem Statement Definition 268
2. SCORE: Excellent 268
3. Criteria 6. Evaluation of the CRIM MBSE Model to support the V&V Plan's M&S Requirements and Acceptability Criteria 268
4. SCORE: Excellent 268
5. Criteria 7. Evaluation of the CRIM MBSE Model to support the V&V Plan's M&S Assumptions, Capabilities, Limitations and Risks/Impacts 268
6. SCORE: Poor or Needs Work 268
7. Criteria 8. Evaluation of the CRIM MBSE Model to support the V&V Plan's V&V Methodology 269
8. SCORE: Good 269
9. Criteria 9. Evaluation of the CRIM MBSE Model to support the V&V Plan's V&V Issues 269
10. SCORE: Poor or Needs Work 269
11. Criteria 10. Evaluation of the CRIM MBSE Model to support the V&V Plan's Key (V&V) Participants definition. 270
12. SCORE: Excellent 270
13. Criteria 11. Evaluation of the CRIM MBSE Model to support the V&V Plan's V&V Resources 270
14. SCORE: Poor or Needs Work 270
C. RESULTS EVALUATION: CRITERIA 12 TO 15 270
1. Criteria 12. Evaluation of the CRIM MBSE Model to Support Defining the M&S System (problem) entity. 273
2. SCORE: Good 273
3. Criteria 13. Evaluation of the CRIM MBSE Model to Support Defining the M&S Conceptual Model 273
4. SCORE: Good 273
5. Criteria 14. Evaluation of the CRIM MBSE Model to Support Defining M&S Data Validity 274
6. SCORE: Good 274
7. Criteria 15. Evaluation of the CRIM MBSE Model to Support Defining the M&S Computerized Model 274
8. SCORE: Good 274
D. RESULTS EVALUATION: CRITERIA 16 TO 18 274

1. Criteria 16. Evaluation of the CRIM MBSE Model to support M&S Conceptual Model Validation. 275
2. SCORE: Excellent 275
3. Criteria 17. Evaluation of the CRIM MBSE Model to support M&S Operational Validation. 276
4. SCORE: Good 276
5. Criteria 18. Evaluation of the CRIM MBSE Model to support M&S Computerized Model Verification 276
6. SCORE: Poor or Needs Work 276

V. CONCLUSIONS 279

A. RESULTS AND RECOMMENDATIONS 279
1. Results Summary 279
2. Recommendations 282
3. Method recommendation 282
4. Evaluation Recommendation 282

B. CONCLUSIONS 282
1. Future Research 285

APPENDIX 287

LIST OF REFERENCES 297

INITIAL DISTRIBUTION LIST 305

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Black-box and white-box examples 17

Figure 2. Taxonomy example. 18

Figure 3. SysML generalization relationship. The syntax in SysML should read a "Book" is a type of "Library Item." Source: No Magic (n.d.). 20

Figure 4. Systems and Services Viewpoint from the Marine Corp Systems Command (MCSC) Ontology and Conceptual Data Model (CDM). 23

Figure 5. Conceptual Data Model Format. This format shows the graphical language syntax used in the CDM. Source: Vaneman et al. (2022, 37). 24

Figure 6. Students in Two- and Four-year Institutions in the United States as a Fraction of 18 to 21Year-Olds: 1890 to 1970. Source: Golding and Katz (1999, 41). 29

Figure 7. Systems architecting "Process Flow Details." Source: College of Research and Engineering (2019b, 17). 32

Figure 8. Systems architecting in relation to the Systems Engineering "Vee" Adapted from Buede and Miller (2016, 10) 33

Figure 9. Architecture Definitions. Source: Rhoades (2022, 15). 35

Figure 10. Examples of form: entities of form and their relationships. Drawings and diagrams convey information about how form exists in a system – some information is conveyed abstractly, other information more concretely. Adapted from Crawley et al. (2016) 39

Figure 11. Functional hierarchy and decomposition of functions. Source: Faisandier and Roedler (2023). 44

Figure 12. Functional analysis and decomposition, functional flow block diagram (FFBD) traceability and indenture. Source: Systems Management College (2001, 49) 45

Figure 13. Behavior architecture model view. 49

Figure 14. System architecture development. Source: Buede and Miller (2016, 27). 51

Figure 15. Functional allocation example. Adapted from Aleksandravicience and Morkevicius (2018). 52

Figure 16. Functional to physical allocation matrix. Source: Systems Management College (2001, 59)..... 53

Figure 17. Ontology (conceptual map) of an ontology specification. Source: Cambridge Semantics (n.d.)..... 55

Figure 18. Example of an ontology view. Source: Xue et al. (2022)..... 56

Figure 19. The Zachman Framework for Enterprise Architecture The Enterprise Ontology. Source: Zachman (n.d.)..... 58

Figure 20. DoDAF Viewpoints. Source: Chief Information Officer (n.d.b)..... 60

Figure 21. Presentation techniques for architecture visualizations. Source: Chief Information Officer (n.d.c). 61

Figure 22. Mapping of architecture views with respect to the systems engineering process. Source: College of Research and Engineering (2019a, 20)..... 62

Figure 23. Systems Engineering Technical and Technical Management Processes. Source: Department of Defense (2022, 5)..... 65

Figure 24. The Traditional, Top-Down Systems Engineering (TTDSE) view focusing on system definition maturation phases and their respective V&V processes. Source: Buede and Miller (2016, 20). 66

Figure 25. The Waterfall model. Source: Buede and Miller (2016, 22). 67

Figure 26. The Spiral model. Source: Buede and Miller (2016, 21)..... 68

Figure 27. The Systems Engineering "Vee" focusing on Systems engineering processes. Source: Buede and Miller (2016, 10). 70

Figure 28. Development process for the interface architecture. Source: Buede (2009, 321)..... 71

Figure 29. The systems engineering recursive process. Source: Systems Management College (2001, 6)..... 72

Figure 30. Recursive systems engineering design pattern and overall pattern for system levels of development and specification outputs. Source: Systems Management College (2001, 75). 73

Figure 31. The SYSMOD zigzag pattern. Source: Weilkiens (2015, 51)..... 75

Figure 32. Systems engineering technical review (SETR) events mapped to the systems engineering "Vee." Source: Systems Management College (2001, 65)..... 76

Figure 33. The Systems Engineering "Vee" referencing the Technical and Technical Management processes, and the Systems Engineering Technical Review (SETR) events. Source: College of Research and Engineering (2019e, 72)..... 77

Figure 34. Systems architecting design process mapping to technical reviews and system baselines. Source: College of Research and Engineering (2019c, 43)..... 78

Figure 35. System technical baselines specification tree. This image is from the author's personal collection and is not publicly accessible. Source: College of Research and Engineering (2019d, 60)..... 79

Figure 36. Specification types. Source: Systems Management College (2001, 75)..... 80

Figure 37. The specification process leading to the technical data package. Source: Systems Management College (2001, 76)..... 81

Figure 38. Systems engineering "Vee" validation, verification, and acceptance (VV&A) concept mappings. Source Buede and Miller (2016, 344)..... 89

Figure 39. Verification Action from the Systems Engineering Body of Knowledge. Source: Snoderly and Faisandier (2023)..... 91

Figure 40. Ansys MBSE domain perspective. Source: McBee (2022)..... 93

Figure 41. SysML diagram taxonomy..... 99

Figure 42. Summary of SysML diagrams. Source: Crawley, Cameron, and Selva (2016, 46)..... 101

Figure 43. SysML and OPM logical relationships showing hierarchies of composition and abstraction, and logical representation of classes (elements of definition) and instances (elements of usage). Source: Crawley, Cameron, and Selva (2016, 48)..... 102

Figure 44. Model element types allowed in each of the nine SysML diagram kinds. Source: Delligatti (2013, 20)..... 103

Figure 45. Block definition diagram notation summary. Source: Holt and Perry (2019, 141)..... 105

Figure 46. Internal block diagram notation summary. Source: Holt and Perry (2019, 151)..... 106

Figure 47. Requirement diagram notation summary. Source: Holt and Perry (2019, 181)..... 107

Figure 48. Activity diagram notation summary. Source: Holt and Perry (2019, 213) 108

Figure 49. Use case diagram notation summary. Source: Holt and Perry (2019, 222) 109

Figure 50. The MagicGrid Framework v.1. Source: Aleksandraviciene and Morkevicius (2018, 11)..... 112

Figure 51. The MagicGrid Framework v.2. Source: Aleksandraviciene and Morkevicius (2021, 12)..... 113

Figure 52. Uncertainty taxonomy and means of uncertainty mitigation. Source: Gansch and Adee (2020, 1348)..... 118

Figure 53. Modeling uncertainty example. Source: Gansch and Adee (2020, 1347). 120

Figure 54. Conceptual representation of a model or modeler's knowledge about the system..... 123

Figure 55. Model or modeler's epistemic uncertainty management concept mapping, adapted from the "Rumsfeld matrix" and the Johari window..... 125

Figure 56. The Johari Window. Source: Wikipedia (2023b). 126

Figure 57. Taxonomy of uncertainties by their classification. Source: Oberkampf and Roy (2010, 54). 127

Figure 58. Deterministic vs. stochastic evolution of a system's state. Source: Cantot and Luzeaux (2013, 60)..... 129

Figure 59. Stochastic model parts and sources of uncertainty. Source: Rhoades (2020, 4). 131

Figure 60. Basic steps in an uncertainty analysis. Source: Oberkampf and Roy (2010, 70). 133

Figure 61. Three types of error sources in solution verification—from the perspective of a Partial Differential Equation simulation. Source: Beisbart and Saam (2019, 77). 134

Figure 62. Sargent's simplified version of the model development process. Source: Sargent (2013, 14). 140

Figure 63. Phases of M&S and the role of V&V. Source: Schlesinger (1979, 1).... 141

Figure 64. Verification and validation activities and products. Source: Oberkampf and Roy (2010, 30). 142

Figure 65. Simulation V&V process model. Source: Beisbart and Saam (2019, 595). 143

Figure 66. The M&S development and M&S VV&A life cycle. Source: Fonseca i Casas (2023, 5). 144

Figure 67. V&V relationships between the "Real-world" and "simulation-world" paradigms. 149

Figure 68. Graphical model examples. 158

Figure 69. Validation test classification. Source: Fonseca i Casas (2023, 10). 160

Figure 70. Proposed documents for each product of the modeling cycle. Source: Fonseca i Casas (2023, 6) 161

Figure 71. The When, What How Why (W3H) testing table, Part 1. Source: Fonseca i Casas (2023, 11) 162

Figure 72. The When, What How Why (W3H) testing table, Part 2. Source: Fonseca i Casas (2023, 12) 163

Figure 73. Testing simulation models using frequentist statistics. Source: Beisbart and Saam (2019, 483). 164

Figure 74. Framework for conducting validation experiments. Source: Beisbart and Saam (2019, 609). 165

Figure 75. The accreditation conceptual mapping. Source: Pace (2004, 165). 166

Figure 76. The V&V levels of independence spectrum. Source: Oberkampf and Roy (2010, 49). 167

Figure 77. Optimal availability-cost curve example. Source: Sherbrooke (2004, 4). 171

Figure 78. "Physics" of the problem. Adapted from Sherbrooke (2004, 6-7). 173

Figure 79. Multi-Echelon pipeline model to enable "end-to-end simulation." Source: Institute for Defense Analyses (2023b). 174

Figure 80. O-I-D sustainment structure. Source: Institute for Defense Analyses (2023a, 12). 175

Figure 81. Modeling parts as a single echelon. Source: Institute for Defense Analyses (2023a, 23). 175

Figure 82. Modeling parts as a single echelon (continued). Source: Institute for Defense Analyses (2023a, 25). 176

Figure 83. CRIM MBSE model domain of interest. Adapted from Pace (2004)..... 180
Figure 84. M&S system boundary partitions. 183
Figure 85. Freefall equation described as a graphical model showing inputs, functions or operations, and a single output. 185
Figure 86. Parts of an algebraic expression. Source: BYJU'S (n.d.). 186
Figure 87. Modeling process summary 187
Figure 88. MagicGrid QuickStart template view 188
Figure 89. Overall modeling results. 189
Figure 90. Proposed SETR events that roughly or loosely coordinate with the MagicGrid v.1 Framework. Adapted from Aleksandraviciene and Morkevicius (2018)..... 192
Figure 91. Rough approximation of the system's baselines overlap with the MagicGrid v.1 framework. 193
Figure 92. MagicGrid v.1 QuickStart template guide for Stakeholder Needs. Image from CSM v.2021.x. 194
Figure 93. Stakeholder Needs Modeling Results 196
Figure 94. MagicGrid v.1 QuickStart template guide for Use Cases. Image from CSM v.2021.x. 197
Figure 95. General Use Cases modeling results. 200
Figure 96. High-Level Stakeholder Use Cases modeling results 201
Figure 97. "Low-Level Stakeholder" use cases modeling results with pasted reference artifacts (Innoslate CRIM Use Case Diagram). Zoom in to see small details. 201
Figure 98. Low-Level Stakeholder Use Cases modeling results. SysML model elements only. 202
Figure 99. MagicGrid v.1 QuickStart template guide for the System Context. Image from CSM v.2021.x. 203
Figure 100. "System Context" block definition diagram showing the entities that compose the system context in its entirety. 204
Figure 101. System Context internal block diagram showing the constituent parts of the system context and their connective structure. 205

Figure 102. MagicGrid v.1 QuickStart template guide for modeling the Functional Analysis. Image from CSM v.2021.x. 206

Figure 103. Forecast Metrics functional analysis, modeled in an activity diagram. Zoom in to see finer details. 208

Figure 104. "Forecast Metrics" functional analysis, modeled in an activity diagram. This graphic shows only the SysML model elements. 209

Figure 105. Optimize Return on Investment 210

Figure 106. MagicGrid v.1 QuickStart template guide for Measures of Effectiveness. 211

Figure 107. CRIM system level MoE and MoP statements 213

Figure 108. MagicGrid v.1 QuickStart template guide for modeling the Logical Subsystems Communication 214

Figure 109. CRIM logical system structure 216

Figure 110. Conceptual model Validation (W4 Measures of Effectiveness). 218

Figure 111. MagicGrid v.1 QuickStart template guidance for the System Structure 219

Figure 112. CRIM System Structure, block definition diagram with a pasted artifact. Zoom in to see small details. 220

Figure 113. CRIM's internal block diagram with pasted artifacts. Zoom in to see small details. There is an enlarged version of this image in the appendix. 221

Figure 114. CRIM (proposed) System Behavior with reference artifacts. Zoom in for finer details. 224

Figure 115. CRIM System behavior, SysML model elements only. 225

Figure 116. MagicGrid v.1 QuickStart template guide for System Requirements 227

Figure 117. CRIM System Requirements with a reference Innoslate artifact. Zoom in to see finer details. 228

Figure 118. MagicGrid v.1 QuickStart template guide for modeling System Parameters 230

Figure 119. CRIM System Parameters 231

Figure 120. MagicGrid v.1 QuickStart template guide for modeling Subsystem Behavior 233

Figure 121. CRIM Subsystem Behavior, “Input Data” activity diagram. Zoom in to see small details. 234
Figure 122. CRIM Subsystem Behavior, “Analyze Data” activity diagram. 235
Figure 123. CRIM Subsystem Behavior, “Analyze Data” activity diagram showing only SysML model elements. 235
Figure 124. SIMLOX Prep & Analysis and Simulation activity diagram with pasted reference artifacts 238
Figure 125. SIMLOX Prep & Analysis and Simulation activity diagram, SysML model elements only. 238
Figure 126. Perform Driver Tree Analysis activity diagram with pasted reference artifacts. 240
Figure 127. Perform Driver Tree Analysis activity diagram, SysML elements only. 240
Figure 128. MagicGrid v.1 QuickStart Template guide for modeling Subsystem Structure 241
Figure 129. Driver Tree Subsystem Structure with pasted artifacts 243
Figure 130. Driver Tree Subsystem Structure, SysML model elements only 243
Figure 131. SIMLOX Prep. & Analysis Suite Subsystem Structure with pasted reference artifacts. 244
Figure 132. SIMLOX Prep. & Analysis Suite Subsystem Structure, SysML model elements only. 244
Figure 133. Subsystem Parameters bdd diagram. 246
Figure 134. Honeybee Component Structure 249
Figure 135. Driver Tree Mathematical Relationships model internal block diagram, only showing a pasted graphical artifact for this component. 250
Figure 136. SIMLOX Multi-Echelon, Multi-Indenture model component parameters 252
Figure 137. Model generated Operational Flight Profile performance parameter. 253
Figure 138. Model generated Multi-Indenture validity performance parameter 254
Figure 139. Model generated metric distributions performance parameter 255
Figure 140. Multi-echelon, multi-indenture model performance parameters. 256

Figure 141. Driver Tree component parameters 257
Figure 142. Navy External Data Sources view. 258
Figure 143. NAVAIR Funding Accounts view. 258
Figure 144. Taxonomy of Data classes. 259
Figure 145. CRIM System Actors view 259
Figure 146. CRIM stakeholders view 260
Figure 147. Qualitative Evaluation Rubric for sub-criteria/measures 1–4 of 18 total. 263
Figure 148. Qualitative Evaluation Rubric for sub-criteria/measures 5–11 of 18 total. 267
Figure 149. Sargent's simplified version of the model development process. Source: Sargent (2013, 14). 271
Figure 150. Qualitative Evaluation Rubric for sub-criteria/measures 12–15 of 18 total. 272
Figure 151. Qualitative Evaluation Rubric for sub-criteria/measures 16–18 of 18 total. 275
Figure 152. Overall Rubric score distribution for the CRIM MBSE model. The overall score distribution gives a rough idea of how effective the MBSE model may be for supporting VV&A efforts or deriving VV&A products. 281
Figure 153. Thesis research question conceptual model 284
Figure 154. Systems and Services Viewpoint from the Marine Corp Systems Command (MCSC) Ontology and Conceptual Data Model (CDM). Zoom in to see smaller details. This image is a slightly larger version of Figure 4. Source: Vaneman et al. (2022,41). 287
Figure 155. "Physics" of the Problem: Diagram synthesized and translated from Sherbrooke's (2004) textual description. Zoom in to see smaller details. This image is a slightly larger version of Figure 78. 288
Figure 156. CRIM's internal block diagram with pasted artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 113. ... 289
Figure 157. SIMLOX Prep & Analysis and Simulation activity diagram with pasted reference artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 124. 290

Figure 158. Perform Driver Tree Analysis activity diagram with pasted reference artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 126. 291

Figure 159. Driver Tree Subsystem Structure with pasted artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 129. 292

Figure 160. SIMLOX Prep. & Analysis Suite Subsystem Structure with pasted reference artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 131. 293

Figure 161. Driver Tree Mathematical Relationships. Zoom in to see smaller details. This image is a slightly larger, cropped version of Figure 135. 294

Figure 162. Navy External Data Sources view. Zoom in to see smaller details. This image is a slightly larger version of Figure 142. 295

LIST OF TABLES

Table 1. DOD and DON Digital Engineering Transformation Strategy Goals 6

Table 2. Table mapping systems engineering technical reviews to their respective system baseline. 78

Table 3. Examples of Design Considerations from Blanchard and Fabrycky and the Defense Acquisition University—considerations are not matched for similarity. Adapted from Blanchard and Fabrycky (2011) and DAU (n.d.a). 82

Table 4. Similarities between Systems Engineering and M&S VV&A concepts..... 88

Table 5. Taxonomy of Model types (example)..... 116

Table 6. Modeling uncertainty example: discussion. Adapted from Gansch and Adee (2020)..... 121

Table 7. Stakeholder Needs Information and Data Collected..... 195

Table 8. Use Case Information or Data Collected 198

Table 9. System Context Information or Data Collected..... 203

Table 10. Functional Analysis Information or Data Collected 207

Table 11. Measurements of Effectiveness Information or Data Collected 212

Table 12. Logical Subsystems Communication Information or Data Collected 214

Table 13. System Structure Information or Data Collected..... 219

Table 14. System Behavior Information or Data Collected..... 222

Table 15. System Requirements Information or Data Collected 227

Table 16. System Parameters Information or Data Collected..... 230

Table 17. Subsystem Behavior Information or Data Collected 233

Table 18. Component Behavior Information or Data Collected..... 236

Table 19. Subsystem Structure Information or Data Collected 242

Table 20. Component Structure Information or Data Collected..... 247

Table 21. Component Parameters Information or Data Collected..... 251

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
AIAA	American Institute of Aeronautics and Astronautics
ASME	American Society of Mechanical Engineers
CDM	Conceptual Data Model
CRIM	Cost Readiness Impact Model
DE	Digital Engineering
DET	Digital Engineering Transformation
DMSO	Defense Modeling and Simulation Office.
DOD	Department of Defense
DON	Department of the Navy
DT	Digital Transformation
IEEE	International Electrical Engineering
M&S	Modeling and Simulation
MBSE	Model-Based Systems Engineering
MCSC	Marine Corp Systems Command
MIL-STD	Military Standard
ML	Machine Learning
NAVAIR	Naval Air Systems Command
NMSO	Navy Modeling and Simulation Office
OR	Operations Research
PMA	Program Management Office-Air
SE	Systems Engineering
SECNAV	Secretary of the Navy
SECNAVINST	Secretary of the Navy Instruction
SET	Systems Engineering Transformation
SME	Subject Matter Expert
SoI	System of Interest

V&V	Validation and Verification
VV&A	Validation, Verification, and Accreditation

EXECUTIVE SUMMARY

This thesis researches a reusable model-based systems engineering (MBSE) framework and process (methodology) that can be used to support Validation, Verification and Accreditation (VV&A) of Modeling and Simulation (M&S) products. VV&A of Navy M&S products or systems is required by the *Department of the Navy Modeling, Simulation, Verification, Validation, and Accreditation Management SECNAVINST 5200.46* (Secretary of the Navy 2019). The *Department of Defense Standard Practice, Documentation of Verification, Validation and Accreditation (VV&A) for Models and Simulation MIL-STD-3022 W/CHANGE 1* (Department of Defense 2012, 1) primarily establishes templates for the four core VV&A products, i.e., the "Accreditation Plan," "V&V Plan," "V&V Report," and "Accreditation Report," but does not provide a methodology that can be used to develop a V&V plan or VV&A framework.

This research provides a framework and methodology that can be used by M&S stakeholders to support their VV&A processes and products. This research uses the MagicGrid framework by Dassault Systèmes as the main architectural framework that provides structure and guidance for the modeling methodology. The tool used for modeling was Cameo Systems Modeler (CSM) (version 2021.x). The modeling language used was the Systems Modeling Language (SysML). The author slightly modified the MagicGrid framework so that validation and verification (V&V) elements are captured as part of the system "Parameters" column. The M&S product is treated like a system and modeled according to the MagicGrid framework structure and guidelines. The resulting product is an MBSE model that captures the system architecture of the M&S product at various levels of decomposition, including system V&V elements that are useful for VV&A.

To demonstrate this thesis's concept, the Naval Air Systems Command (NAVAIR) Cost Readiness Impact Model (CRIM) was used as the M&S subject for this framework and methodology. Documents and information were collected to support modeling of CRIM. The amount of information and details collected limited the amount and richness of details that could be modeled. The overall result was an MBSE model of CRIM. Diagrams populate the MagicGrid template index; these diagrams are made up of the

underlying MBSE model elements. The diagrams capture and convey the primary source of information about CRIM's system architecture, and the V&V elements associated with CRIM.

To evaluate the resulting CRIM MBSE model, the author designed a qualitative rubric that uses various criteria to evaluate the model's effectiveness to support VV&A of the M&S product. By association, the framework and methodology that produced the CRIM MBSE model is also evaluated, but indirectly. In this specific case, the CRIM MBSE model was evaluated against the rubric's criteria to determine if the model can be used to effectively support future CRIM VV&A efforts or to derive CRIM VV&A products, such as the V&V plan. The author's evaluation resulted in six counts of criteria rated as "Excellent," eight counts of criteria rated as "Good," and four counts of criteria rated as "Poor or Needs Work." The author recognized that the CRIM MBSE model's effectiveness was hindered due to the limited information and knowledge available to support modeling; this was not a fault of the methodology or the modeling tool.

The framework and methodology presented in this thesis can be followed and applied by M&S stakeholders who must perform VV&A on their M&S products. The results of this framework and methodology result in an MBSE model of the M&S product or system, which can be used to support M&S VV&A efforts and derivation of VV&A products.

References

Department of Defense. 2012. *Department of Defense Standard Practice Documentation of Verification, Validation, and Accreditation (VV&A) for Models and Simulations*. MIL-STD-3022 w/CHANGE1. Washington, DC: Department of Defense. http://everyspec.com/MIL-STD/MIL-STD-3000-9999/MIL-STD-3022_CHG-1_46836/

Secretary of the Navy. 2019. *Department of the Navy Modeling, Simulation, Verification, Validation, and Accreditation Management*. SECNAVINST 5200.46. Washington, DC: Department of the Navy.

ACKNOWLEDGMENTS

This thesis research is dedicated to my wife, Beth, and my family. My loved ones were with me on this journey; they supported me along the way, joining their dedication, understanding, and their sacrifice to mine.

I give special thanks to my advisors, Dr. Van Bossuyt and Professor Rhoades. Dr. Van Bossuyt was a great beacon of light; he guided me through this research process and provided the guidance that allowed me to refine my research rationale and methods. His guidance helped me ground my work and he was always encouraging, which was very much needed in this busy master's program. Professor Rhoades was instrumental in the development of my modeling skills. I am still on my way to become an expert modeler, there is no doubt I have much practice ahead of me, but his lectures and the knowledge he imparted on me paved the road for my ability to have the confidence and skills to give the modeling methods I employed as part of my research a shot.

I would also like to recognize Jason Thomas, the new Director of Digital Transformation at NAVAIR, a NPS doctoral student, and my amicable project lead. I joined his team around April of 2022. He provided me with the mentoring and ideas that helped me derive this thesis's research questions. My motivation for producing quality work is based on my desire to give something of value back to Jason and NAVAIR. I hope that my work will be useful to our project in the future, and that others may find my work useful as well.

Last, but certainly not least! I thank God for His gifts and blessings. There were many times I struggled and was dismayed, but in the face and kindness of everyone that helped me through this thesis and master's program I saw His love and felt His helping hand.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This chapter presents the inspiration and background for this research. This chapter also presents the research's objectives, contributions, research primary and secondary questions, research limitations, and the organization of this overall document.

A. INSPIRATION

This thesis is inspired by the Navy's need to develop and employ Modeling and Simulation (M&S) across its purview and use M&S to answer difficult questions. Specifically, the Navy is interested in questions that emerge from the defense acquisition processes which present a mixture of engineering, logistics, and management questions. M&S is being driven by the Digital Engineering (DE) Transformation Strategies of Department of Defense (DOD) and the Department of the Navy (DON). Models play a key role in the successful realization of the DOD's and DON's DE strategy objectives.

This thesis specifically researches a Model-Based Systems Engineering (MBSE) centric framework and process (methodology) that supports Validation, Verification, and Accreditation (VV&A) of M&S products. VV&A of the Navy's M&S products is required by the *Department of the Navy Modeling, Simulation, Verification, Validation, and Accreditation Management SECNAVINST 5200.46* (Secretary of the Navy 2019). Navy M&S products can also adhere to or be required to satisfy the Department of Defense Standard Practice *Documentation of Verification, Validation, and Accreditation (VV&A) for Models and Simulations MIL-STD-3022 w/CHANGE 1* (Department of Defense 2012). VV&A of M&S products supports *governance* of the *authoritative-source-of-truth* as envisioned by the DOD's Digital Engineering (DE) Strategy.

Conducting VV&A of M&S models satisfies policy and requirements set by the *Department of the Navy Modeling, Simulation, Verification, Validation, and Accreditation Management SECNAVINST 5200.46* (Secretary of the Navy 2019), and Department of Defense Standard Practice *Documentation of Verification, Validation, and Accreditation (VV&A) for Models and Simulations MIL-STD-3022 w/CHANGE 1* (Department of Defense 2012). Currently there are no published guides that detail the framework or steps

required to perform M&S VV&A. The *Department of Defense Standard Practice, Documentation of Verification, Validation and Accreditation (VV&A) for Models and Simulation MIL-STD-3022 W/CHANGE 1* (Department of Defense 2012) primarily establishes templates for the four core VV&A products (the Accreditation Plan, V&V Plan, V&V Report, and Accreditation Report), but does not provide a methodology that can be used to develop a V&V plan or framework.

B. BACKGROUND ON THE DOD'S AND DON'S DIGITAL ENGINEERING TRANSFORMATION

This section offers more specific insights into the history and background of the DOD's and DON's Digital Engineering (DE) Transformation and is primarily intended for readers who are not familiar with this background. These insights provide the context of the driving force that is pushing the implantation of M&S and digital engineering transformation in general.

The DOD envisions that M&S will proliferate and play a key role in the Navy's acquisition future. It is important to have confidence in the results of models when making decisions based on these models. This confidence is earned by the evaluation of the M&S product through a systematic, comprehensive, and rigorous application of VV&A processes. To be effective, VV&A should be a carried out in a unified manner, not disjointly or piecewise without a broader concerted strategy or plan. This thesis's research can help stakeholders derive a unified VV&A plan that can be applied to existing M&S products or future ones. The term "M&S product" is a generalization so that models and simulation models can be referred to in general. There are models whose utility is not dependent on their ability to simulate. This thesis does not focus explicitly on simulation models only. However, the focus of this thesis is on M&S products that help the users quantify answers to their questions. The type of models in mind here are more of the type employed by Operations Research (OR) or Management Science type disciplines; these often do include simulation type models.

In June of 2018, the Department of Defense (DOD) published its Digital Engineering (DE) Strategy where it states that: "To help ensure continued U.S.

technological superiority, the Department is transforming its engineering practices to digital engineering, incorporating technological innovations into an integrated, digital, model-based approach" (Department of Defense 2018, 1).The Department of Defense (2018,3) defines digital engineering as, "an integrated digital approach that uses authoritative sources of system data and models as a continuum across disciplines to support life cycle activities from concept through disposal."

The DOD's DE strategy gives examples of models that can be "developed, integrated and used as the foundation of an authoritative source of truth" (Department of Defense 2018, 5). These examples include:

- Specialty Engineering Models,
- Management Models,
- Design Models,
- Manufacturing Models,
- Verification and Validation Models,
- System Models and
- Product Support Models. (Department of Defense 2018, 5)

The following quotes are from the DOD's DE Strategy and concisely capture the roles that models play:

> Models will be used as the basis for defining, evaluating, comparing, and optimizing alternatives and making decisions. The models will span all disciplines and will provide a unified representation that enables concurrent engineering and other program activities…Models are used to answer questions, reason about the solution, support decisions, and communicate clearly and unambiguously at all levels of fidelity and across life cycle activities. (Department of Defense 2018, 7)

It is very apparent that models play a key, if not central role in the successful realization of the DOD's DE Strategy. Although not explicitly stated, the DE strategy seems to be implying a few tenets which can be conveyed as: centralization or consolidation of data; integration of data sources; and automation of data processes. These tenets appear to be bound by the more emergent and overarching *data governance*. The author interprets data governance to mean, We can wield or exploit our data to support our organization's processes and overarching objectives. The data does not do much good if we cannot use it for what we need our data to do. Data governance implies that the data is

structured or setup to support exploitation of this data and that the data is the *right* (valid) data needed—this means that data governance could push organizations to rethink how they capture data or capture data that they need but do not have. The tenets mentioned earlier complement a "systems design" point of view and characteristics. Data automation and governance in a DE environment is supported by the following statements in the DE strategy:

> Exchange of information between technical disciplines or organizations should take place via model exchanges and automated transformations, whenever possible. (Department of Defense 2018, 7)

> Governance will ensure the models and data are formally managed and trusted throughout the life cycle. In addition, stakeholders will collect, share, and maintain the models and data accurately. (Department of Defense 2018, 9)

Data governance processes will help ensure the consistency and accuracy of the data and will maintain the "authoritative source of truth" posture which enables stakeholders to make confident "data-driven" decisions (Department of Defense 2018, 9). The DE Strategy proceeds to envision that the "authoritative source of truth" should be used as the "technical baseline" from which stakeholders should make "informed and timely decisions to manage cost, schedule, performance, and risks" (Department of Defense 2018, 10).

The DE strategy outlines the larger scale goal of establishing an "end-to-end digital engineering enterprise" (Department of Defense 2018, 12) which encompasses the "digital engineering infrastructure" (Department of Defense 2018, 16) to support the models. This end-to-end construct would realize and support the necessary flow of data across the models contained in the broader digital enterprise. It is important to consider *data governance* and the end-to-end digital enterprise because these make up an important layer in the mechanization and use of models. Data governance and the end-to-end construct ultimately support the usage of models.

1. On Digital Transformation

The Naval Air Systems Command (NAVAIR) is encouraging broad Digital Transformation (DT), and Systems Engineering Transformation (SET) initiatives (efforts) across the organization. NAVAIR has also published a Digital Systems Engineering Transformation Strategy which aligns with the Department of Defense (DOD) Digital Engineering Strategy

Models play a central role in the Department of the Navy's Digital Engineering Transformation strategy. NAVAIR's Digital Transformation Lead, Jason Thomas (2023) wrote a column piece for the NAWCAD Communicator, a NAWCAD news media outlet, where he made the following insightful statements that give a glimpse of the DT picture at NAVAIR:

> Digital transformation is really about evaluating, changing, and adapting how we conduct our business, and leveraging the right digital technologies to deliver products and services to the warfighter at the pace of need. (Thomas 2023)
>
> Digital transformation is the connective tissue that permeates all facets of our work and at all sites. It is the way we're ensuring connectivity, interoperability, and integration of all domains of our work and people. (Thomas 2023)
>
> Current efforts underway include aligning and integrating our work in a seamless thread, developing and establishing authoritative models for re-use and integration, and digital environments cross multiple security levels, security for fielded and new acquisition systems, establishing state of the art digital engineering labs, and integrating cost and affordability analysis in our digital models. (Thomas 2023)
>
> Digital transformation at NAWCAD and NAWCWD is being institutionalized by looking at full life cycle development and support of our systems and the workforce that supports our joint mission of delivering integrated warfighting capability at a cost we can afford. (Thomas 2023)
>
> Our current modus operandi is insufficient to meet fielding capabilities at the pace of need and at cost we can afford. (Thomas 2023).

One of the benefits of DT is that it advances the availability, accessibility, integration/synthesis, and automation of data and information related processes in a manner

that is useful to the organization's needs. As the pace of DT picks up, it is expected that data will be more widely captured and more available, thus giving NAVAIR greater opportunities for conducting M&S.

Table 1 includes the objectives of the Department of the Navy's (2020), Digital Systems Engineering Transformation Strategy and the DOD (2018) Digital Engineering Strategy. See Table 1.

Table 1. DOD and DON Digital Engineering Transformation Strategy Goals

DON's Digital Systems Engineering Transformation Strategy objectives (Department of the Navy 2020, 6–7).	DOD Digital Engineering Strategy Goals and Focus Areas (Department of Defense 2018, 4)
1. Formalize the development, integration and use of models.	**1.** Formalize the development, integration and use of models to inform enterprise and program decision making.
2. Provide an enduring authoritative knowledge source.	**2.** Provide an enduring authoritative source of truth.
3. Incorporate technological innovation to improve the engineering practice.	**3.** Incorporate technological innovation to improve the engineering practice.
4. Establish the supporting infrastructure and environments for the Digital Engineering practice.	**4.** Establish a supporting infrastructure and environments to perform activities, collaborate, and communicate across stakeholders.
5. Transform the culture and workforce to adopt and support Digital Engineering across the life cycle.	**5.** Transform the culture and workforce to adopt and support digital engineering across the life cycle.

The Navy's strategy closely follows the DOD's; both list the development and integration of models as one of their main objectives. It is anticipated that more models will be developed, and more simulations will be employed as a standard practice for supporting analysis, evaluation, and decision making. The Department of the Navy's policy is to perform VV&A on M&S models, as stated in the *Department of the Navy Modeling, Simulation, Verification, Validation, and Accreditation Management SECNAVINST 5200.46* (Secretary of the Navy 2019).

2. On Systems Engineering Transformation

NAVAIR is responsible for the acquisition and sustainment of naval aviation capabilities. NAVAIR is driving Systems Engineering Transformation (SET) efforts in an attempt to speed up the systems engineering process, which in turn speeds up acquisition life cycles (Moschler 2019). The overarching goal is to deliver capabilities to the fleet in a faster more cohesive manner. The Navy's Digital Engineering Strategy specifies that digital engineering is employed by using digital models to "analyze, design, test, produce, field, and sustain capabilities" (Department of the Navy 2020, 8). Model-Based Systems Engineering (MBSE) is mentioned alongside digital engineering, but digital engineering is implied to be a broader domain that includes MBSE. The following are definitions for MBSE and Digital Engineering:

> Model Based Systems Engineering (INCOSE) –[the] Formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. (Department of the Navy 2020, 8)

> Digital Engineering (Navy) – An integrated, computation-based approach that uses authoritative sources of system data and models across disciplines to support life cycle activities from concept through disposal. (Department of the Navy 2020, 8)

MBSE can be broadly defined as systems engineering where the systems engineering process is facilitated by models throughout the system's life cycle regardless of any specific type of model. Other terms that are similar to MBSE include "model-driven" or "model-centric." However, the general community in NAVAIR appears to have understood the definition of MBSE as primarily consisting of the "Three Pillars of MBSE*"* as defined by Delligatti (2013, 4): modeling language, modeling method, and modeling tool. Based on the author's personal observations, Delligatti's training is broadly used in NAVAIR and is quite popular among those taking up MBSE.

C. BACKGROUND ON THE COST READINESS IMPACT MODEL

Speed to the fleet is a NAVAIR initiative championed by Vice Admiral Dean Peters when he was the NAVAIR commander. This goal encourages delivering capabilities faster,

at a pace that maintains technological advantage. This initiative heavily relies on the digital transformation of current systems engineering and acquisition processes. Improving the fleet's *readiness* was another complementary goal. The challenge of improving readiness brings us closer to the central problem that derived this proposal's research question. Fleet readiness is not free, and usually involves cost trade-offs across the fleet's capabilities and readiness, e.g., one platform may be prioritized over another given budget constraints.

Affordability is the subsequent, additional initiative championed by Peter's successor, Vice Admiral (VADM) Carl Chebi, who is the current NAVAIR commander (at the time of publication, 2023). "One Team One Fight – Delivering the Warfighting Capability the Fleet Needs to Win, at a Cost We Can Afford" (NAVAIR News 2022) is a slogan championed by NAVAIR leadership and serves as a good summary for the current direction and trade-offs that NAVAIR is pursuing under VADM Chebi's command.

NAVAIR leadership's task is to make optimal decisions that balance *mission readiness* and *affordability*. Objectively supporting these types of decisions requires analyzing and evaluating a myriad of data and information. NAVAIR has developed and is incrementally improving and maturing several analytical models that receive input data from O&M databases and calculate results that forecast mission readiness and affordability impacts. These models and the supporting model infrastructure aggregate to form the Cost and Readiness Impact Model (CRIM). CRIM packages and unifies the capabilities of two analytical models and delivers both models' capabilities to NAVAIR O&M users. One model is the Driver Tree (DT) model. The other model is best defined as the multi-echelon, multi-indenture model, often referred to as the SIMLOX model, because the model is configured to run on SIMLOX, a discrete-event Monte Carlo simulation software by Systecon and is used for predictive analytics.

Readiness and *affordability* are of key value to NAVAIR. There is significant motivation to improve both areas and optimize the trade-offs between the two. CRIM gives NAVAIR leadership and stakeholders a way to analyze and evaluate investment inputs and estimate cost and readiness benefits and trade-offs.

D. OBJECTIVES AND CONTRIBUTION

1. Primary Objective

This thesis seeks to develop a re-usable framework and process for supporting VV&A of M&S models. This framework and process will be applied to a currently implemented M&S subject, the Cost Readiness Impact Model (CRIM), so that a concrete application of this framework and process can be demonstrated and evaluated for the benefits that it can offer. The M&S VV&A framework and process developed here is essentially the methodology that was carried out during this research. The methodology produced an MBSE model of the M&S system (CRIM) which includes V&V elements and their relationships to the M&S system entities. This MBSE model can be used as a map or framework to inform and guide M&S VV&A activities.

CRIM addresses management decisions that balance affordability and fleet mission readiness—a top priority for NAVAIR. CRIM is an excellent subject to apply the M&S VV&A framework and process to because of CRIM's characteristics. Some important characteristics of CRIM are its: (1) large structural scale, (2) complexity, (3) systems-of-systems structure, and (4) end-to-end data integration and automation. CRIM's characteristics will help exercise the application of this thesis's VV&A framework and process more fully. Results from applying the VV&A framework and process on CRIM offer value to NAVAIR because the results will help the planning of CRIM's undergoing VV&A efforts.

2. Secondary Objective

To the DOD professional, especially the program manager (PM) or someone in a similar role who must undertake the responsibility and accountability of successfully supporting M&S products and their VV&A efforts:

It is the hope of the author to sufficiently and appropriately address concepts essential to M&S VV&A and other accompanying concepts used to support this research, such as Model-Based Systems Engineering (MBSE) which plays a key role in the methodology. This document provides the DOD professional general awareness of what exists in the M&S VV&A domain and presents basic concepts and terminology that would

allow the DOD professional to manage M&S projects more effectively. It is the hopes of the author give DOD professionals a better understanding of these concepts so that they can communicate more effectively with technical subject matter experts (SMEs) and with other non-technical stakeholders such as sponsors, leadership, and partners.

As a result of this objective the literature review in Chapter II presents an extensive compilation of concepts and information. However, the aim is **not** to define the full, exhaustive set of concepts for each of the domains presented, but primarily to define the collection of concepts that align to supporting the objective of this thesis research.

3. Contribution

This thesis provides a methodology that develops an MBSE model of the M&S product or system. This methodology is applicable to any M&S product or system; thus, it can be reused. The MBSE model can serve as the M&S VV&A framework because it identifies the M&S and V&V entities and their relationships. The MBSE model framework organizes M&S and V&V entities and relationships in an ontologically structured manner. From this model, users can infer information about what V&V methods are applied to what elements of the M&S system. This contribution is important because a reusable VV&A framework supports evaluating the effectiveness and suitability of an M&S product and supports the confident use of the M&S product. Currently there are no published guides that detail the framework or steps required to perform M&S VV&A. The *Department of Defense Standard Practice, Documentation of Verification, Validation and Accreditation (VV&A) for Models and Simulation MIL-STD-3022 W/CHANGE 1* (DOD 2012) primarily establishes templates for the four core VV&A products (the Accreditation Plan, V&V Plan, V&V Report, and Accreditation Report), but does not provide a methodology that can be used to develop a V&V plan or framework. M&S stakeholders are left to their own devices to develop a VV&A framework for their M&S product. This is not necessarily a bad thing, but it is a daunting, open-ended task that requires much effort, logical reasoning, and definition in order to develop a concrete VV&A framework that outlines VV&A entities and V&V methods and actions.

As a byproduct, this thesis's research contributes significant value to the NAVAIR Cost Readiness Impact Model (CRIM) stakeholders by providing them a system architecture, MBSE model of CRIM, which by design, captures V&V elements and relationships that will support CRIM's VV&A efforts.

An additional contribution of this research is the rubric that the author developed to qualitatively evaluate the CRIM MBSE model for its effectivity in supporting M&S VV&A elements. The author believes that this rubric can be used to evaluate any descriptive graphical model of an M&S product or system. This rubric was used to evaluate the CRIM MBSE model for its effectiveness in supporting CRIM's M&S VV&A efforts.

E. THE RESEARCH QUESTIONS

The primary research question is:

Can Model-Based Systems Engineering (MBSE) be leveraged to develop a re-usable framework and process for Validation Verification and Accreditation (VV&A) of Modeling and Simulation (M&S) models?

The secondary research question is:

If a new M&S model is being conceptualized or developed, can the framework developed by this thesis help guide and facilitate the design and development of M&S architecture and system requirements?

F. SCOPE, LIMITATIONS, AND ASSUMPTIONS

This research is limited to providing a framework and methodology for deriving an MBSE model of the M&S product, which can support M&S VV&A efforts. This research heavily employed MBSE modeling activities. The author's ability to completely model all the necessary CRIM system architecture elements was due primarily to his lack of knowledge about the elements that could not be modeled completely or precisely.

A successful modeling effort requires interdisciplinary cooperation. Ideally the author would have engaged with the CRIM subject matter experts (SMEs) in modeling activities. The author is part of the CRIM project team, so he has general knowledge of

CRIM's systems architecture and access to CRIM documents. However, CRIM's project objectives prioritize the development and integration of CRIM components. CRIM also supports simulation requests from NAVAIR Program Management Offices-Air (PMAs). The CRIM development team was very busy accomplishing this other work. Therefore, fundamental systems engineering and systems architecture activities, such as systems modeling, are not prioritized or formalized by the CRIM project team. This is not the fault of the CRIM project team. CRIM sponsors and advocates have given the CRIM project team an urgent and ad-hoc request to deliver capability to the fleet as quickly as possible. Thus, compressed timelines and limited project resources have focused primarily on producing *system products*, but not on developing *enabling products* such as technical documentation, e.g., technical requirements, system technical baselines, measures of effectiveness, etc. This mode of project execution has been expressed at times by the CRIM project team as "We're building the airplane as we are flying it."

The author collaborated with ManTech to model the CRIM system architecture, but these modeling efforts were done independently from the CRIM development team and simulation model SMEs. The author relied on project and technical documentation that is appropriate for high-level concept communications, but too abstract or too ambiguous to precisely model CRIM's finer details. The author was specifically looking for the conceptual model details of CRIM's Driver Tree model and multi-echelon, multi-indenture SIMLOX model. The conceptual model captures modeling assumptions and working principles of the computerized/implemented model. The author was able to get high-level details for these conceptual models, but not sufficient details to fully model the more discrete details.

G. ORGANIZATION OF STUDY

This chapter covers background information about the DOD's and DON's digital engineering transformation strategies; NAVAIR's readiness and affordability objectives; and the Cost Readiness Impact Model. This background provides the context that explains what inspired this research and why this research is important.

Chapter II is a literature review of concepts that the author used to support his research rational, methodology, and evaluation of results. This chapter is intended to provide the reader all the accompanying concepts that guided the author's research. This chapter can be used independently by readers who wish to get an overview in the domains of: Systems Theory, Systems Architecture, Systems Engineering, Systems Engineering Validation, Verification and Accreditation (VV&A), Model-Based Systems Engineering (MBSE), The MagicGrid Framework, Modeling and Simulation (M&S), M&S VV&A, and Multi-Echelon, Multi-Indenture Inventory Systems theory. The author also includes a short section that covers previous Cost Readiness Impact Model V&V research work performed by the Naval Postgraduate School (NPS).

Chapter III presents the main MBSE modeling tasks that were performed as part of this research's methodology. The author follows the same pattern for each task: (1) The author provides modeling objectives and guidance for each modeling task—this provides readers useful insights that that they can follow if they wish to adopt or modify this method. (2) the author provides a table that summarizes the information or artifacts used to support the modeling task; (3) The author summarizes what was modeled and provides images of his modeling work; (4) the author provides a section that outlines future work that can be done to improve what was modeled.

Chapter IV evaluates the modeling results. The author provides a rubric that outlines effectiveness criteria that is used to evaluate the MBSE model for its ability to support M&S VV&A efforts. The author scores his results based on each of the criteria presented.

Chapter V concludes the research work; it summarizes the result findings and provides a concluding answer to the primary and secondary research questions. This chapter also recommends future research that can be performed.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LITERATURE REVIEW AND THEORETICAL FRAMEWORK

The concepts and terminology from the below listed areas were used by the author to support his research rationale and logic. This chapter provides readers valuable concepts that can be applied to M&S VV&A. This chapter covers concepts and terminology from the following various areas:

- Systems Theory
- Systems Architecture
- Systems Engineering
- Systems Engineering Validation, Verification and Accreditation (VV&A)
- Model-Based Systems Engineering (MBSE)
- The MagicGrid Framework
- Modeling and Simulation (M&S)
- M&S VV&A
- Multi-Echelon, Multi-Indenture Inventory Systems theory.

This chapter additionally includes a short summary that covers previous Cost Readiness Impact Model V&V research work performed by the Naval Postgraduate School (NPS).

A. SYSTEMS THEORY

This section aims to introduce some of the key philosophical aspects of systems theory and how systems engineers (or others) think about and define systems. These concepts are presented so they may be distinguished in a foremost manner even though they are widely and often used in Systems Engineering, MBSE, M&S, and other scientific and technical disciplines. These concepts are incorporated throughout the methodology

presented in this research; thus, it is important for these concepts to be identified, defined, and appropriately understood.

There are many definitions of a system, most are similar with varying degrees of details depending on the perspective and considerations of the one providing the definition. Here are some simple but effective definitions and characteristics of a system:

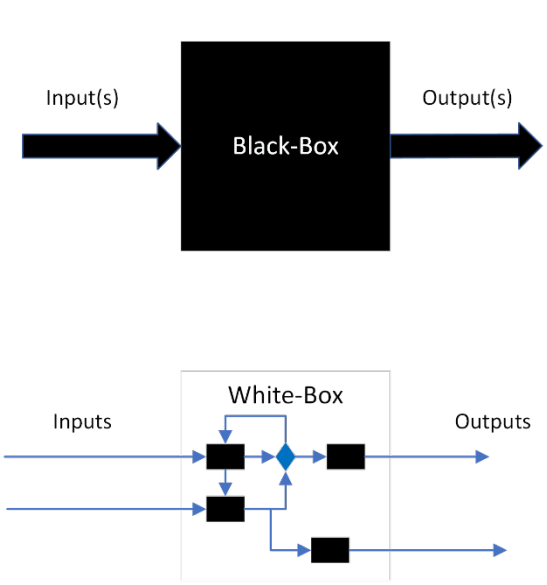
- A system is a set of interrelated components functioning together toward some common objective(s) or purpose(s) (Blanchard and Fabrycky 2011, 3).
- A system is an arrangement of parts or elements that together exhibit behaviour or meaning that the individual constituents do not. (INCOSE n.d.)
- Systems can be either physical or conceptual, or a combination of both. (INCOSE n.d.)
- *Physical systems* are those that are manifested in physical form [time and space]. They are composed of real components and may be contrasted with *conceptual systems*, where symbols represent attributes of components. Ideas, plans, concepts, and hypotheses are examples of conceptual systems. (Blanchard and Fabrycky 2011, 6)

Systems can be natural (from nature) or man-made engineered systems. Regardless of which type of system is studied, concepts from systems theory allow us to define and reason about existential aspects of the *system of interest* so that we may better understand it and communicate our understanding to others.

1. Systems Hierarchy, Decomposition, Black-Box And White-Box Abstraction, And General Logical Abstraction

"Systems thinking" or the "systems viewpoint" looks at the system from the "top down rather than from the bottom-up" (Blanchard and Fabrycky 2011, 5). This means that a system is viewed at the most abstract level first, depicted as a "black-box," (Blanchard and Fabrycky 2011, 5) and placed in the context of its environment and how the black-box system interacts with its environment. The next step is to "open" the black-box to see what is in it. By opening the black-box we see the "internal" view of the system's composition (or decomposition depending upon someone's perspective) and we are able to see or acknowledge the subsystems. We now have a "white-box" view of the system, but at this level, we may still wish to abstract the subsystems as black-boxes. This black-box, white-

box, black-box perspective is recursive, and usually stops at the lowest level of concern—typically an individual component. See Figure 1.



The black-box is an abstraction of a system or overall system function. *Internal* components are abstracted away.

The inputs and outputs can also be abstracted to the level where an overarching description must be used.

A white-box perspective allows the viewer to see the *inside* parts of the black-box. This view offers a more concrete view of the internal elements and removes a certain level of ambiguity of the system because it more precisely defines the truthful existence or representation of system and how it works. Inside the white-box are other black-boxes.

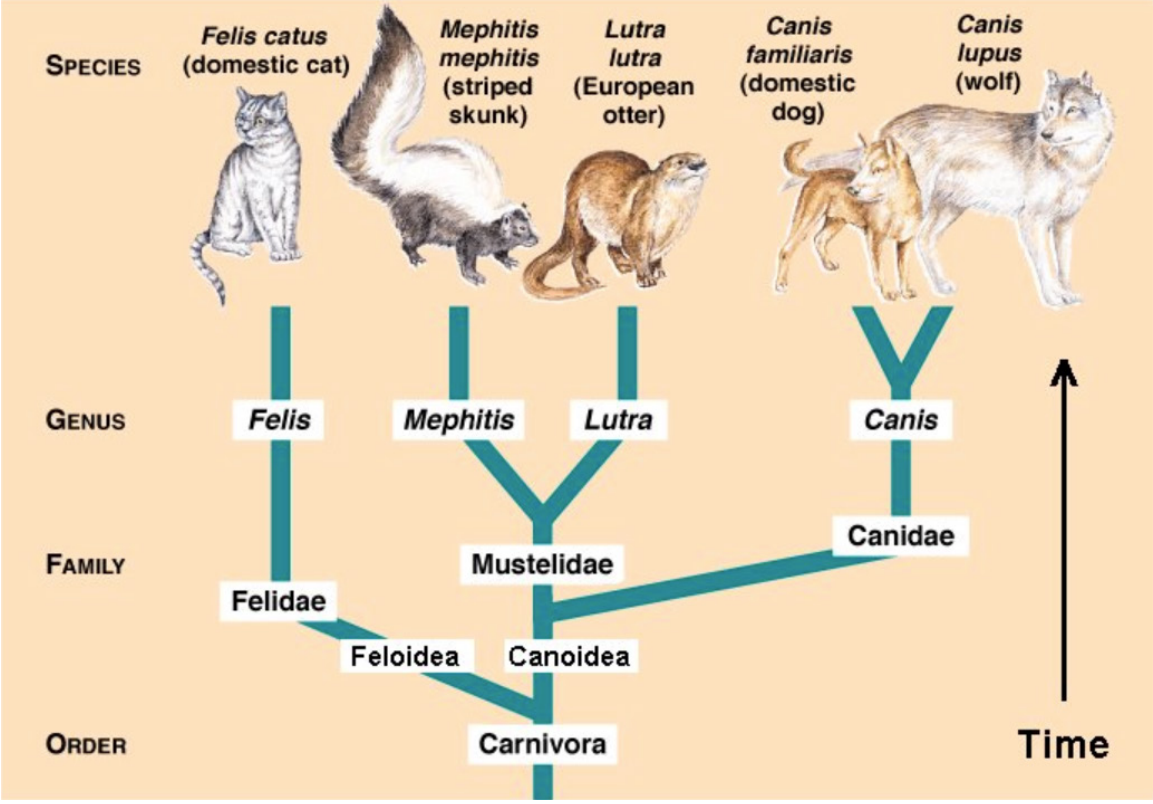
Decomposition is the widely used term that describes when a system or system element is reduced from its abstract state to its more concrete parts. A user-defined convention for *hierarchy* levels is established to help organize and classify the levels of decomposition.

Figure 1. Black-box and white-box examples

The trail left by this recursive abstract-to-concrete decomposition process defines a hierarchy of composition. *Hierarchy* is a very useful concept for analyzing systems. Two forms in which hierarchy can describe systems is through abstraction of *composition* or inversely, *decomposition*, but there is also hierarchy of *classification abstraction* often referred to as a *taxonomy*.

A classification abstraction or taxonomy is a useful systems reasoning device, but it is less intuitive than concepts of composition or decomposition, thus it is worthwhile to cover the concept of abstraction with regards to classification. A taxonomy is more meta-physical and deals with concepts of the *essence* of something and its classification identity. A taxonomy hierarchy is traversed either to the more generalized concepts "upwards," or "downwards" to the most specialized concepts—down even to the level of manifestations/instantiations of individual things that concretely represent the upper logical concepts. This type of relationship is captured by systems engineering graphical

modeling languages such as the Systems Modeling Language (SysML). A *generalization* relationship is interpreted by saying that the more concrete thing is a "type of" the more abstract thing. In a parent-child taxonomy hierarchy a "child" is the subclass of the superclass "parent" and inherits the parent's properties. Figure 2 shows an example of this type of taxonomy, illustrating the classification of the animal kingdom's order, *carnivora*.



Hierarchy of abstraction and classifies the specialization of each subclass – in this case the taxonomy has exclusive branches, but in other taxonomy examples it is possible to have an individual belong to more than one parent class. Source: Carr (n.d.).

Figure 2. Taxonomy example.

The taxonomy in Figure 2 has *mutually exclusive* or *disjoint* branches, meaning that some member of a branch cannot simultaneously be a member of another branch. For example, at the genus level, members can only belong to one genus, i.e., Felis, Mephitis, Lutra, or Canis. The instances or individuals of this taxonomy are the animals that exist in time and space. Each higher classification is an abstraction of these individual animals.

Abstraction levels can be used to classify or categorize groups with similar features or attributes up until the most generalized level that possesses the commonality among *all* the members.

Usually, a taxonomy is considered to possess an arboresque or tree-like form with separate branches, but taxonomies could be more complex. For example, there could be two classes of automobiles, a *car* or a *truck*. These classes are classified primarily by the automobile's structural form. The automobile is the *super-class* or *parent*. However, one can add another classifying property such as color. Now, a class can be defined by automobiles that are blue, which would include members from the *blue car* and *blue truck* classes but would exclude red (or any other color) car or truck. This notion of a taxonomy follows *set-theory* logic that would allow a taxonomy hierarchy to be quite complex.

Although a hierarchical structure could be maintained, some would not consider a complex, set-theory based taxonomy to be a taxonomy anymore. Tamr Core. (n.d.) recommends following the mutually exclusive, collectively exhaustive (MECE) principle when designing or managing a taxonomy. A complex set-theory based taxonomy could be considered to be more of an ontology, which defines entities and their relationships. In an ontology, relationships can exist across any classification, e.g., a *person* owns a *car*. In this example, the *person* and the *car* are two different disjoint classes linked by the relationship *owns*. However, a taxonomy should focus on the hierarchical structure of *belonging*. Normally, a person and a car would not belong in the same hierarchy structure or taxonomy, but there could be an application or perspective that could relate their belonging, e.g., *things that move* or *things that have mass.*

Abstraction philosophy goes back to Plato's theory of "forms" or "ideas," where Plato theorized that a thing's essence transcends its physical form, in space and time, and the thing's true absolute, ideal nature is the abstract idea of the thing. The *essence* is the property or properties that make the thing what it is; without its essence the thing loses its true identity. In contrast, an *accident* is a property or set of properties that a thing may possess but are not necessary for the things' true identity; in other words, the thing can lose its accidental properties and still remain its true self. One may have heard the philosophical "what makes a chair a chair?" argument attributed to Plato's theory. In this argument,

accidental properties of a chair may be its color, the material, the shape, etc. There are many instances of a chair; changing, removing, or substituting some of these chair properties will still result in a chair as long as the chair's essence is intact. Figure 3 illustrates a generalization relationship, which takes a more concrete instance or class and abstracts it to a higher classification.

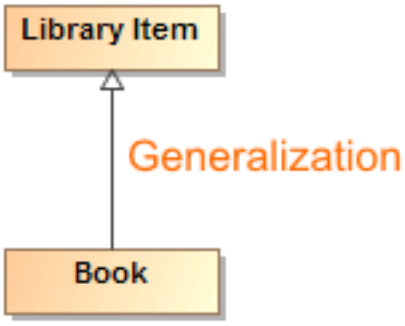


Figure 3. SysML generalization relationship. The syntax in SysML should read a "Book" is a type of "Library Item." Source: No Magic (n.d.).

It is useful for a systems thinker to think about or identify the essence of a system and its functions in the context of the system's purpose or problem domain, stakeholder needs, utility and value delivered. This will help conceptualize the "right" system which serves as the foundation for design and is aligned to the stakeholder needs. The reason why abstraction of classification was covered more extensively in this section is because abstract thought was used as part of this research's methodology, specifically when modeling using Cameo Systems Modeler (CSM) and SysML. In this scenario, the experimental subject of this research, CRIM, pre-existed a formally defined logical architecture which helps provide *design-to* specifications and other information. Concrete implemented parts of CRIM already exist in the solution-domain, but it is difficult to trace them to more logical, solution-neutral elements in the problem-domain that should form the essence or basis of CRIM's design. This is actually a common occurrence when there is a legacy or existing system and the stakeholders of the system wish to develop an MBSE model of the system. When documentation of the system is poor or poorly managed, the systems engineers and modelers have to work "backwards" to derive what should be the

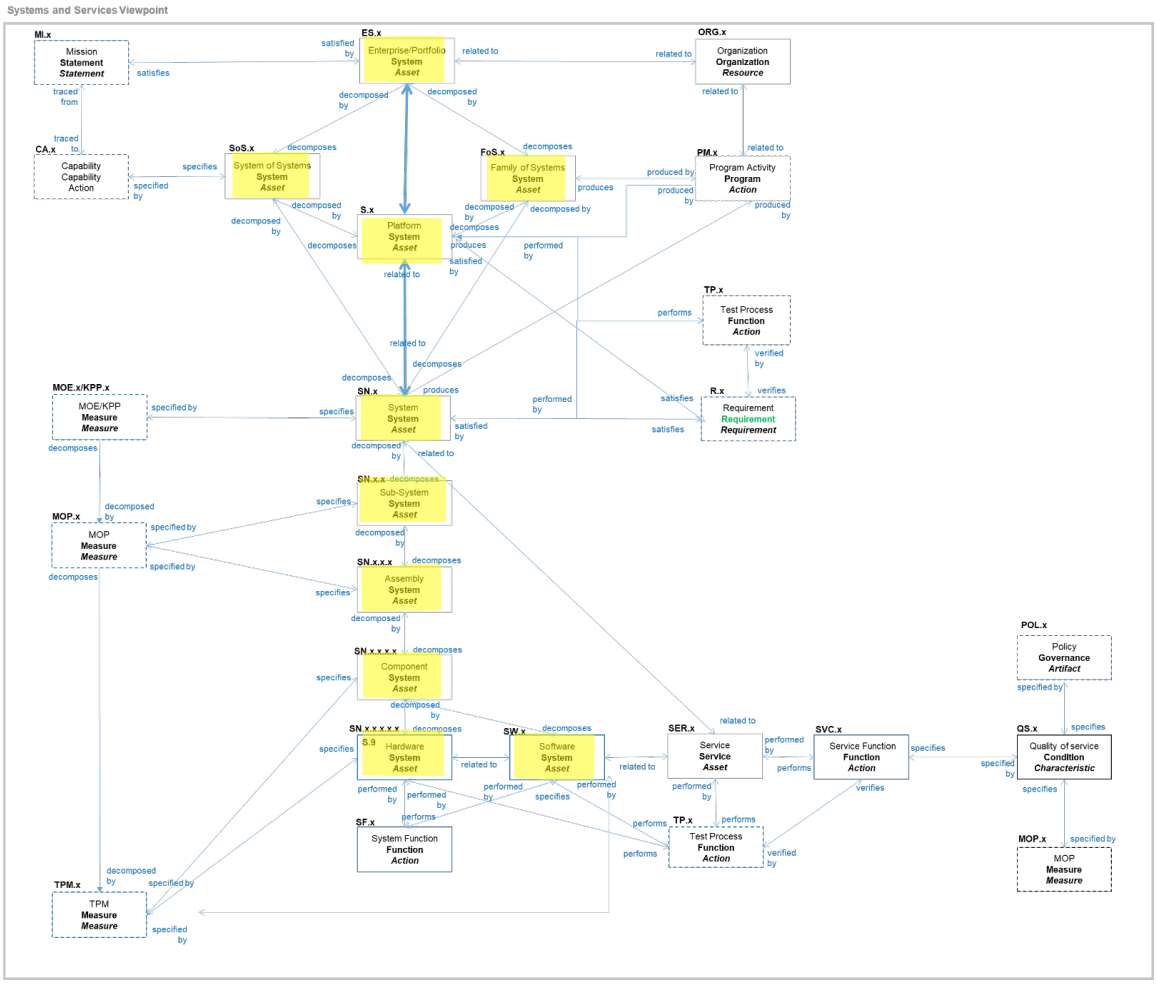
higher-level technical baselines such as the functional baseline and the allocated baseline, sometimes even the requirements baseline needs to be developed or re-constructed. The problem-domain, solution-domain, and solution-neutral concepts are covered later in this chapter.

Figure 4 is a viewpoint from Vaneman's (2022) conceptual data model (CDM) relationships based on the Marine Corp Systems Command (MCSC) ontology and shows an example of both a composition hierarchy and an abstraction classification hierarchy or taxonomy. The syntax in this CDM shows the composition hierarchy through relationships represented by arrows with the written relationship type at the end of the arrow. The relationships are usually bilateral so that an arrow has relationship types specified at both ends. In this example with the highlighted elements, the relationships are inverse of each other: "decomposes" and "decomposed by." The highlighted elements show a hierarchy trail. If the hierarchy is followed going "down" from the larger more complex structural entities to the lowest, the relationships would read as follows:

- Enterprise/Portfolio "decomposes" Family of Systems
- Enterprise/Portfolio "decomposes" System of Systems
- System of Systems "decomposes" Platform
- System of Systems "decomposes" System
- Family of Systems "decomposes" Platform
- Family of Systems "decomposes" System
- Platform "related to" System and System "related to" Platform (symmetric relationship)
- System "decomposes" Sub-System
- Sub-System "decomposes" Assembly
- Assembly "decomposes" Component

- Component “decomposes” Hardware
- Component “decomposes” Software
- Hardware “related to” Software and Software “related to” Hardware (symmetric relationship)

One may notice that the way the CDM relationships read may not be very intuitive. An alternative, equivalent, and inverse relationship to “decomposes” would be “composes” such that a Sub-System “composes” a System. This way of reading the relationship is alternative to reading the original CDM relationship which reads as: Sub-System “decomposed by” System. The original syntax is not naturally intuitive, but it makes more sense when the logic is verified by using the “composes” relationship. In this case “composes” and “decomposed by” are logically equivalent. Using the “composes” relationship reads more naturally, and makes the inverse relationship, “decomposes,” more natural too. Read more implicitly, a syntactically more appropriate way to read the examples from the CDM is as follows: System “decomposes [into]” Sub-System(s).



Zoom in to see smaller details. A slightly larger version of this image is available in the appendix. Source: Vaneman et al. (2022,41).

Figure 4. Systems and Services Viewpoint from the Marine Corp Systems Command (MCSC) Ontology and Conceptual Data Model (CDM).

A hierarchy of abstraction or taxonomy is also present in the CDM, but it is not immediately noticeable. The taxonomy syntax is in the form of some sort of "nested notation." Readers familiar with nested notation in SysML may understand the reference to the nested notation term. What this means is that the taxonomy relationships are expressed inside the element's box. The system element box has three rows or levels of entities with the most concrete entity being at the top and the most abstract at the bottom. The way this that this relationship is read is as follows: Data Type is a "type of" Sub-Entity; Sub-Entity is a "type of" Entity. A specific example from the CDM is System is a "type

of" System and System is a "type of" Asset. The relationship System is a "type of" System is a trivial self-identity relationship, but it may be more useful in other cases. Another example is Sub-System is a "type of" System which is a "type of" Asset.

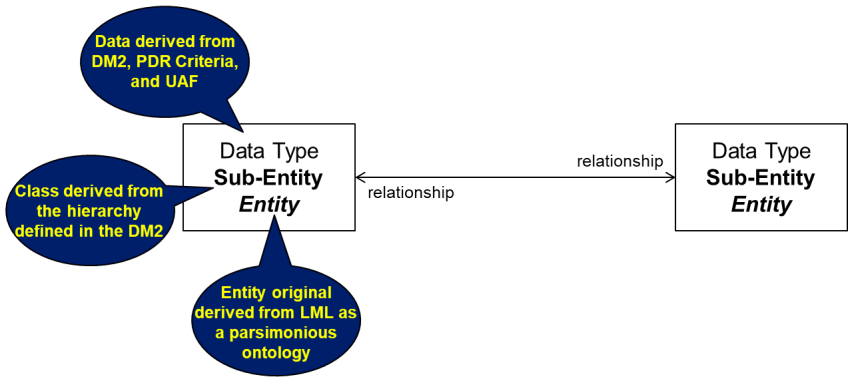


Figure 5. Conceptual Data Model Format. This format shows the graphical language syntax used in the CDM. Source: Vaneman et al. (2022, 37).

2. The Machine Age's Mechanistic Understanding Of World: Reductionism And Determinism

The concepts in this section become important when considering system behavior properties, such as operations or functions of mathematical equations or models and of software systems or programs. This is because math and software can be very deterministic or mechanical in behavior when it does not involve any randomness, such as random input variables. Modarres (2016, 166), who has published various textbooks in engineering risk analysis which involves analyzing software, states that "software has deterministic behavior, whereas hardware behavior is both deterministic and probabilistic." The general deterministic nature of mathematical equations and models such as those from physics, and of software programs is universally accepted. Mathematical models can form the underlying concepts used to build a simulation model which is often implemented by software. Therefore, the concepts discussed here are helpful for analyzing, understanding, and describing M&S products.

Some exceptions or unique cases of software that cannot be considered to be mainly deterministic are that of models and simulation programs that employ complex "adaptive

programing" (Pace 2004, 167) because these programs change during operation (Pace 2004). These "adaptive programs" use technology and techniques that may include "artificial intelligence [AI], expert systems, genetic algorithms, fuzzy logic, machine learning [ML], etc.," which allows the modeling and simulation "structure and parameters" to "differ from one run/iteration to the next as the program adapts" (Pace 2004). These adaptive programs exhibit advanced random or probabilistic behavior and are different than the M&S systems that are considered here, which primarily consist of a deterministic framework that processes stochastic or random/variable values.

Determinism is a concept that has been well defined and applied by philosophers and mathematicians. Here is a definition that is deemed appropriate for the context of this thesis research.

> A deterministic system is a system where an initial state completely determine the system's future states. Thus, there is no randomness in producing the future states. If a deterministic system is given some initial inputs, the model will produce the same states every time. (Swarthmore College 2009)

An individual element such as a number value can be deterministic, or even more complex elements such as equations, set of equations or mathematical theories, or as discussed here, a simulation model.

Causal-based laws or *causality* are a fundamental aspect of determinism, particularly when determinism is viewed from a *mechanistic* perspective which depicts elements as being rigidly linked and dependent as if they were part of a machine. In a mechanistic perspective, all phenomena could be described by a *mechanism* and explained by the relation of *cause-and-effect*; "one thing or event was taken to be the *cause* of another (its *effect*)" (Blanchard and Fabrycky 2011, 14). These causal-based relationships are *axiomatic* – self-evidently true, unquestionable, and accepted as factual without proof. They are stated by *fiat* (Latin for "it shall be" or "let it be done.") or by decree or arbitrary order. Therefore, such a rigid framework offers useful logical reasoning: cause and effect are "*necessary and sufficient*" (Blanchard and Fabrycky 2011, 14) for each other. The statement *necessary and sufficient* has a logical implication. This means that if a cause exists or existed (is true) then that implies the existence of the effect (is also true);

conversely, if the effect exists or existed (is true) then that implies that the cause exists or existed (is also true). Alternatively stated, there can be no effect without a cause, and no cause without an effect.

$$Cause \underset{implies}{\Longleftrightarrow} Effect$$

Given a deterministic framework, the idea of *reductionism* is appropriate and useful to analyze and describe the deterministic subjects. *Reductionism* "consists of the belief that everything can be reduced, decomposed, or disassembled to simple indivisible parts" (Blanchard and Fabrycky 2011, 13). Reductionism can be applied to something that is complex and hard to understand, as a whole, and where a better understanding might be achieved if its individual elements are better understood. There is a three-part analytical process for reductionism: the first part involves reducing or deconstructing a complex thing down into its independent and indivisible parts; the second part is explaining the behavior of the individual parts; finally, the third part is aggregating the individual explanations, which are partial explanations of the whole, and synthesizing the explanation of the whole complex thing (Blanchard and Fabrycky 2011). This technique is effective when solving a problem whose nature is mathematical because it allows the problem solver to: (1) decompose the problem into simpler, more manageable problems, (2) compute and obtain solutions for individual problems, and then (3) aggregate individual solutions to obtain the solution of the whole. Blanchard and Fabrycky (2011, 14) state that during the *Machine Age*, "understanding the world was taken to be the sum, or result, of an understanding of its parts, which were conceptualized as independently of each other as was possible." This was the mathematical and scientific philosophy used for understating the world during the Industrial or *Machine Age* (Blanchard and Fabrycky 2011). Essentially, according to this philosophy, *the whole is equal to the sum of its parts*.

3. The Systems Age: Expansionism, Teleology, Synthetic Thinking, Systems Approach

The transition from the *Machine Age* to the *Systems Age* happened around the 1940s (Blanchard and Fabrycky 2011). It is notable that this timeframe also corresponds to the Second World War (WWII) which likely contributed to the necessity of approaching

problems and developing solutions using a more modern systems approach. The author speculates that there were several drivers to this transition:

- Advancements in technology: by the 1940s, technology that was present at the time would be easily recognizable today: Airplanes, rockets, tanks, boats, road-vehicles, radio, telephones, radar, film, etc., were being widely employed by many nations of the time and seeing steady advancements in performance.

- Increasing Complexity: Technology such as aircraft, which was probably one of the main technologies driving systems engineering at the time, were seeing increasing complexity as engineers were designing more functions enabled by more parts. This was made possible also by improving form-fit-function of technology – more powerful, more efficient, smaller, and lighter engines, on-board radio, and sometimes radar, more sophisticated hydraulic or electro-mechanical controls, gyroscopic-targeting/computing gun sights, etc.

- System Purpose (teleology): These complex technological systems were being designed with a purpose or role in mind. Designers were likely influenced by questions like: What mission does the system need to do, what does it need to do best, etc., what is the optimal performance and what are the trade-offs? A special purpose would help drive the design so that the optimization of design was done at the system level and not as a part level.

- Education and Skills: America's higher education saw formative years from 1890 to 1940 where higher education's modern form emerged around 1910 (Goldin and Katz 1999). Some factors that contributed to this transformation are the "application of science to industry" and the growth of science and experimental methods" (Goldin and Katz 1999, 38). For industry in the late 1800s "there emerged a growing importance of chemistry and physics, most notably in the manufacture of steel, rubber,

chemicals, sugar, drugs, nonferrous metals, petroleum, and goods directly involved in the use or production of electricity (Kevles 1979)" (Goldin and Katz 1999, 38-39). Firms, federal and state governments were hiring "trained chemists and physicists" at an "increasing rate" (Goldin and Katz 1999, 39). Goldin and Katz (1999, 39) further specify that, "the number of chemists employed in the U.S. economy increased by more than six-fold between 1900 and 1940 and by more than three-fold as a share of the labor force; the number of engineers increased by more than seven-fold over the same period (Kaplan and Casey, 1958, table 6)." In the late 1800s universities began to subdivide and specialize subjects and such became defined as separate specialized fields (Goldin and Katz 1999). According to Goldin and Katz (1999), the higher education enrollment rate from 1940 to 1970 nearly quadrupled, but it should be noted that the enrollment rate "increased more than three-fold from 1910 to 1940, and by five-fold from 1890 to 1940," (Goldin and Katz 1999, 41) indicating that enrollment rates were also high in the earlier periods.

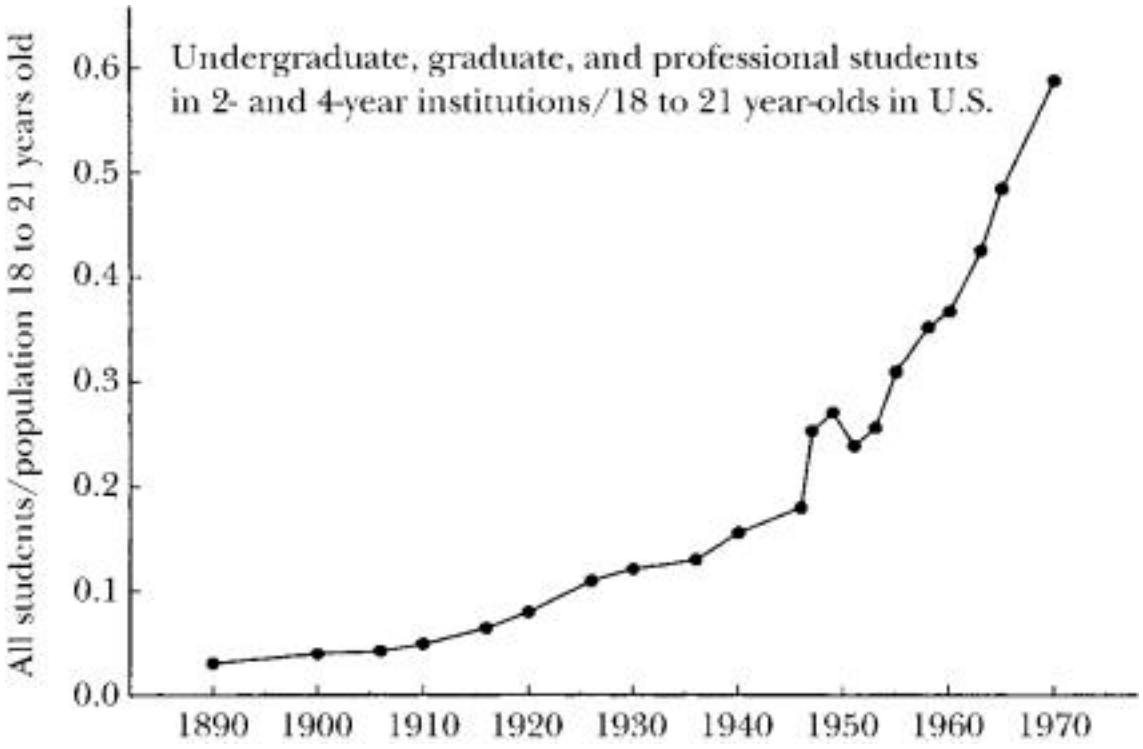


Figure 6. Students in Two- and Four-year Institutions in the United States as a Fraction of 18 to 21Year-Olds: 1890 to 1970. Source: Golding and Katz (1999, 41).

During the *Systems Age*, the doctrine of *expansionism* was employed by thinkers. Expansionism considers that "all objects and events, and all experiences of them, as parts of larger wholes" (Blanchard and Fabrycky 2011, 14). This perspective is different, but it is compatible, and complementary to reductionism and allows the designers to think about elements not only as interrelated parts, but as a higher-level system with emergent properties (Blanchard and Fabrycky 2011). However, there is somewhat of a dichotomy in philosophy between the two concepts: In *reductionism*, the functioning whole is equal to the sum of its functioning parts, but in *expansionism* "the sum of the functioning parts is seldom equal to the functioning of the whole" (Blanchard and Fabrycky 2011, 15). The quoted statement about expansionism by Blanchard and Fabrycky (2011) invites the philosophical question, *can the value of the functioning whole only be greater than or equal to the sum of the functioning parts, or can it be less?* That argument will not be entertained here, but in general, systems thinkers predominantly consider the condition that the

functioning whole **is** greater than the sum of its functioning parts due to the synergistic nature of the cooperating elements.

Two modes of thinking behind the reductionist and expansionists viewpoints are defined by Blanchard and Fabrycky (2011, 15); these are the "analytical thinking" and "synthetic thinking" modes:

1. Analytical thinking: is "outside-in" thinking where an "explanation of the whole was derived from explanations of its parts" (Blanchard and Fabrycky 2011, 15). Analytical thinking supports the reductionist perspective.

2. Synthetic thinking: is "inside-out" thinking and allows one to achieve understanding that is not obtained through analysis of "collective phenomena" (Blanchard and Fabrycky 2011, 15). Systems thinking is a specialization of synthetic thinking and happens when it is applied to systems problems (Blanchard and Fabrycky 2011). Blanchard and Fabrycky (2011, 15) specify that, "the synthetic mode of thought, when applied to systems problems, is called *systems thinking* or the *systems approach*" . Synthetic thinking supports the expansionist perspective.

Systems thinking is *teleologically* oriented. Teleology, simply put, is the philosophy that considers the final goal or ultimate purpose of things. Some related things have a higher purpose or a richer value than others. For example, one might ask, *"Does one live to eat, or does one eat to live?"* Teleologically addressing such questions, in the context of a system, can help the designer understand the higher purpose of some design (emergent) functions over others. During the Systems Age, designers confronted complex problems that required complex solutions. Such solutions took the form of a system. Ideally, the system would address or solve most of the components of the problem, thus satisfying the purpose for which the system was built.

B. SYSTEMS ARCHITECTURE

1. Systems Architecture Overview

Understanding system architecture concepts is critical for understanding and defining systems. The concepts related to the system architecture provide designers a well-organized, rational conceptual framework that allows designers to define a system formally and efficiently. Defining a system goes hand-in-hand with design; the design process where the system architecture is developed is called *system architecting*, and the designer that performs system architecting is called a *system architect*. System architecting is usually done early in the system's development and supports the system's high-level design. Figure 7 helps us visualize the system architecture process and helps us contextualize what the system's "high-level" design means. The system architecting process starts with inputs from stakeholders that help define needs and well-refined requirements. System architecting helps capture the functional analysis that defines *what* the system needs to do and traces system functions back to requirements. Concepts that satisfy the system's functions are evaluated and compared. Each concept presents a more concrete form that defines *how* the concept accomplishes the system's required functions. Finally, there is a refinement of the system's form and a mapping of the system's components to the required functions. This is the "high-level" (preliminary) design, and it serves as the basis for the "design-to" specifications that are used for the detailed (engineering) design which produces the system's "build-to" or "buy-to" specifications.

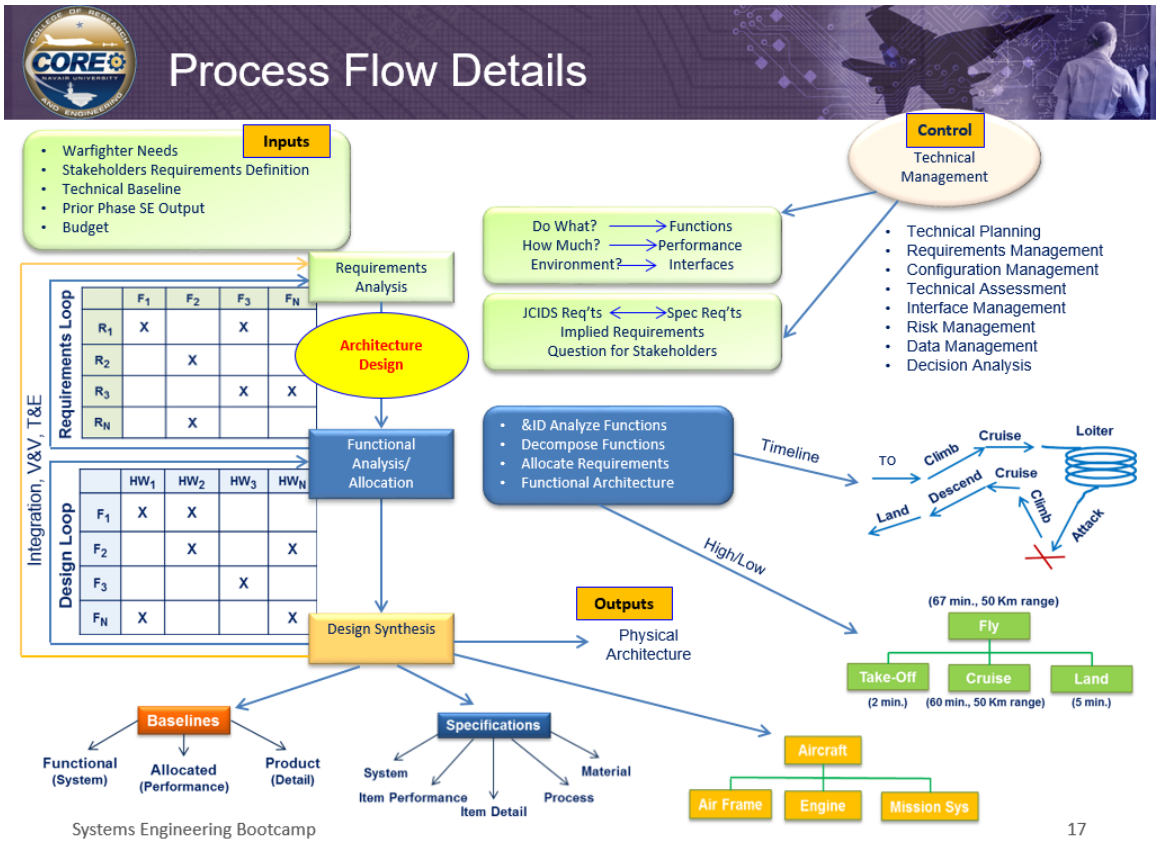


Figure 7. Systems architecting "Process Flow Details." Source: College of Research and Engineering (2019b, 17).

The detailed design of a system is not really considered to be part of system architecting; by this point most or all of the major elements of the system's design, which give the system is essential attributes and identity, have been largely settled. However, during detailed design, engineers can still define functions, elements of form and their form-function mapping—these are at the more concrete end of the system definition spectrum, so in a way they can continue system architecting activities. Figure 8. Shows the systems engineering "Vee." Systems architecting can be considered to be part of systems engineering; particularly the upper-left portion of the systems engineering "Vee," where the system architecture is developed and defined. Systems engineering is considered to be the fuller process of design that supports the system's life cycle, which includes the right part of the "Vee," and although not shown in this graphical model, systems engineering supports the system's operations and maintenance phase, which includes engineering

changes to the fielded system to ensure its utility up to the end of the system's service life, or in some cases modernizing and extending the system's service life.

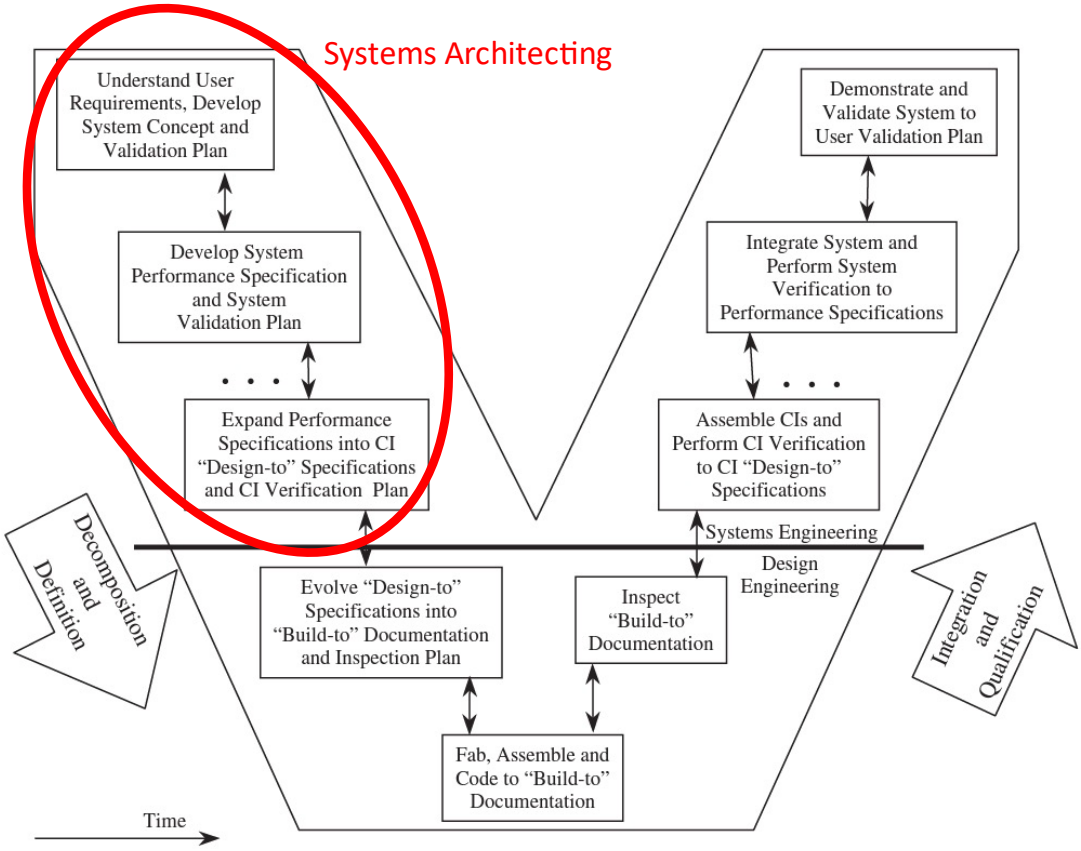


Figure 8. Systems architecting in relation to the Systems Engineering "Vee" Adapted from Buede and Miller (2016, 10).

Why think of the system from the perspective of its architecture? Thinking of the system architecture lets you address the problem domain in a solution-neutral way by first defining the solution-neutral functions of *what* the system solution must do to address the stakeholder needs, not *how* the system solution must meet the needs. Starting with a more solution-specific approach can add design constraints early on and can cause problems later if designers evaluate that the chosen solution-specific concept insufficiently address all the stakeholder needs or if the design is too limited, gives sub-optimal performance or is inflexible to future changes. Addressing the problem-domain in a solution-neutral manner

gives the designers more freedom to explore and evaluate a larger envelope of architectural concepts. Architectural concepts imply the form the system is going to take and are evaluated on their ability to address all of the known stakeholders' needs and design considerations while also determining the optimal performance trade-offs. Some architecture concepts may offer better design flexibility for addressing unknowns. Modular architectures are usually more flexible but can sacrifice performance. Once architectural concepts have been explored, designers may select the best suited conceptual architecture. The conceptual architecture usually implies a form, either abstractly or more concretely.

2. Systems Architecture Definitions

A few definitions of system architecture, quoted from various sources, are presented below for comparison.

Crawley et al. (2016):

> system architecture is the embodiment of concept, the allocation of physical/informational function to the elements of form, and the definition of relationships among the elements with the surrounding context. (Crawley, Cameron, and Selva 2016, 110).

Blanchard and Fabrycky (2011):

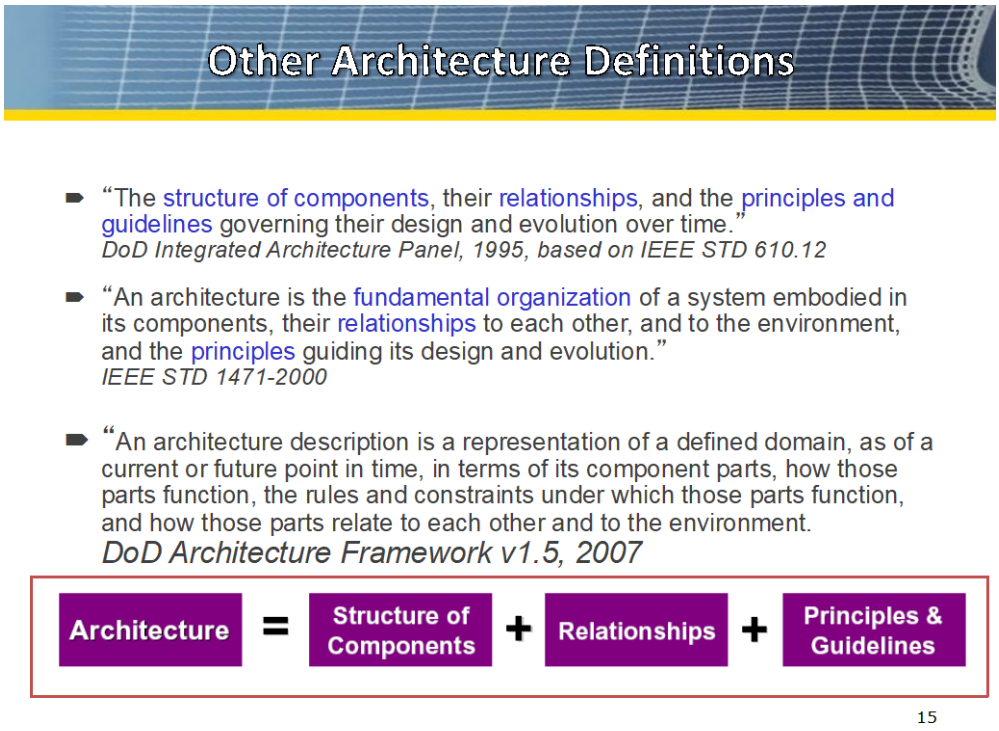
> The basic system architecture…deals with a top-level system structure (configuration), its operational interfaces, anticipated utilization profiles (mission scenarios), and the environment within which it is to operate….
>
> Architecture describes how various requirements for the system interact. This, in turn, leads into a description of the *functional architecture*, which evolves from a functional analysis and is a description of the system in functional terms. From this analysis, and though the requirements allocation process and the definition of the various resources requirements necessary for the system to accomplish its mission, the *physical architecture* is defined. (Blanchard and Fabrycky 2011, 93)

Ulrich and Eppinger (2012), who present the point of view, not from a systems engineering perspective, but from more of an industrial product design and development point of view:

> The physical elements of a product are typically organized into several major physical building blocks, which we call *chunks*. Each chunk is then made up of a collection of components that implement the functions of the

> product. The *architecture* of a product is the scheme by which the functional elements of the product are arranged into physical chunks and by which the chunks interact. (Ulrich and Eppinger 2012, 185)

Ulrich and Eppinger (2012) describe architectural *elements of form* as "chunks." Other definitions for architecture are provided in Figure 9, which also provides a graphical definition that complements the textual definitions.



This figure provides additional textual system architecture definitions and a more graphical definition at the bottom of the image.

Figure 9. Architecture Definitions. Source: Rhoades (2022, 15).

3. Modular vs. Integral Architecture.

Ulrich and Eppinger (2012) provide simple, but effective descriptions of *modularity*. Ulrich and Eppinger's (2012) notions of modularity classify a product's architecture based on how much modularity is exhibited by the product (system). The classification of modularity provides a spectrum—on one end, a product may predominantly exhibit a *modular architecture* and at the opposing end, a product may

predominantly exhibit an *integral architecture.* Ulrich and Eppinger (2012, 185) provide the following descriptions for a modular architecture:

> A *modular architecture* has the following two properties:
>
> - Chunks implement one or a few functional elements in their entirety.
> - The interactions between chunks are well defined and are generally fundamental to the primary functions of the product.
>
> The most modular architecture is the one in which each functional element of the product is implemented by exactly one physical chunk and in which there are a few well0defined interactions between the chunks. Such a modular architecture allows a design change to be made to one chuck without requiring a change to other chunks for the product to function correctly. The chunks may also be designed quite independently of each other. (Ulrich and Eppinger 2012, 185)

When "forward" engineering or reverse engineering a system, the designer should mind the architecture's modularity because it can greatly impact the attributes of the system. One particular impact could be the architecture's sensitivity to change and adaptation. Ulrich and Eppinger (2012, 188) state that "modular chunks allow changes to be made to a few isolated functional elements of the product without necessarily affecting the design of other chunks." Ulrich and Eppinger (2012, 188) list some of the motives for change, "upgrades, add-ons, adaptations, wear, consumption, flexibility in use, [and] reuse."

Designers should also consider when they can consolidate functionalities within an architecture's elements. Ulrich and Eppinger (2012, 185) provide the following descriptions for an *integral architecture*:

> An *integral architecture* exhibits one or more of the following properties:
>
> - Functional elements of the product are implemented using more than one chunk.
> - A single chunk implements many functional elements.
> - The interactions between the chunks are ill defined and may be incidental to the primary functions of the products.
>
> A product embodying an integral architecture will often be designed with the highest possible performance in mind….Many functional elements may be combined into a few physical components to optimize certain dimensions

> of performance; however, modifications to any one particular component or feature may require extensive redesign of the product. (Ulrich and Eppinger 2012, 185–186)

Ulrich and Eppinger's (2012) chapter ten, "Product Architecture," is recommended as a reference that explains architecture in "layman's" terms. The explanations are relatable without needing an existing background in systems architecture or systems engineering. This chapter also goes into greater details about the types of modular architecture: "slot-modular," "bus-modular," and "sectional-modular" architecture(Ulrich and Eppinger 2012, 186–187). Other references such as Crawley et al. (2016), Buede (2009; 2016), Weilkiens et al. (2015; 2017), Kossiakoff et al. (2020), and Blanchard and Fabrycky (2011) provide more systems engineering/architecture centric definitions and concepts.

4. The Physical Architecture

a. Form = Entities of Form + Structure (Formal Relationships)

The physical architecture is presented first, before, the functional architecture; even though, in the sequence of design, the functional architecture should be defined and understood before defining the physical architecture. There is a maxim that was coined by architect Louis H. Sullivan in his 1896 essay, "The Tall Office Building Artistically Considered." In his essay, Sullivan wrote, "form ever follows function," or as many of us know and remember it today, "form follows function." The Merriam-Webster dictionary defines "maxim" as, "a general truth, fundamental principle, or rule of conduct." Sullivan's maxim emphasizes the importance of first defining the functional purpose of architecture. Presenting the physical architecture before the functional architecture may seem counter-intuitive, but form is a more concrete characteristic, it is easier to grasp, and may establish some insights that will support understanding the functional architecture section.

According to Crawley et al. (2016, 53), "form is about existence…form is what the system *is*." *Form* is the implementation of design concepts into real-life manifestations or instances. For design communication purposes, form can be abstractly described and organized by its decomposition and hierarchy (Crawley et al. 2016). Form is what enables functionality and delivers value (Crawley et al. 2016). Larger *entities* of form can be decomposed or reduced to *sub-entities* of form, also commonly referred to as "elements"

or "chunks" of form (Crawley et al. 2016, 55). Crawley et al. (2016) provides the following definition of form:

> Form is the physical or informational embodiment of a system that exists or has the potential for stable, unconditional existence, for some period of time, and is instrumental in the execution of function. Form includes the entities of form and the formal relationships among the entities. Form exists prior to the execution of function…Form is a product/system attribute. (Crawley, Cameron, and Selva 2016, 54)

Along with entities of form, another important concept of form is *structure*. Structure is the "relationship among the entities of form" (Crawley et al. 2016, 54). Crawley et al. (2016, 64) states that there are two main types of structural relationships, "*spatial/topological relationships"* and "*connections*." *Spatial relationships* "capture absolute or relative location or orientation," while *topology* "is about placement and suggests relationships such as "within," "contained in," "surrounded by," "overlapping," "adjacent," "touching," "outside of," or "encircling" (Crawley et al. 2016, 64). *Connectivity* answers the question "what is connected, linked, or joined to what?" (Crawley et al. 2016, 71). Designers (and probably manufacturers too) should ask themselves the following question, "*Is this spatial or topological relationship key to some important functional interaction or to the successful emergence of function and performance?*" (Crawley et al. 2016, 64). With respect to M&S, logical relationships, particularly those of *connectivity* among M&S entities may be of more importance while *spatial relationships* may be more suited for describing physical systems. Additionally, spatial/topological relationships are hard to formally express in SysML because the language does not unambiguously convey semantics about how elements are arranged in real life. For example, using a SysML internal block diagram (ibd), one can model the block part-properties such that they are arranged in a spatial manner that is relative to the real-life layout of the system being described. For example, an ibd of a car block could have four wheel parts at each of the four corners of the frame/body part of the car—this approach of arranging SysML elements to imply spatial/topological relationships of the system can in the best of cases be somewhat useful, but most likely, a folly if done extensively. Crawley et al. (2016, 63) provide the following definition of "formal relationships":

> Formal relationships, or structure, are the relationships between objects of form that have the potential for stable, unconditional existence for some duration of time and may be instrumental in the execution of functional interactions. (Crawley, Cameron, and Selva 2016, 63)

According to Crawley et al. (2016, 54), form "is a *system attribute*; it is an independent way in which to view or characterize the system," and form "is also part of the solution that the architect is proposing." The reader should note that with regards to software, Crawley et al. (2106 , 78) states that "the code (or pseudocode) is the form: It exists, it exists prior to the execution of function, and it is instrumental of function….when it is operated, this form is interpreted as an instruction that, when executed, leads to function."

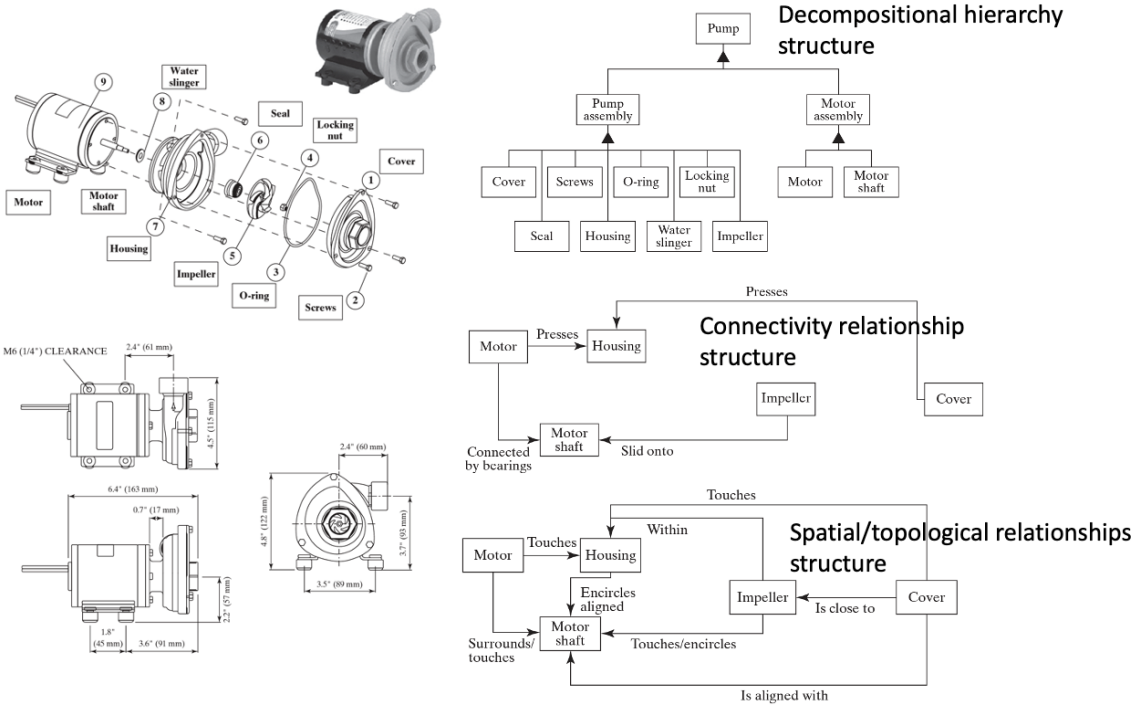


Figure 10. Examples of form: entities of form and their relationships. Drawings and diagrams convey information about how form exists in a system – some information is conveyed abstractly, other information more concretely. Adapted from Crawley et al. (2016).

b. The System Boundary and the System Context

When the system is decomposed into sub-systems, assemblies, components, and parts, the result is a definition and classification of separate distinguishable entities; at this point, *boundaries* become defined. During early stages of conceptual design, boundaries can also be conceptual, based arbitrarily on perspective or some frame-of-reference. Boundaries can help the make the system design more manageable; for example, responsibility for designing system sub-entities can be assigned based on the sub-entity boundary—which should also support the sub-entity interfaces. Conversely, an existing system may be *reversed-engineered* where a designer studies and analyzes the existing forms (and functions) of the system and then develops an abstraction of the system. The abstraction of the system defines the reverse-engineered, higher-level physical and functional (logical) architectures. This skill was employed as part of this thesis's research modeling work; understanding concepts of form and function were critical for developing the descriptive MBSE model.

The definition or demarcation of boundaries lets us distinguish the *system boundary* and the *system context*. As the name implies, the *system boundary* demarcates the limits of the *system-of-interest* (SoI) entity at the system level—which by association, also demarcates the limits of all of the system sub-elements. The *system interface* with external systems or the environment happens at the system boundary. The system interface or interfaces (plural) is a complex, abstract concept that is composed of all the (more concrete) system's sub-entity interfaces with other external systems or the environment—this sets apart the system interfaces from the system's *internal interfaces* which occur or exist inside the system boundary. The *system context* places the system-of-interest in context of its *environment* and other *external* or *accompanying systems* that the system-of-interest interacts with. Understanding the system context is important because it lets us understand the interactions of the system-of-interest with its environment and other external systems.

5. The Functional Architecture

It is advantageous to first consider the functionality at the system-level then look inwards into the system. A system is developed and realized to serve a purpose, i.e., the

needs of its stakeholders. The situation and the particular needs of the stakeholders can be translated and captured as the *problem domain* of interest. The problem domain should be well defined, with a lot of detail, and as complete as possible; ensuring that there are no critical omissions that would be detrimental to the designer's knowledge and ability to produce the *right system* that addresses the problem domain. Defining the problem domain provides the *problem definition*. It is crucial that both stakeholders and designers understand the problem definition which is directly associated to the stakeholder needs. This aspect of design and architecture is often overlooked. Defining the problem is rigorous and requires a lot of critical thinking and the necessary skills to successfully accomplish this task. Having a well-defined problem is a valuable asset because it allows the stakeholders and designers to conceptualize solutions, conditions, or outcomes that would resolve the problem domain and address the stakeholder needs.

Crawley et al. (2016, 137) presents two very useful architecture concepts for developing solutions: the *solution-neutral* function and the *concept*. Before defining these concepts, let us consider reverse engineering and its inverse, "forward engineering." In reverse engineering, the form (what the system is) is known and the task is to understand how the system works (its functions). In "forward engineering," designers take the philosophical approach that they must *first* specify *what* the system must do, based on the problem definition, and not specify *how* the system will exist or what form it will take. Many designers and engineers will believe that they understand everything they need to know about the system and will jump to conclusions by immediately specifying the components of the system—these components have a very concrete determination of the functions that will be enabled by their respecting components; only progress will tell if the design is suitable.

a. The solution-neutral function

When stakeholders and designers seek to define solutions or conditions that will resolve the problem domain and the stakeholder needs, they should first seek to define *solution-neutral* or implementation-neutral (agnostic) functionality that helps specify *what* the system must do. Crawley et al. (2016, 139) defines the *solution-neutral function* as "the

function of a system stated without reference to how the function is achieved." Defining a solution-neutral function usually involves abstracting or generalizing the function into its most essential concept or identity. For example, consider the following scenario:

- A first responder emergency team arrives at the incident scene and must let their remotely located incident commander know what the situation status is—this is one of the mission operational concepts that support the overall mission. In this scenario, the solution-neutral function that a designer may define is "communicate," stated as an action-verb. To be more elaborate, the relation may be described as follows: "*The first responder emergency team must "communicate" the situation status to the incident commander*." This definition does not specify *how* the team must communicate with its commander. Imagine the following:
    - Mission environment context: an area devastated by a hurricane with no functioning cell-towers.
    - External/accompanying system context: other friendly entities such as other response teams (search-and-rescue, police, federal agencies, etc.) and other existing infrastructure, such as the Iridium or Starlink satellite constellations. Another operational concept may be communication *interoperability* between friendly entities, but we will not consider this one.
    - Mission task parameters: the mission objective is for the team to provide the situation status to their commander within 30 minutes of arrival at the scene with an 80% success rate of accomplishing this task. If the team is not able to communicate withing the first 30 minutes, the team must communicate within the next 3 hours with a 95% likelihood of success. The mission threshold for this operational concept is for the team to communicate the situation status to their commander at least once a day, regardless of the time.

- NOTE: This is an informal way to define *System Operational Requirements.* For a more formal approach, see Blanchard and Fabrycky's (2011, 61) section on "Defining System Operation Requirements," 1) *Mission definition*, 2) *Performance and physical parameters*, 3) *Operational deployment or distribution,* 4) *Operational life cycle (horizon)*, 5) *Utilization requirements*, 6) *Effectiveness factors*, and 7) *Environmental factors*.

- At this point, engineers may start conceptualizing alternative solutions that define *how* this task can be satisfied. Cell-phones will not work because the cell-towers are down. Very High Frequency/Ultra High Frequency (VHF/UHF) radios may work if the commander is within range. High Frequency (HF) radios have beyond-line-of sight/over-the-horizon range but require user skill and transmissions are dependent upon environmental factors—some say it is an art to establish communication via HF. Satellite communication devices, such as a satellite phone or some other satellite network terminal will likely work well in this situation, but service subscriptions and user rates can be expensive.

- Depending on the mission parameters, available design choices and their attributes (performance, cost, serviceability, etc.) the designer may pick the most effective solution or set of diverse solutions—if redundancy improves the reliability of achieving mission success. In any case, the solution-neutral function, "communicate," provides a broad and flexible domain for exploring solution concepts.

System architects who practice MBSE or are familiar with *use-cases* may define concise *use-cases* of the system that satisfy stakeholders' needs. One or more stakeholder needs are satisfied when a use-case is completely executed by the system. The full set of use-cases should cover all of the defined stakeholder needs. If a *materiel* solution is deemed necessary, as opposed to a *non-materiel* solution, the stakeholders and designers must identify *what* the system must do and express it through defining *solution-neutral*

functionality. For context, a *non-materiel* solution is defined as "a resolution to an identified need that doesn't require a material solution or a system to be developed" (AcqNotes 2017b). A non-materiel solution can be developed from Doctrine, Organization, Training, Leadership and Education, Personnel, Facilities and Policy (DOTMLPF-P) analysis (AcqNotes 2017b). A *materiel* solution is a solution that is satisfied by hardware *and* software products. Designers, acting in the capacity of a system architect, must perform a functional analysis, usually from the top-down. Figure 11 depicts a top-down (static) functional decomposition which is developed from performing a functional analysis at each system hierarchy level.

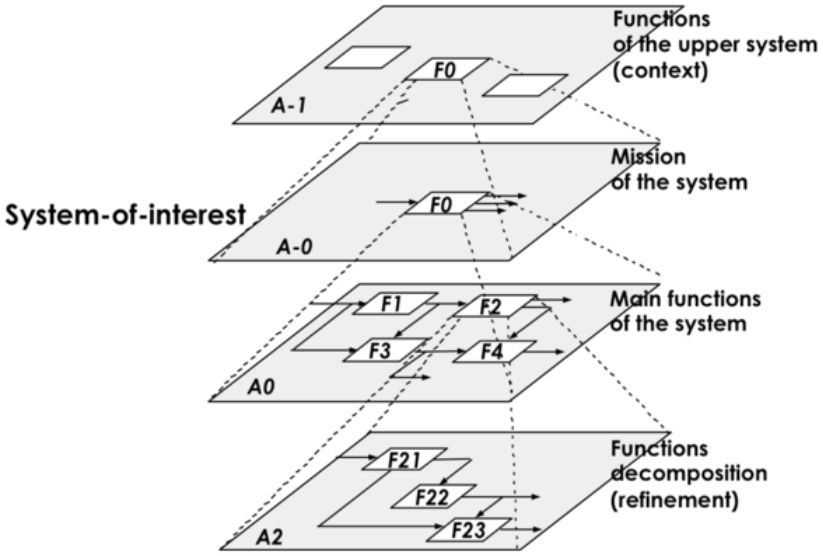


Figure 11. Functional hierarchy and decomposition of functions. Source: Faisandier and Roedler (2023).

Another example of a functional analysis depicting a system's functional decomposition hierarchy is presented in Figure 12.

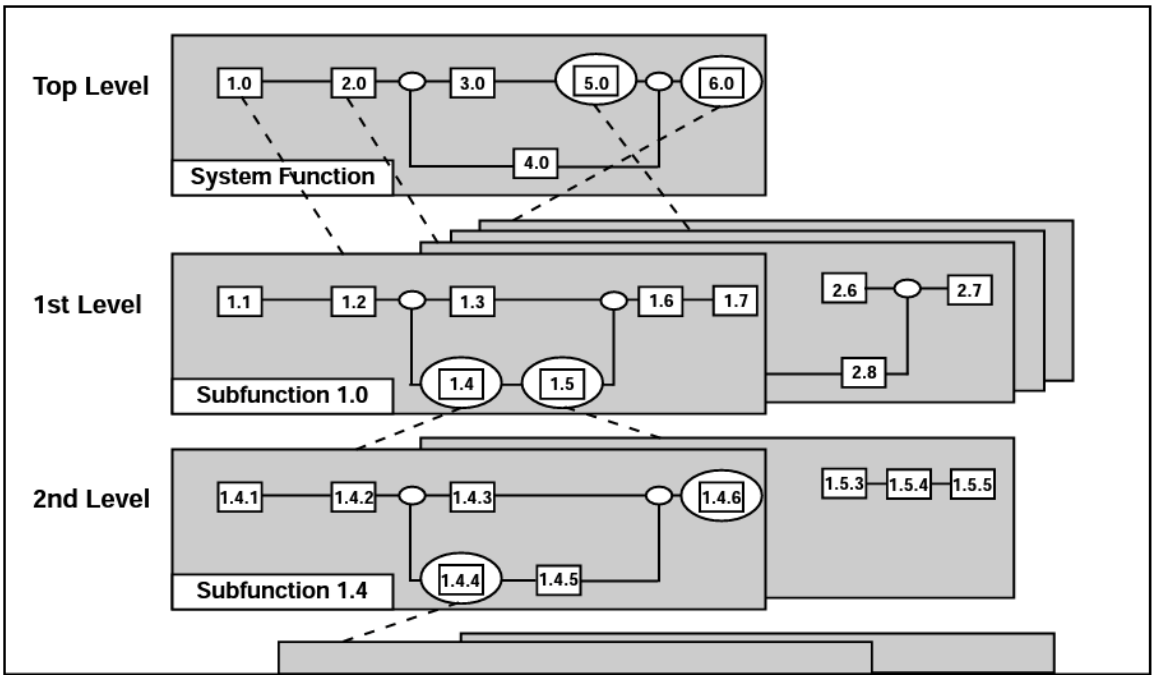


Figure 12. Functional analysis and decomposition, functional flow block diagram (FFBD) traceability and indenture. Source: Systems Management College (2001, 49).

b. The (architecture) concept

The problem definition and the system context inform the solution-neutral functionality; helping designers conceptualize the system's interactions and parameters. The next step of architecting is to identify architecture *concepts*. Crawley et al. (2016, 142) provides the following descriptions and definition of a *concept*:

- The intellectual distance from the solution-neutral function to the architecture of the system is a large gap to jump. In order to help, we create an intellectual construct known as a *concept*.
- [A] *concept* is the high-level mapping of function to form…[the] concept is a way to conveniently and concisely convey the vision of the system.

Additionally, Crawley et al. (2016, 142) states that the "concept":

- Is the transition point from the solution-neutral to the solution-specific.
- Must allow for value-related functions to be executed, enabled by form.
- Establishes the vocabulary for the solution and is the beginning of the development of architecture.
- Implicitly sets the design parameters of the system.

- Implicitly sets the level of technology.

c. The functional architecture

After concepts have been defined, the solution-neutral functional analysis is mature and mostly settled and so is the functional analysis of the solution concepts. The functional analysis describes the *behavior* of the system at its various hierarchy levels. Before defining system *behavior*, let us define a *function*, which is the building block of emergent behavior. Crawley et al. (2016, 109) defines a function as the "actions, activities, operations, and [state] transformations that cause or contribute to performance." Crawley et al. (2016, 127) then defines *behavior* as "the sequence of functions (and associated changes in state) that form executes in order to deliver value and it is a product attribute." The key idea behind the notion of behavior is the "sequence of functions" (Crawley et al. 2016, 127) or *logical* functional relationships. *Behavior* lets us think about the structure of functions and supports the definition of the system's *functional architecture*. Crawley et al. (2016, 94) state that:

> Within a system, there are internal functions and relationships among these internal functions. Together these define the *functional architecture* of the system. The *externally* [emphasis added] delivered function, performance, and "ilities" emerge from these internal functions and relationships.

d. The (functional) value pathway

Crawley et al. (2016) present mind enriching concepts of how or when *value* is delivered by a system. Ultimately, Crawley et al. (2016, 89) state that "function and value are always delivered at an interface in the system boundary." This is a unique concept that sets apart and elevates the functional interactions and inputs/outputs that occur at the system boundary. This is very much a *black-box* perspective of the system and conveys the importance of paying special attention to the system interfaces because this is where the system delivers something of value to its users, external/accompanying systems, and the environment. From a white box perspective, it is understood that there is a logical sequence of functional interactions that happen "inside" the system that lead up to the delivery of functional outputs or outcomes at the system boundary, which ultimately deliver value or achieve the purpose (or one of the purposes) of the system. This leads to the concept of the

value pathway. Crawley et al. (2016, 102) states that the "value pathway" is "one of the main features of the functional architecture," and is the logical functional pathway "along which value develops." The value pathway usually starts with an input to the system, "moves" or is transformed along the way by the system's internal functions and is delivered as a "value-related" (Crawley et al. 2016, 102) output. For more information on the *value-pathway* and "secondary value-related" functions (not covered here), see Crawley et al. (2016, 102–109)

e. Discussion on the logical architecture

The *logical architecture* is largely synonymous with the functional architecture. Faisandier and Roedler (2023) provide the definition of the *logical architecture* in the *Systems Engineering Book of Knowledge*, which currently identifies the International Council on Systems Engineering (INCOSE), the Institute of Electrical and Electronics Engineers (IEEE) Systems Council, and the Stevens Institute of Technology as its "stewards"; the definition of the logical architecture is as follows:

> Its [the logical architecture model] purpose is to elaborate models and views of the functionality and behavior of the future engineered system as it should operate while in service. The logical architecture model of a engineering system of interest (SoI) is composed of a set of related technical concepts and principles that support the logical operation of a system. It may include a functional architecture view, a behavioral architecture view, and a temporal architecture view. (Faisandier and Roedler 2023)

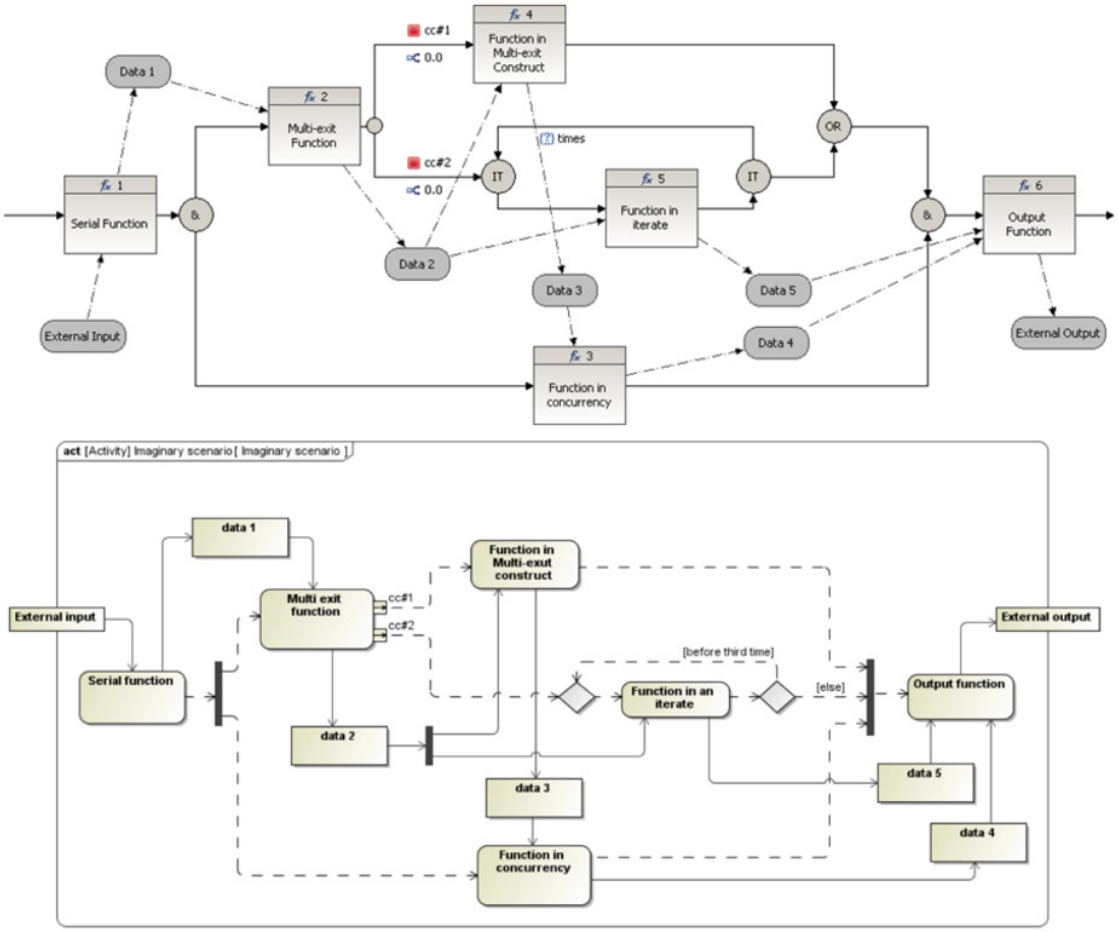
According to Faisandier and Roedler (2023), the *logical architecture* "may include a functional architecture view." In MBSE, a view is a visualization of the underlying model, so Faisandier and Roedler (2023) are implying that there is a model. Faisandier and Roedler (2023) continue to define the *functional architecture model,* stating that this model "is a set of functions and their sub-functions that defines the transformations performed by the system to complete its mission." Figure 11 provides an example of this model view. Readers should keep in mind that system architecture terminology and vocabulary is not standard, and the same or similar terms can convey slightly or somewhat different meanings. For example, Faisandier and Roedler (2023) define the *behavioral architecture model* as:

> An arrangement of functions and their sub-functions as well as interfaces (inputs and outputs) that defines the execution sequencing, or conditions for control or data-flow, and performance level necessary to satisfy the system requirements…[and] can be described as a set of inter-related scenarios of functions and/or operational models.

This definition of the behavioral architecture model is not significant enough to distinguish it from the functional architecture as it is described in this thesis document because the functional architecture encompasses the behavior that results from the logical sequence and structure of functions or actions that the system performs internally. The term "emergent behavior" or even just simply "behavior" is usually reserved for describing the observed behavior at the system-level context. The term "emergent" is used because sometimes the behavior is a mixed result of planned, intended, or controlled design functions and even unintended (often undesired), uncontrolled functions. The behavior that emerges is considered to be more than the sum of the individual functions. For example, the electromagnetic compatibility (EMC) discipline must consider the emergent or resultant effects of electromagnetic interference that affect overall system performance and must attempt to minimize undesired effects that are within the designer's control. Electromagnetic interference can manifest from the primary electromagnetic functions (intended emissions from an antenna) interfering with other system sensors or electronic equipment. The inverse effect can also happen when secondary or non-intended electromagnetic sources such as emissions from phone power chargers or passive intermodulation (PIM)—caused by system connectors, antennas, cables, loose connectors, rusty bolts, nearby metallic objects such as fences, etc.—cause interference with the primary electromagnetic functions.

Perhaps the behavior definition provided by Faisandier and Roedler (2023) is distinguished because there are models or diagrams that represent the functionality of a system as a tree-like taxonomy which classifies all of the systems main functions. This abstraction is helpful for understanding the functional set that the system is capable of, but it conveys little to no insight on *how* the system functions are implemented in an operational scenario. Thus, a functional taxonomy only provides a limited view of the functional architecture. A simple example of a functional taxonomy is represented on

Figure 7, where the main function, “Fly,” is decomposed to the following sub-functions, “Take-Off,” “Cruise,” and “Land.” It is not clear if the Faisandier and Roedler’s (2023) definition of a functional architecture also had a functional taxonomy in mind. Figure 13 presents Faisandier and Roedler’s (2023) idea of a model view that describes functional behavior.



The top view is in the form of an extended functional flow block diagram (eFFBD), while the bottom depicts the same subject described in SysML. Adapted from Faisandier and Roedler (2023).

Figure 13. Behavior architecture model view.

6. The Allocated Architecture

The *allocated architecture* is simply the mapping of form to function, or as Kossiakoff et al. (2020, 238) define it, "this [the allocated architecture] is a simple accounting of which functions are performed by which components within the system." This supports overall traceability across the system elements and leads back to the originating stakeholder needs and requirements. The allocated architecture helps the designer analyze the allocation or mapping of form to function and ensures that there are no functions without an element of form (that enables them), or any elements of form that haven't been assigned any particular function, which raises the question if they are needed or not. Stated differently, "the main purpose of the allocated architecture is to ensure that no function is left behind so that a component in the physical architecture must satisfy the function," and conversely, "the allocated architecture seeks to ensure that each component is gainfully employed and has some function to perform" (Kossiakoff et al. 2020, 239).

One important consideration of the allocation phase is the trade-off between performance attributes. For example, efficiency of functions vs. reliability. Kossiakoff et al. (2020, 239) mentions that complex systems "would have multi-function components, where one component may perform more than one function," and that caution should be taken because "if the component fails that may cause many functions to be unable to perform." Alternatively, "another approach is multiple components completing the same function – in this case, it may either a cooperative effort or a redundancy of components (e.g., spares ready to perform the function in the case that the leading component fails)" (Kossiakoff et al. 2020, 239).

Buede and Miller (2016) present a simple graphic depicting the origins of the various system architectures, shown in Figure 14.

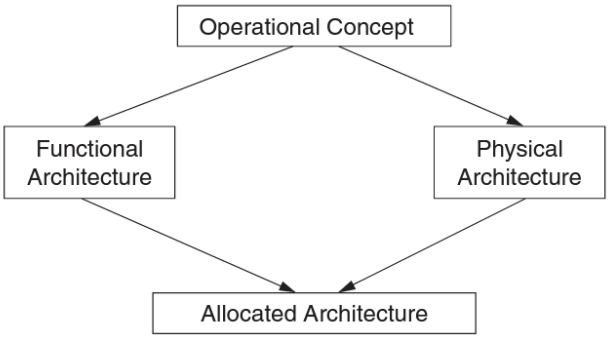


Figure 14. System architecture development. Source: Buede and Miller (2016, 27).

This graphic does not convey the natural or realistic development process of these architectures. The functional and physical architectures are not created separate from each other, but as Buede and Miller (2016, 27) note, "the two [functional and physical] architectures are developed in parallel but with close interaction to ensure that the allocated architecture is meaningful when the functional and physical architectures are combined [mapped or related to each other]." The functional architecture may be developed first, and the physical architecture may be developed shortly after, but good modeling practice should compel the designer to start allocating or mapping the elements of form to the respective functions they enable as soon as these details are known.

An example of an allocated view is shown in Figure 15. This view only represents a portion of the system's full allocated architecture. The system architecture in this example is captured as a SysML model; what is shown is merely a view of the underlying model. The diagram on the left shows an activity diagram representing the "Feel comfortable temperature" behavior (in reference to a car's air conditioning and heating sub-system); the functions have not been allocated yet. The diagram on the right shows allocation by incorporating *swimlanes* that represent the element of form that performs the functions inside the swimlane. Functions are allocated to elements of form by placing the action inside the element's swimlane. In this case the "vehicle occupant" and the "climate control unit" are the elements of form that perform the actions allocated to them inside their swimlane.

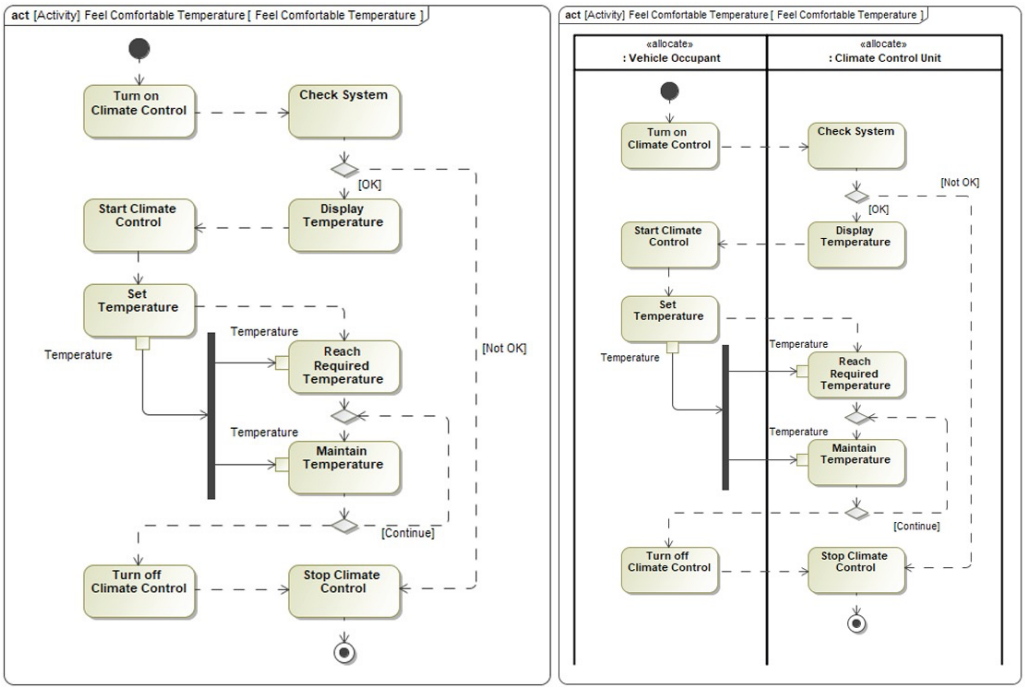


Figure 15. Functional allocation example. Adapted from Aleksandravicience and Morkevicius (2018).

The information provided in an allocated view, such as an activity diagram with swimlanes, can be represented in a table or matrix format. An example of how an allocated architecture can be represented using a table or matrix form is shown on Figure 16.

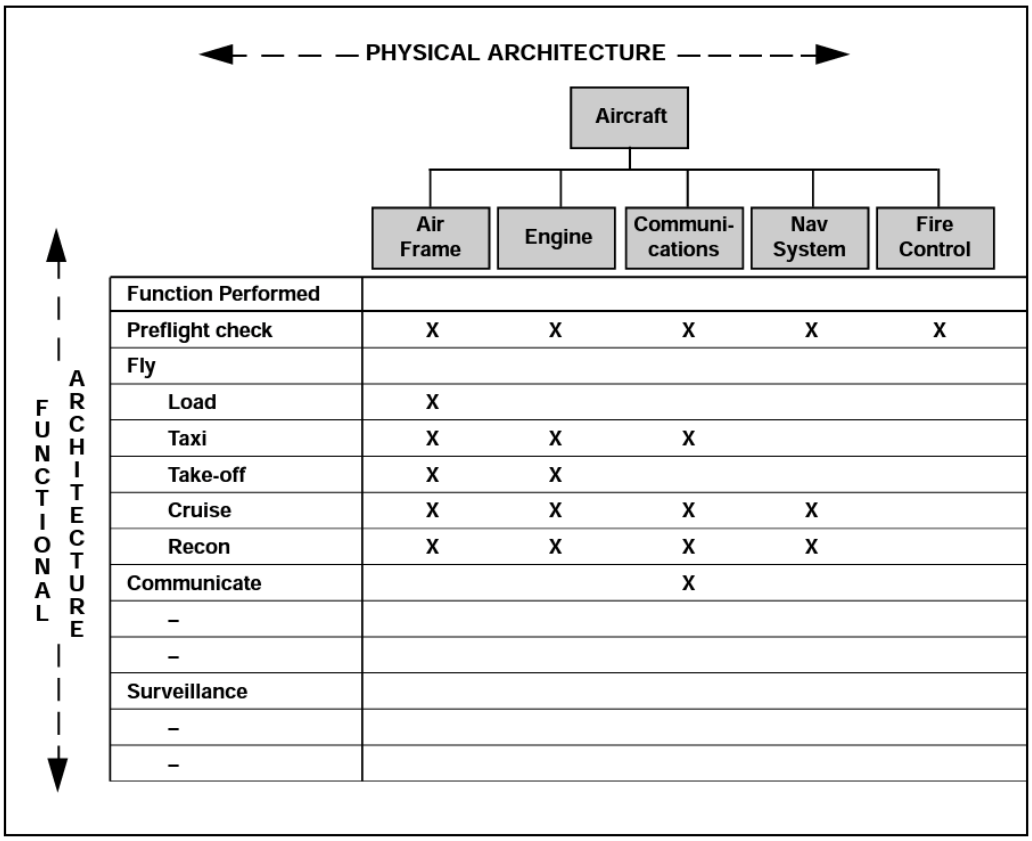


Figure 16. Functional to physical allocation matrix. Source: Systems Management College (2001, 59).

7. Architecture Presentation Frameworks

A *presentation* or *architecture framework* is an *ontological* way to classify and structure information about the domain of interest through a set of framework-defined views. The views capture information about the system or domain of interest according to the framework guidelines or specifications, then the framework classifies the views according to their type or category and structures the views according to how the views relate to each other. The framework serves as the underlying mapping for how information is captured, or how new information can be captured according to the framework. A framework can help standardize the type of information that is captured or how it is captured and can help maintain overall information consistency. Views usually target information of sub-domains of interest and show specific domain entities and specific (logical) relationships of those entities within the view. Sometimes views can capture

information that is exclusive only to that view, such that only that view presents the information of that sub-domain; other views can capture information that is intersecting, capturing information across more than one sub-domain, meaning that information in one view, may appear in one or more other views.

The views are important because they are the primary mechanism through which users: (1) encode new information into the framework, and (2) consume information from the framework. The view provides information through a *visualization* that conveys the information graphically the viewer who then processes the semantics of the graphical elements and synthesizes the overall significance of that view. Some "views" are not graphical views at all and exist in other forms such as tables/matrices, or text. Nonetheless, views target specific information about some sub-domain of interest that is part of the overall domain of interest which is captured, structured, and organized by the framework.

This research studied and considered borrowing specific views from various frameworks that could be useful based on the type of information that the view captures. Two primary frameworks were considered, the *Department of Defense Architecture Framework* (DoDAF) version 2.02 (release approved August 2010), and the *Zachman Framework*. The Zachman and DoDAF frameworks were evaluated for use in this thesis's research (not presented in this document). Specifically, the views of the framework were evaluated for the information they capture or represent that can support M&S VV&A domain related information. Primarily, a framework view would be considered for implementation as part of this research methodology if the view: (1) is not covered by the MagicGrid framework, and (2) if the view is deemed useful by capturing important information that would support M&S VV&A.

The author considers the MagicGrid framework to also be an *architectural framework*. The Magic Grid framework focuses primarily on the system's architecture, while DoDAF and the Zachman Framework include information that extends beyond the system and convers other information about the organization and business context layers surrounding the system.

a. Ontology Summary

Originally, *ontology* is a subject that has a long-established history in philosophy; it is different than *epistemology*, which is about knowledge and knowing. The Merriam-Webster dictionary defines ontology as, "a branch of metaphysics concerned with the nature and relations of being," or "a particular theory about the nature of being or the kinds of things that have existence." Scientists and engineers have specialized the philosophical concepts of ontology and applied these concepts to their particular areas of research or practice. With respect to information and computer science, Tom Gruber proposed the following definition of an ontology in 1992, "an ontology is a specification of a conceptualization" (Stanford University n.d.).

Figure 17 provides a very simple yet concise representation of how an ontology (the "specification" of a conceptualization) is derived from a "real world" domain of interest. Scientists and engineers who perform *ontology engineering* perform very similar tasks as engineers who perform *systems architecting*. Both try to define entities and their relationships in an unambiguous and structured manner. However, an ontology has another primary objective, which is to classify individuals or instances according to their abstract hierarchical relationships and identify other relationships that exist between these individuals or even their more abstract classes.

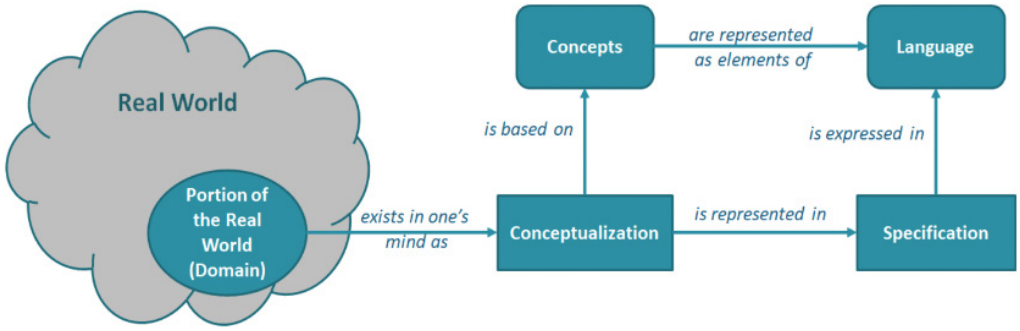


Figure 17. Ontology (conceptual map) of an ontology specification. Source: Cambridge Semantics (n.d.).

Engineers may be more familiar with the term *concept map*. A concept map represents an underlying ontology, similarly to how an MBSE diagram represents an underlying model. Today, ontology engineers use ontology languages such as the Ontology Web Language (OWL) and ontology editing (modeling) tools like Protégé in a similar manner that a systems architect or engineer would us a modeling language such as SysML and an MBSE modeling tool such as Cameo Systems Modeler. Figure 18 is an example of an ontological view, or as engineers may know it, a conceptual map.

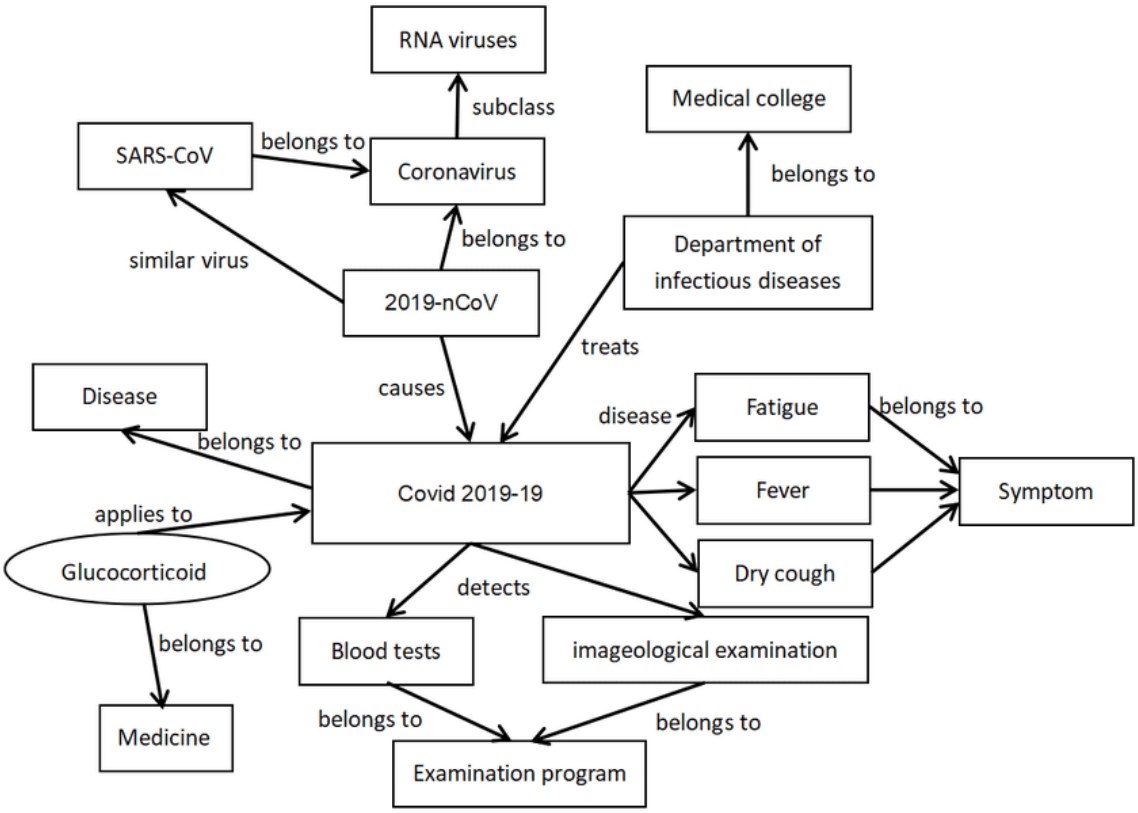


Figure 18. Example of an ontology view. Source: Xue et al. (2022).

b. The Zachman Framework

Concepts about the Zachman Framework are presented first because these concepts also apply to other architecture frameworks even though the concepts may not be stated explicitly or claimed by other architecture framework sources. Specifically, the concepts

in mind are (1) that an architecture framework *is* a form of *ontology*, (2) an architecture framework is more of a structure than a process, (3) the architecture framework is the basis of capturing or defining an architecture, and (4) an architecture framework provides a *schema* or a *metamodel* about how architecture information is classified and structured within the context provided by framework domain.

According to Zachman (n.d.), The Zachman Framework is a "schema," deriving from:

> the intersection between two historical classifications that have been in use for literally thousands of years. The first fundamentals of communication found in the primitive interrogatives: *What*, *How*, *When*, *Who*, *Where*, and *Why*. It is the integration of answers to these questions that enables the comprehensive, composite description of complex ideas. The second is derived from reification, the transformation of an abstract idea into an instantiation that was initially postulated by ancient Greek philosophers and is labeled in the *Zachman Framework*: *Identification*, *Definition*, *Representation*, *Specification*, *Configuration*, and *Instantiation*….
>
> More specifically, the Zachman Framework is an ontology – a theory of the existence of a structured set of essential components of an object for which explicit expressions is necessary and perhaps even mandatory for creating, operating, and changing the object (the object being an Enterprise, a department, a value chain, a 'sliver,' a solution, a project, an airplane, a building, a product, a profession or whatever or whatever)….
>
> The Zachman Framework is NOT a methodology for creating the implementation (an instantiation) of the object. The *Framework* IS the ontology for describing the Enterprise. The *Framework* (ontology) is a STRUCTURE whereas methodology is a PROCESS. A Structure is NOT a Process. A Structure establishes definition whereas a Process provides Transformation. (Zachman n.d.)

Zachman (n.d.) states that "the Zachman Framework is a metamodel," and is "unlike a methodology," because the framework does not "imply anything" about how capturing the framework is implemented. Zachman (n.d.) lists out five implications of a methodology that are *not* implied by the Zachman Framework. Finally, Zachman (n.d.) claims that the Zachman Framework is the "basis for architecture." However, Zachman's (n.d.) predominant focus is how the framework applies to define "enterprise architectures," because modern enterprises are "increasing in complexity" and "changing." Zachman (n.d.) claims that the enterprise architecture "is the determinant of survival." Zachman

(n.d.) is strongly implying that the architecture of an enterprise provides an inherent attribute of the enterprise and that this attribute supports its competitive fitness and ability to maintain a purpose or satisfy needs that feed/support the enterprise's existence.

Figure 19 provides a visualization graphic that depicts the Zachman Framework. The columns contain the five primitive interrogatives: *What*, *How*, *When*, *Who*, *Where*, and *Why*, respectively corresponding to *Inventory*, *Process*, *Distribution*, *Responsibility*, *Timing*, and *Motivation*. The rows contain Zachman's concept reification (abstract to concrete) devices or processes: *Identification*, *Definition*, *Representation*, *Specification*, *Configuration*, and *Instantiation*—each of these reifying processes are assigned to a specialized entity of the enterprise, for example, *representation* corresponds to the architect perspective, and *specification* to the engineering perspective.

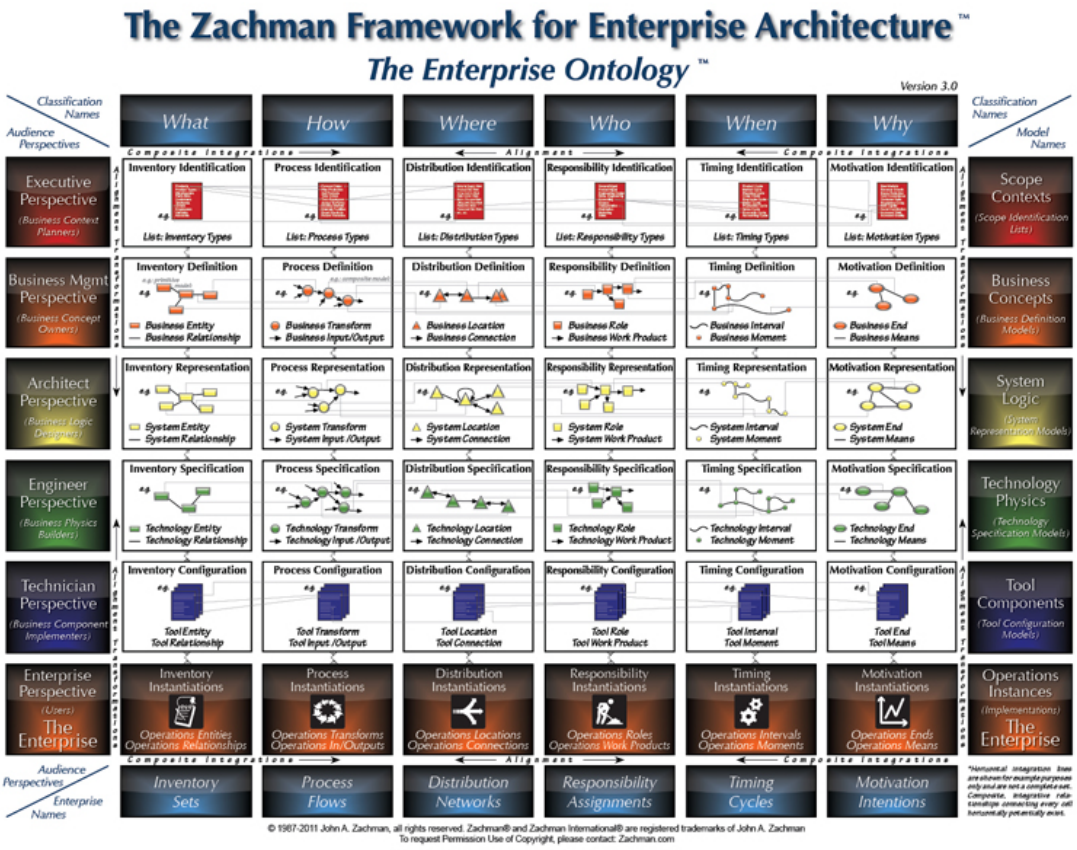


Figure 19. The Zachman Framework for Enterprise Architecture The Enterprise Ontology. Source: Zachman (n.d.).

c. The Department of Defense Architecture Framework

The Department of Defense Architecture Framework was studied and evaluated because of the familiarity that DOD professionals should have with this framework; making it a common framework that DOD stakeholders should recognize and understand.

The Chief Information Officer (Chief Information Officer n.d.a) of the U.S. Department of Defense maintains the website with the latest DoDAF standard (currently version 2.02). Only a brief introduction of DoDAF is presented here. The following are quotes from the Chief Information Officer's DoDAF website that summarize what DoDAF is.

> The Department of Defense Architecture Framework (DoDAF), Version 2.0 is the overarching, comprehensive framework and conceptual model enabling the development of architectures to facilitate the ability of Department of Defense (DOD) managers at all levels to make key decisions more effectively through organized information sharing across the Department, Joint Capability Areas (JCAs), Mission, Component, and Program boundaries. (Chief Information Officer n.d.a)

> DoDAF has been designed to meet the specific business and operational needs of the DOD. It defines a way of representing an enterprise architecture that enables stakeholders to focus on specific areas of interests in the enterprise, while retaining sight of the big picture. To assist decision-makers, DoDAF provides the means of abstracting essential information from the underlying complexity and presenting it in a way that maintains coherence and consistency. One of the principal objectives is to present this information in a way that is understandable to the many stakeholder communities involved in developing, delivering, and sustaining capabilities in support of the stakeholder's mission. It does so by dividing the problem space into manageable pieces, according to the stakeholder's viewpoint, further defined as DoDAF-described Models. (Chief Information Officer n.d.b)

> DoDAF is prescribed for the use and development of Architectural Descriptions in the Department. Specific DoDAF-described Models for a particular purpose are prescribed by process-owners. All the DoDAF-described Models do not have to be created. DoDAF V2.0 is "Fit-for-Purpose," based on the decision-maker needs. DoDAF does not prescribe any particular Views, but instead concentrates on data as the necessary ingredient for architecture development. However, other regulations and instructions from both DOD and CJCS may have particular presentation view requirements. These Views are supported by DoDAF 2.0, and should

be consulted for specific view requirements. (Chief Information Officer n.d.b)

Figure 20 shows the various DoDAF viewpoint categories: *All Viewpoint*, *Capability Viewpoint*, *Data and Information Viewpoint*, *Operational Viewpoint*, *Project Viewpoint*, *Services Viewpoint*, and *Systems Viewpoint*. Descriptions and information are available from the DoDAF website (Chief Information Officer n.d.b).

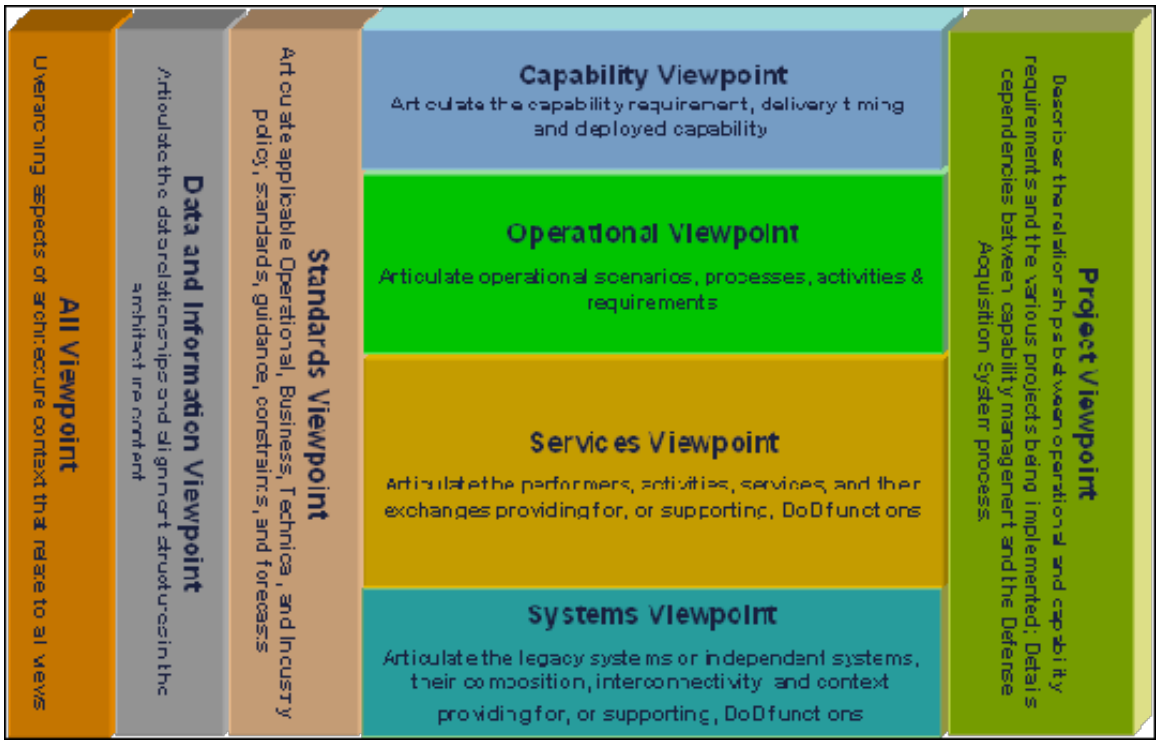


Figure 20. DoDAF Viewpoints. Source: Chief Information Officer (n.d.b).

Data and information *visualization* or *presentation* methods are important because they provide an efficient way for conveying architectural data to stakeholders. Figure 21 roughly illustrates how "presentation techniques pull from the architectural information store and display the data in a variety of meaningful ways to stakeholders." (Chief Information Officer n.d.c)

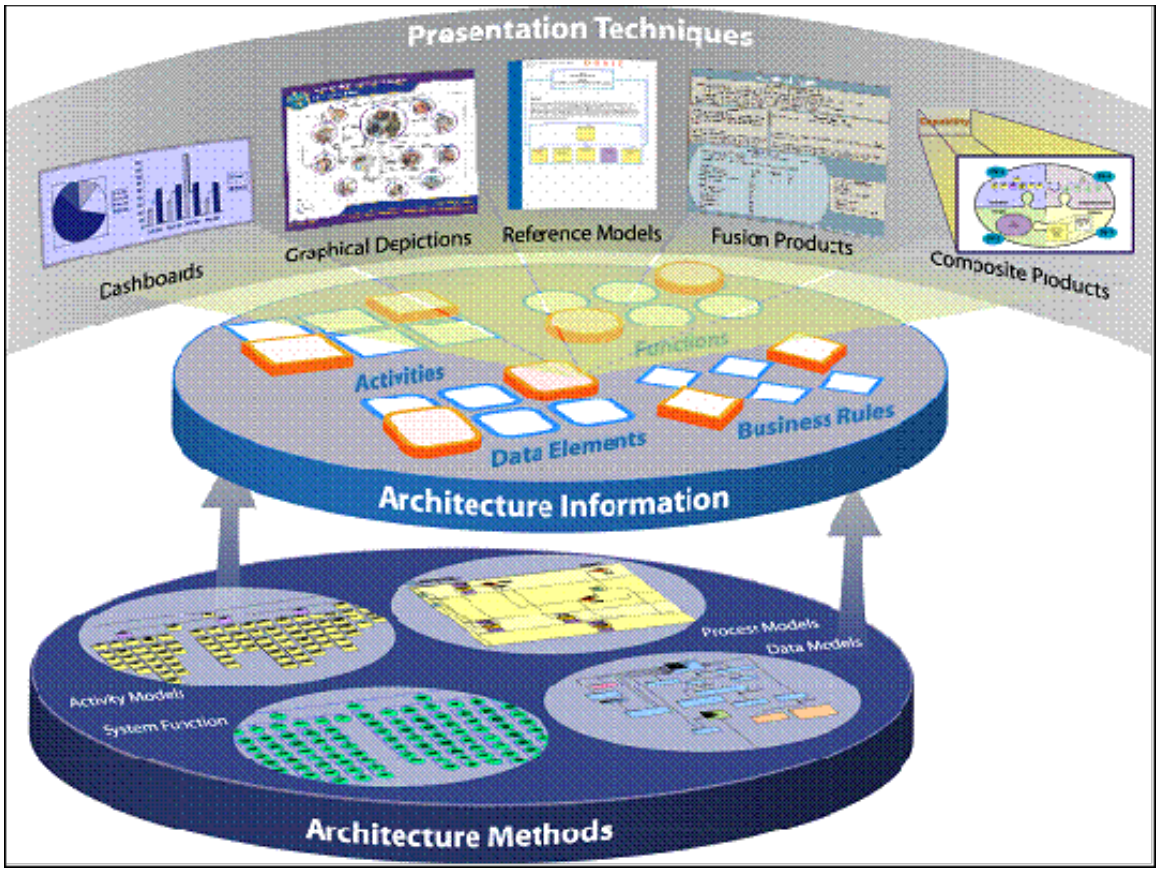


Figure 21. Presentation techniques for architecture visualizations. Source: Chief Information Officer (n.d.c).

It may be of interest to the reader to see how DoDAF views relate to the systems engineering process. Figure 22 organizes DoDAF views according to their categories and shows how of the main (high-level) systems engineering processes map to the corresponding DoDAF views. This graphic is rather abstract and doesn't provide a lot of definition, but it gives a rough idea of how these two subjects relate.

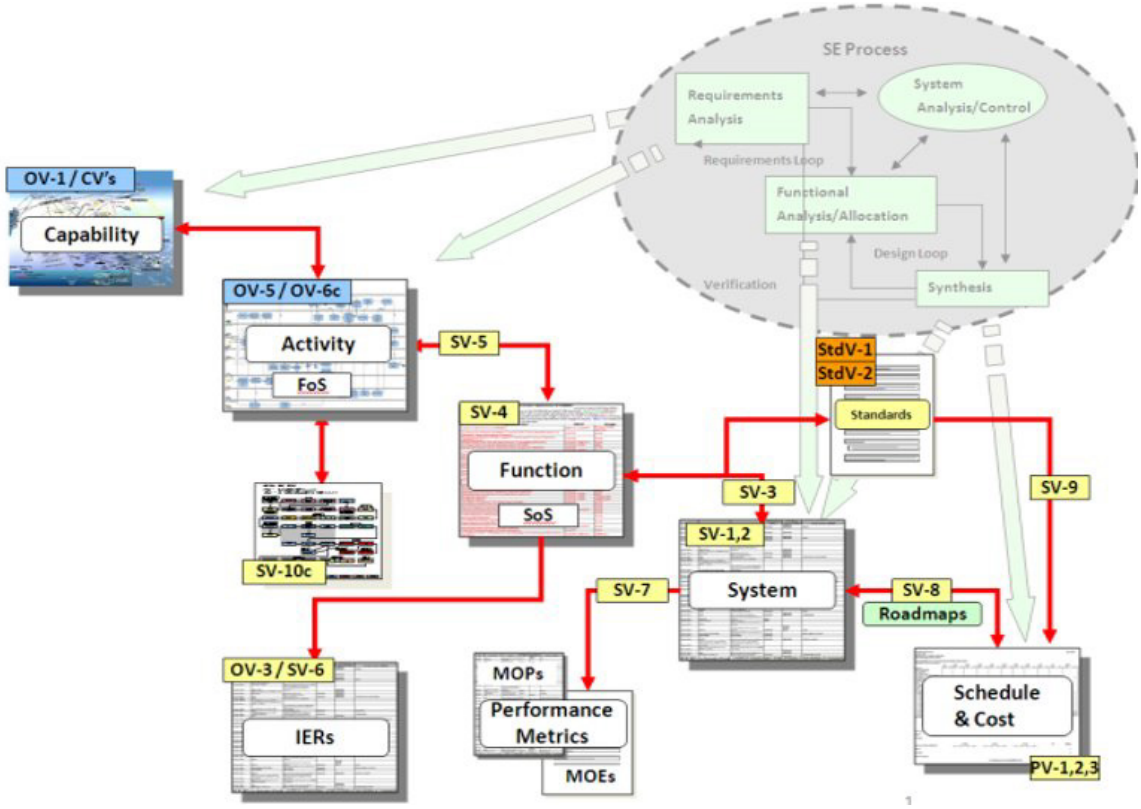


Figure 22. Mapping of architecture views with respect to the systems engineering process. Source: College of Research and Engineering (2019a, 20).

C. SYSTEMS ENGINEERING

This section of the literature review may be the one that most readers are familiar with. Therefore, not a whole lot of detail will be spent in this section; and only a minority of the systems engineering domain will be covered. This section primarily expands only the systems engineering topics that were deemed useful for this thesis research. Additionally, some of these topics should be useful in the case that someone wants to design and build an M&S product by using a systems engineering approach. Why use the systems engineering approach and not just the scientific method approach? The systems engineering approach may provide a fuller set of considerations and methods that can help manage complexities of an M&S product better than the scientific method. The systems engineering method is also about control. Designers are trying to control or force a man-made or engineered outcome, which is to deliver a system product that meets the

requirements and needs of stakeholders; each main phase of the systems engineering process attempts to mature, manage, and control the form of the design until it is ultimately realized. The scientific method is more about knowledge discovery, testing a hypothesis or theories, or further understanding and characterizing natural phenomena. Both are done in logical and systematic fashion, but their end goals and products are different. An M&S product is an engineered product; once realized, the M&S product can support scientific experiments.

1. Systems Engineering Definitions

It is hard to reduce the description of systems engineering to a single definition. Systems engineering is best appreciated and understood by its practitioners. It is important to not only understand the propositional definition of systems engineering, but to also understand the nature and discipline of the systems engineer. The following definitions are provided by INCOSE (n.d.) and Buede and Miller (2016), respectively:

What is systems engineering?

> Systems Engineering is an interdisciplinary and integrative approach to enable the successful realization, use, and retirement of engineered systems, using systems principles and concepts, and scientific, technological, and management methods. (INCOSE n.d.)

What are systems engineers and what do they do?

> Systems engineers must be 'big picture' people. Depth is only achieved by much iteration through the design process, as many as are needed until the system's specifications are sufficiently detailed for individual configuration items (CIs) to be built or purchased. This design process defines what the system must do, how well the system must do it, and how the system should be tested to verify and validate the system's performance. To do this the systems engineers must maintain a very clear focus on the objectives that the system's stakeholders (users, owners, manufacturers, maintainers, trainers, etc.) have defined for the system. (Buede and Miller 2016, 5)

The term *interdisciplinary* is mentioned in the system's engineering description, this relates to nature of systems engineering design where multiple technical and management disciplines are required to participate and support the design and realization of the complex system. For example, disciplines may include electrical engineering,

mechanical engineering, structures engineering, reliability engineering, logistics, test-and-evaluation engineering, software engineering, contracting specialists, program managers, etc.

Another important term that is commonly associated with systems engineering is *concurrent engineering*. The term *concurrent engineering,* "simply means that the systems engineering process should be done with all of the phases (and their associated requirements) of the system life cycle in mind" (Buede and Miller 2016, 7). For example, concurrent engineering should consider logistical concepts, such as how the system is going to be maintained or repaired once fielded—these should not be afterthoughts of the design process.

2. Technical Processes and Technical Management Processes

The *Technical Processes* and *Technical Management Processes* presented here are DOD centric, but these do a universally good job at concisely defining the system engineering process throughout the system's development cycle from operational needs and the system's conception to realization and fielding of the manifested system. These, in a nutshell, are what the systems engineer does.

Technical management processes "provide insight of, and control over the technical development of a system throughout its life cycle," (Defense Acquisition University n.d.b) and "are the foundational, enabling processes and are used consistently throughout the system life cycle to help manage the system development" (Defense Acquisition University n.d.b).

Technical processes are used to "design, develop and analyze the system, system elements and enabling system elements required for integration, test, production, deployment, support, operation and disposal" (Defense Acquisition University n.d.c). The technical processes on the left side of the systems engineering "Vee" perform the "top-down" design, and the technical processes on the right side of the "Vee" perform the "bottom-up" realization. Figure 23 showcases the systems engineering technical and technical management processes in context of the systems engineering "Vee."

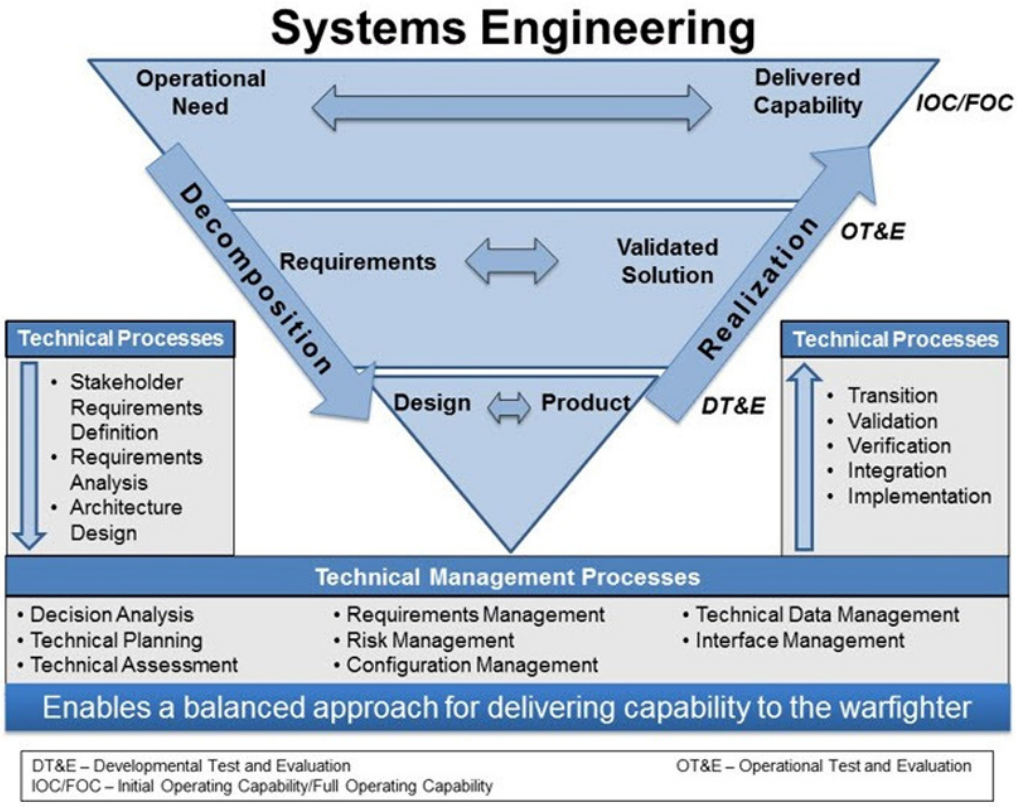


Figure 23. Systems Engineering Technical and Technical Management Processes. Source: Department of Defense (2022, 5).

3. Systems Engineering Approaches

Only a brief overview of the system engineering approaches is presented here. Depending on the system and situation, systems engineers should consider which approach may be best suited to efficiently achieve their design, realization, fielding, and support objectives.

System engineering approaches typically mentioned include: the Traditional, Top-Down Systems Engineering (TTDSE) process (Buede and Miller 2016)—it is arguably the same as the "Vee" model process; the *waterfall* method, and the *spiral* method (Buede and Miller 2016; Blanchard and Fabrycky 2011; Kossiakoff et al. 2020).

Figure 24 provides an illustration of the TTDSE, otherwise known as the systems engineering "Vee" model process.

NOTE: The systems engineering "Vee" is presented in this document various times, but each instance shows slightly different details—this exemplifies how abstract, but also how complex and rich the underlying concepts of the "Vee" really are.

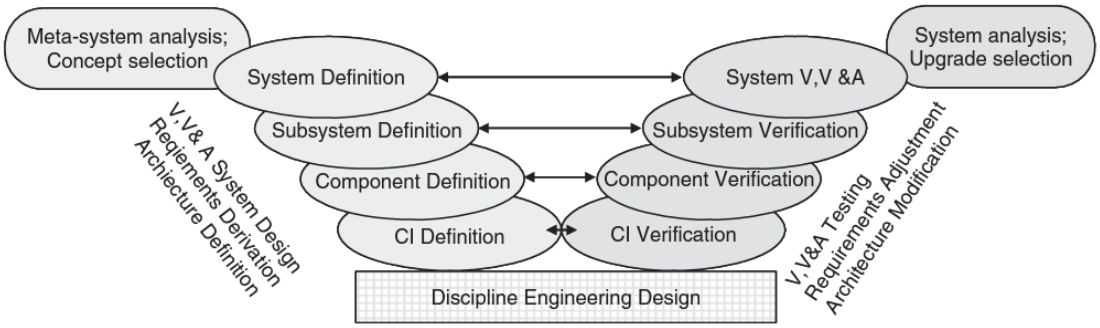


Figure 24. The Traditional, Top-Down Systems Engineering (TTDSE) view focusing on system definition maturation phases and their respective V&V processes. Source: Buede and Miller (2016, 20).

The waterfall model is probably just a poor visual abstraction or simplification of the real design process. The concept most likely emphasized by this model is that of *order* or *sequencing* dictated by the design maturation processes, but not much else. This concept is generally true, but the abstraction is too simple and ambiguous to precisely convey some of the other design complexities. It may portray one abstract "truth," but not the fuller set of "truths." It is mentioned here because this model has been referenced historically, but most recently, in a negative light. Many like to critique it and point out its "flaws"—some critics really took the waterfall abstraction to heart (or too literally), other critics use it as a contrast to prop up their more modern "agile" approaches. Figure 25 shows a graphical example of the waterfall model process.

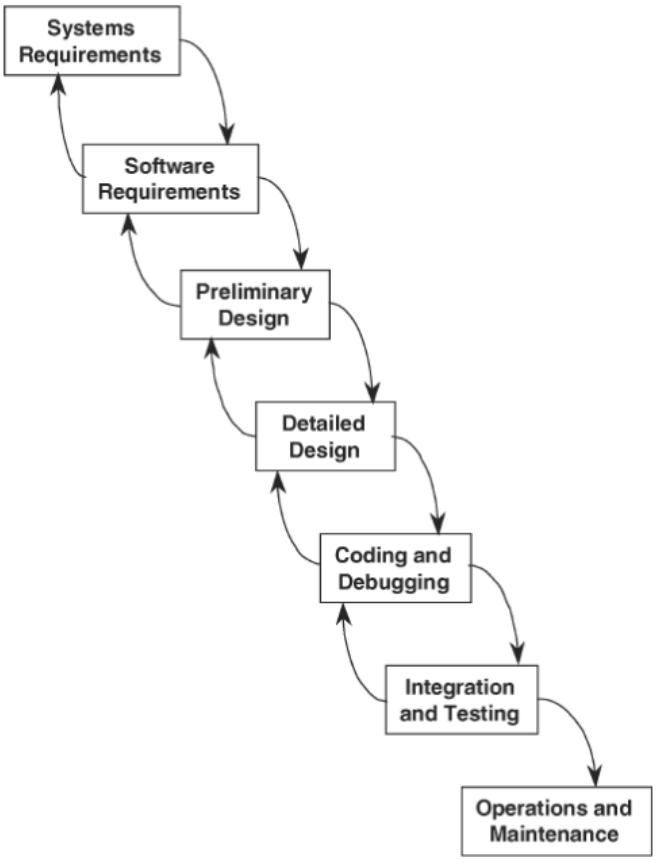


Figure 25. The Waterfall model. Source: Buede and Miller (2016, 22).

The spiral development model is more predominantly used to support software design due to the flexibility and fluidity of software. "According to Buede and Miller (2016, 21), the spiral model "has four major processes…moving clockwise: design, evaluation and risk analysis, development and testing, and planning with stakeholder interaction and approval. These four processes are repeated as often as needed." Buede and Miller (2016, 21) also state that the spiral model "views requirements as objects that need to be discovered, thus putting requirements development in the last of the four phases as part of planning." The spiral approach may be more suitable when stakeholders have not placed much effort into defining the problem domain and do not yet know exactly what the system must do or do not have a well-defined set of requirements to start from. The spiral approach lets designers start designing, building, testing, delivering, and getting feedback

from users through recursive increments or spirals. Each spiral should realize a more complete version/increment of the final product.

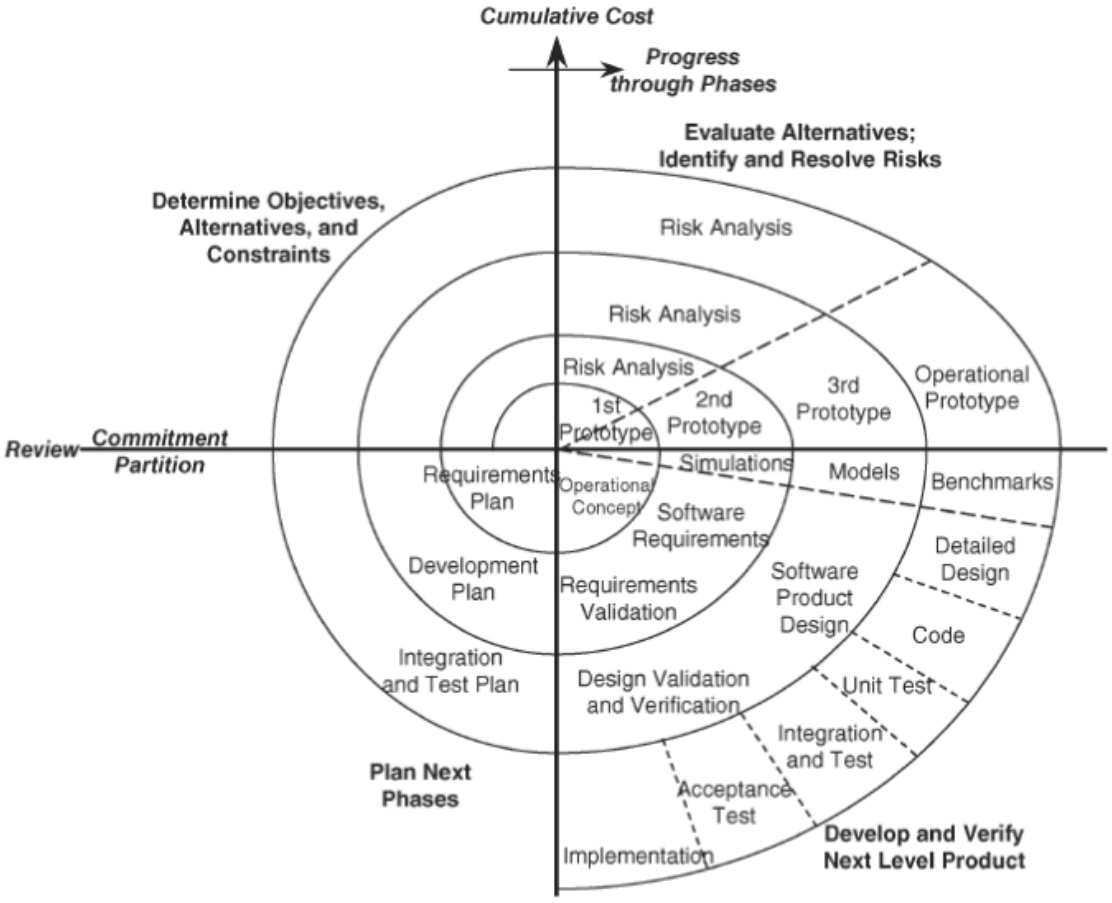


Figure 26. The Spiral model. Source: Buede and Miller (2016, 21).

4. System Engineering Activities

There are sequential phases in the systems engineering process. Each phase explores and matures the design so that the design exists more concretely in each subsequent phase until item "build-to" or "buy-to" specifications have reached leading up the manufacturing and integration of the system. Each phase discovers new knowledge, problems, and challenges, and has a feedback loop to its previous phases. It is most likely that the strongest feedback is to the most previous phase, but feedback can be provided to other previous stages in the design. The greatest design impacts are during the *detailed-*

design when "design engineering" is taking place and designers identify a problem that can only be rectified by making some fundamental change to the system architecture design, which is at a higher, more abstract level existing mostly in conceptual form. Such fundamental changes breach or disrupt the conceptual architecture baseline which includes the logical design for the functional architecture and the still somewhat logical or abstractly represented physical architecture components and their allocation. Even if feedback does not disrupt the higher-level system architecture, Interfaces must be minded and managed carefully to ensure successful integration of components and enabling of their functions.

Figure 27 shows a systems engineering "Vee" with a more detailed overview of the activities that occur at each phase (compared to other "Vee" models). This model shows more explicitly where the "design-to" and "build-to" specifications occur. This graphic is from Buede and Miller (2016) who draw a horizontal line dividing the "Vee." Their philosophy is that below this line, "Design Engineering" takes over where specialized engineering disciplines (electrical, safety, structures, etc.) and perform the *detailed-design*. The design then re-enters the systems engineering domain back when it crosses that horizontal line, going up the right side of the "Vee."

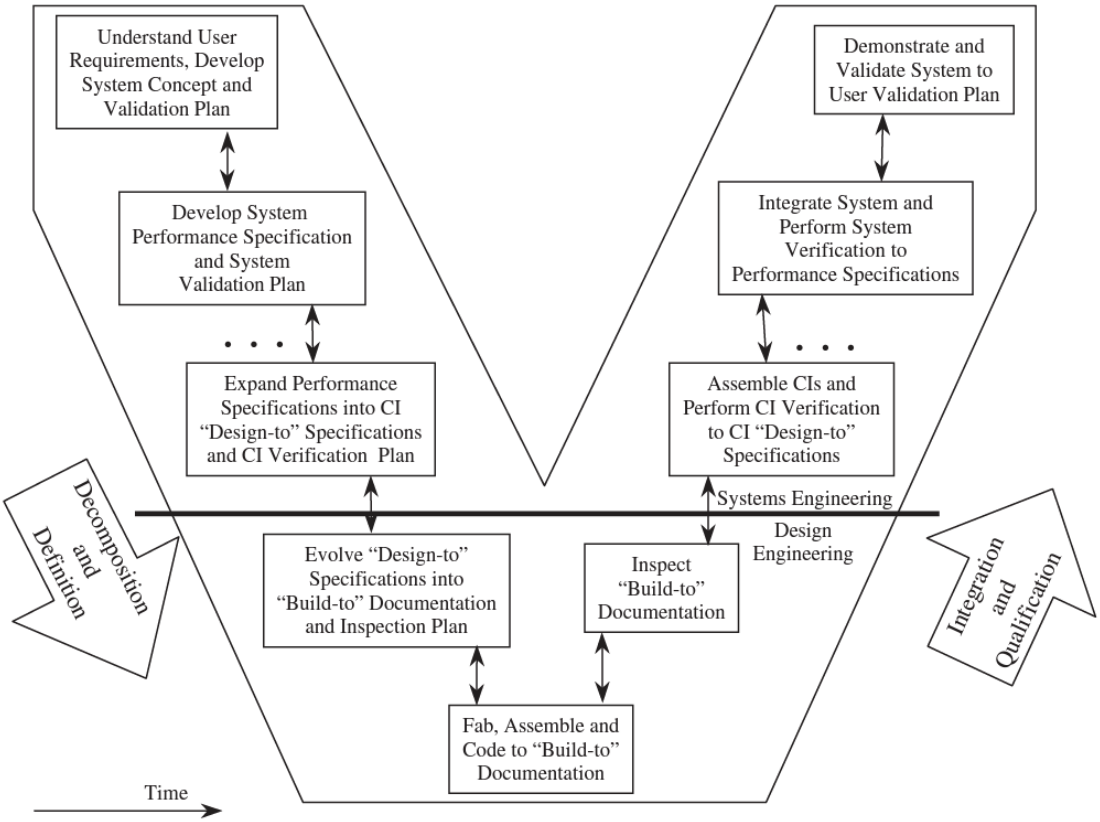


Figure 27. The Systems Engineering "Vee" focusing on Systems engineering processes. Source: Buede and Miller (2016, 10).

Buede (2009, 346) provides another visualization, shown in Figure 28, of the systems engineering design process in the form of an "Integrated Definition for Function Modeling" (IDEF0) diagram—more specifically, this is the upper-left part of the "Vee." The activity "Develop Qualification System" refers to developing or defining the verification plan that will test to see if the system is *built-right* and meets the *design-to* design specifications. The system is "qualified" if it passes the verification tests. However, Buede (2009, 55) states that, "qualification is verification, validation, and acceptance during integration," so it encompasses the entire right-side of the "Vee" up until the system is accepted. By Buede's (2009) definition, "Develop Qualification System," includes developing the validation plan that will validate or perform the final part of the "qualification" process after which the final qualification step is *acceptance*.

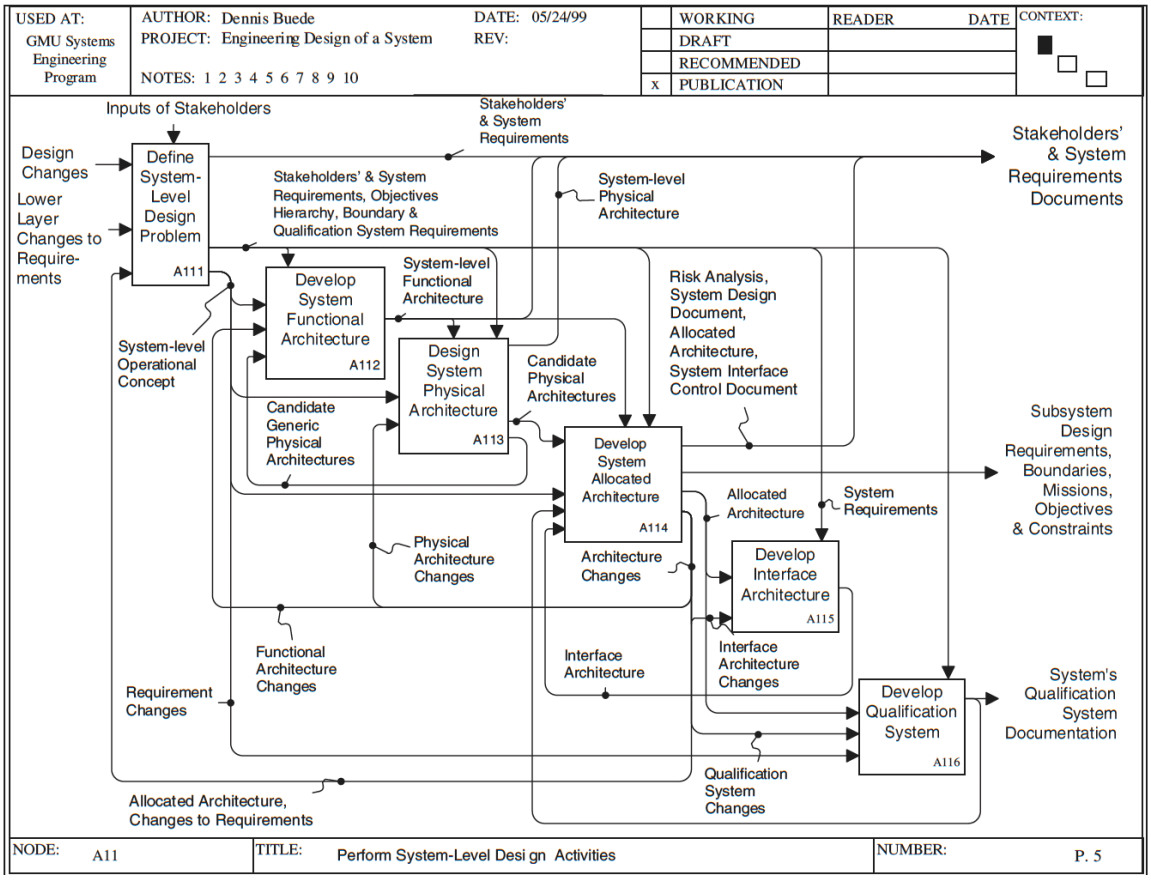


Figure 28. Development process for the interface architecture. Source: Buede (2009, 321).

Each of the main processes or phases follows a similar recursive pattern. Usually, the outputs from a previous phase serve as inputs to the current phase and are translated to more concrete design requirements. This lower-level set of design requirements guide the next iteration of functional analysis, concept (form) definition, and allocation. This means that that more abstract functional design details are made more concrete and precisely defined; then more concrete concepts (or elements of form) are identified and allocated resulting in the next state of the (synthesized) design. Figure 29 Shows a representation of this design recursive pattern.

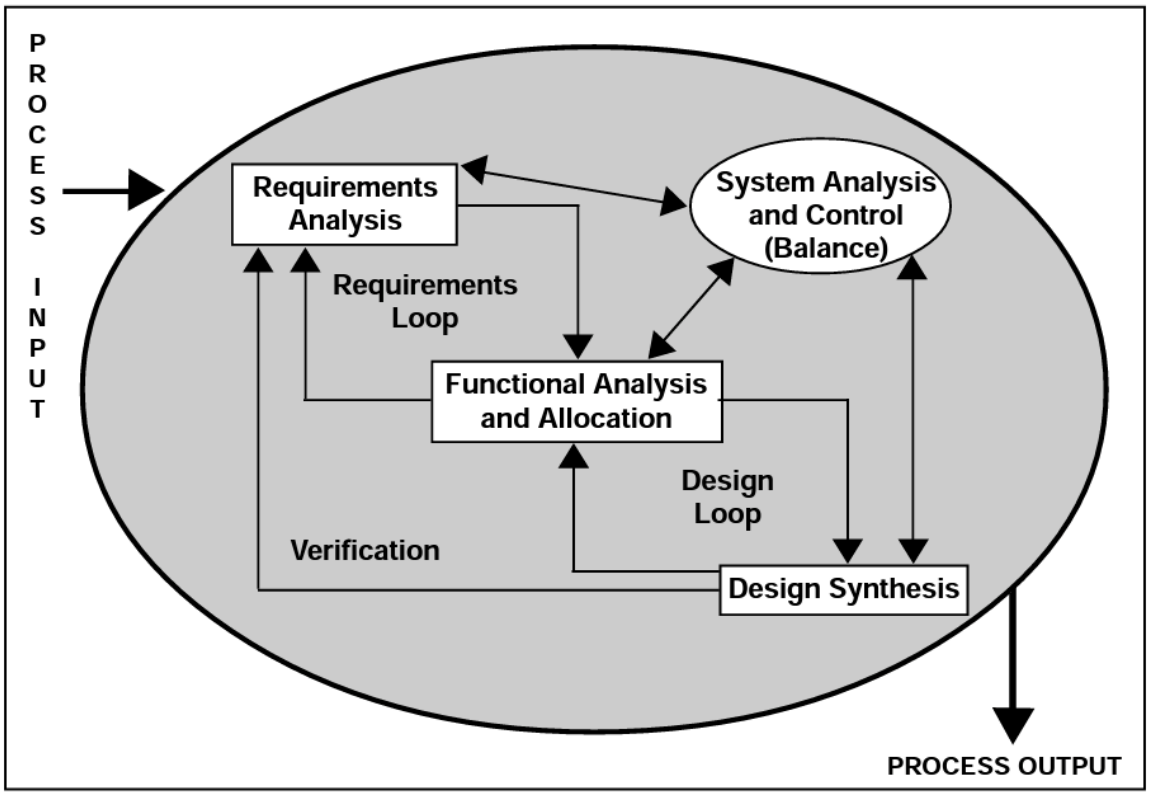


Figure 29. The systems engineering recursive process. Source: Systems Management College (2001, 6).

This pattern also emerges in the overarching systems engineering process. Perhaps this is what some systems engineers refer to when they say, "blowing up the 'Vee'," meaning that this pattern exists at the system-level, but also is also found "nested" in each of the system hierarchy levels, and across (horizontally) for each of the system components identified at each level. Figure 30 shows this pattern occurring at various system design levels, particularly after the *functional* and *allocated baseline*. This starts to delve into the *detailed design* portion where item specifications start to be defined.

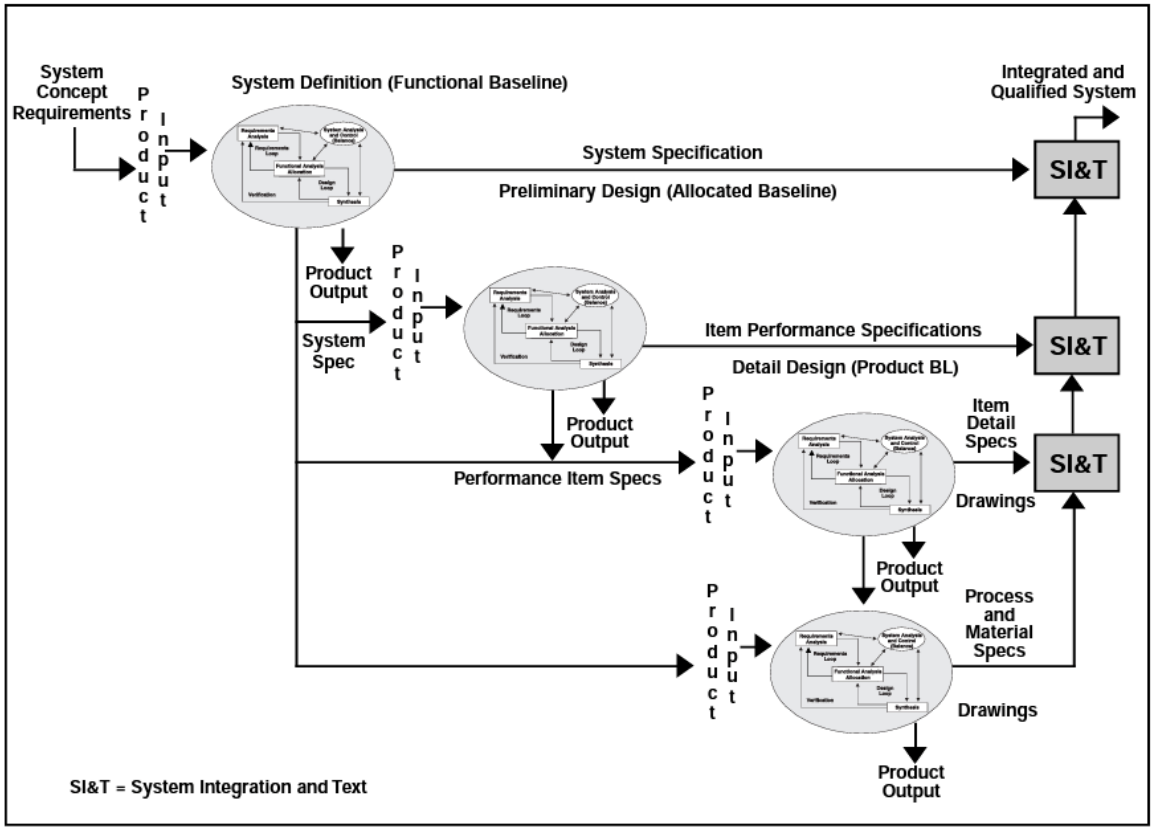


Figure 30. Recursive systems engineering design pattern and overall pattern for system levels of development and specification outputs. Source: Systems Management College (2001, 75).

The SYSMOD zigzag pattern is another model that conveys the recursive systems engineering design process. According to Weilkiens (2015, 50), "Requirements specify what the system should do and the system architecture and design how the system satisfies the requirements," but this general concept isn't so clean in real life. Weilkiens (2015, 50) asks, "what is part of the architecture and what is part of the requirements?" Weilkiens (2015) is talking about the recursive pattern of designing where a concept begins to define *how* the system solution-neutral or "solution-free" (Weilkiens 2015, 50) requirements are satisfied, but then the concept derives another set of design requirements, creating "what/how" (Weilkiens 2015, 50) pairs. Weilkiens (2015, 50) states that the "[how] solution leads to new requirements that contain aspects of the [most recently defined] solution." Weilkiens (2015, 51) states that "the requirements depend on a given architecture and are not free of any solution. They are on another abstraction level and

solution-free from the viewpoint of that level, but they contain solution aspects of the previous level." Weilkiens (2015, 51) then states, "again you derive a solution from the requirements." These concepts are better illustrated in Figure 31.

In Figure 31, there is a "solution-neutral" requirement stating that, "the system must provide live videos from all artifacts of the exhibition," to which then the designers provide a solution concept, i.e., the "Museum Robot." Designers then think within the concept envelope of the "Museum Robot" and envision attributes or operational behavior of this specific proposed solution. Designers then envisioned that museum robot solution must satisfy the following derived requirement "the museum robot must provide a collision detection." Assumingly, the robot navigates the museum halls (and potentially stairs and elevators too), but one of the robot's operational attributes is that it is safe or does not hurt people or damage other things—perhaps a more solution-neutral derived requirement for the robot would be stated as "the museum robot shall not hurt people or damage other things in the museum," from which then the solution concept of an "Anti-Collision System" might be derived.

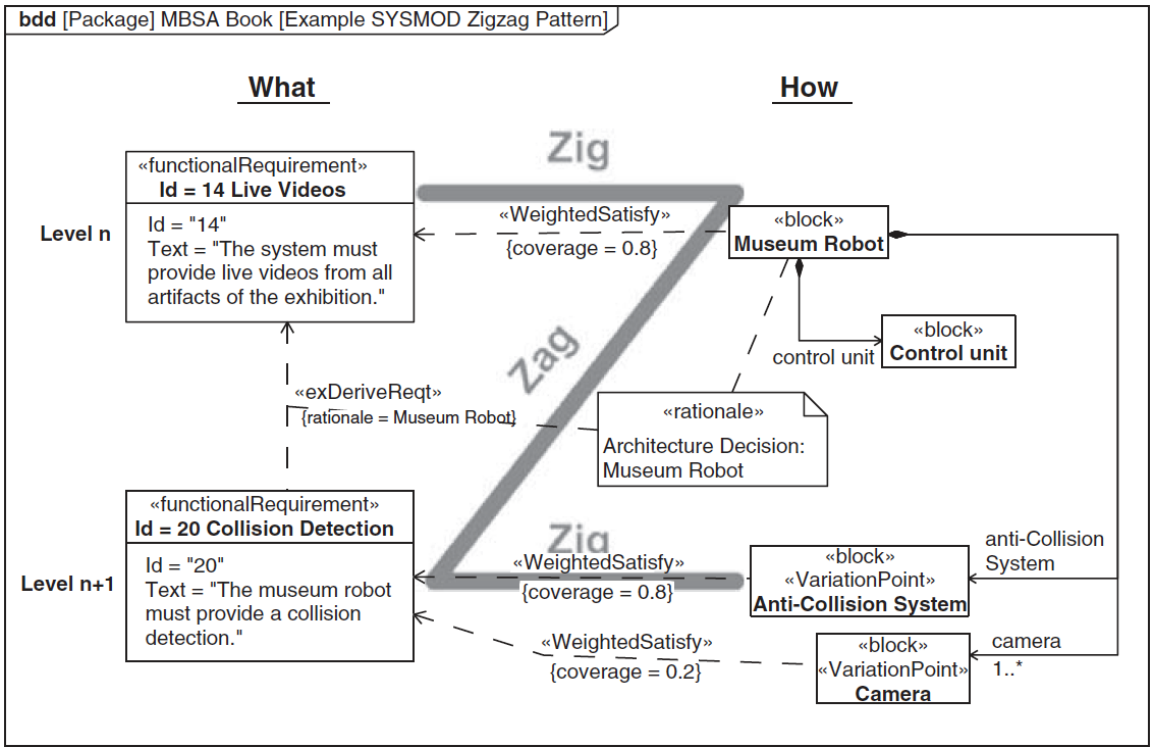


Figure 31. The SYSMOD zigzag pattern. Source: Weilkiens (2015, 51).

Weilkiens (2015, 51) describes the pattern in Figure 31, stating that:

> All in all the logical steps represent a zigzag pattern. "Zig"—from requirements to a system architecture—and "Zag" back from the architecture to the what-side of the derived requirements. The again a "Zig" from the derived requirements to the architecture, and so on.

5. Systems Engineering Technical Reviews, System Baselines and System Specifications

The Systems Engineering Technical Review (SETR) process maps technical reviews to the (DOD 5000) acquisition process (AcqNotes 2021). Each SETR event assessment is "focused on verifying technical health [including risks] and maturity by examining objective products representing the program work accomplishments to date," and "formally review and evaluate whether the systems engineering tasks have been satisfactorily completed prior to proceeding with the next SETR event" (AcqNotes 2021). In this regard, SETR events can serve as systems engineering controls or gates, where sufficient system design maturation and acceptable levels of risk must be demonstrated

before the program can move forward. SETR events are event-driven, not schedule-driven; progress is through a successful chain-of-event achievements that lead to the maturity point required for the SETR event. SETR events are also mapped to the systems engineering "Vee." Figure 32 shows the mapping of SETR events in relationship to the systems engineering processes that occur along the "Vee."

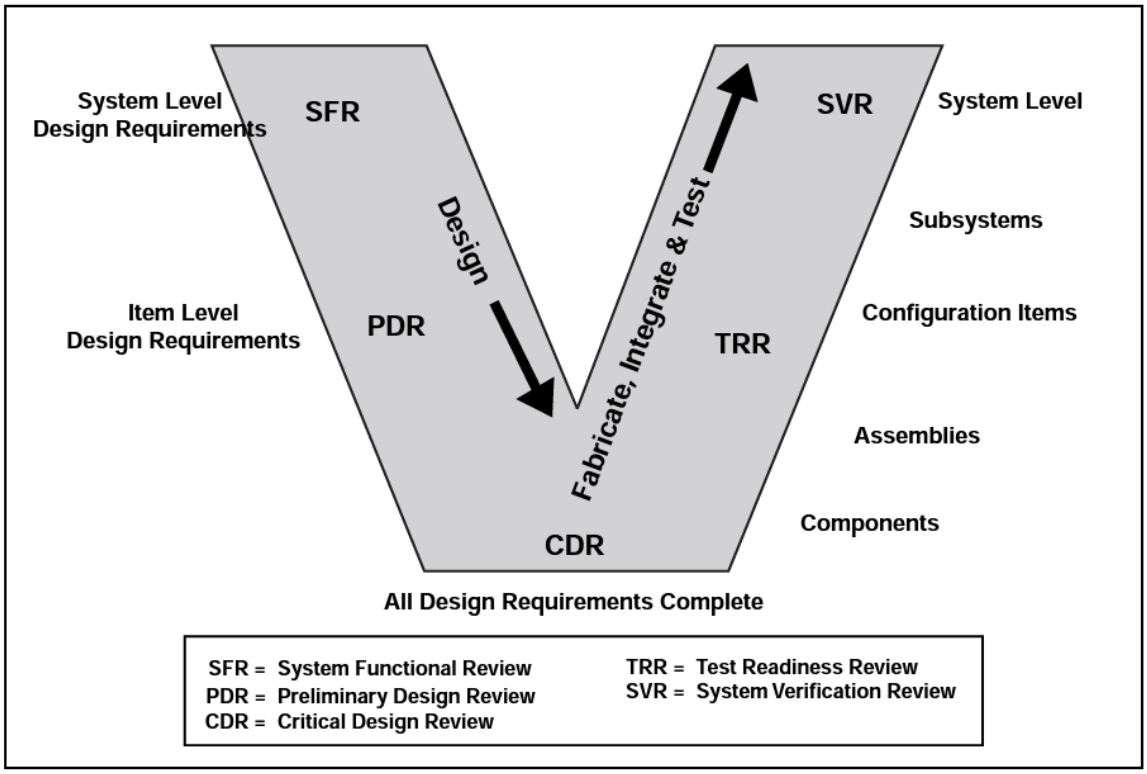


Figure 32. Systems engineering technical review (SETR) events mapped to the systems engineering "Vee." Source: Systems Management College (2001, 65).

Figure 33 shows a similar mapping, but is more granular, showing more explicitly other SETR events that a program may use, and also shows the technical and technical management processes as a reference.

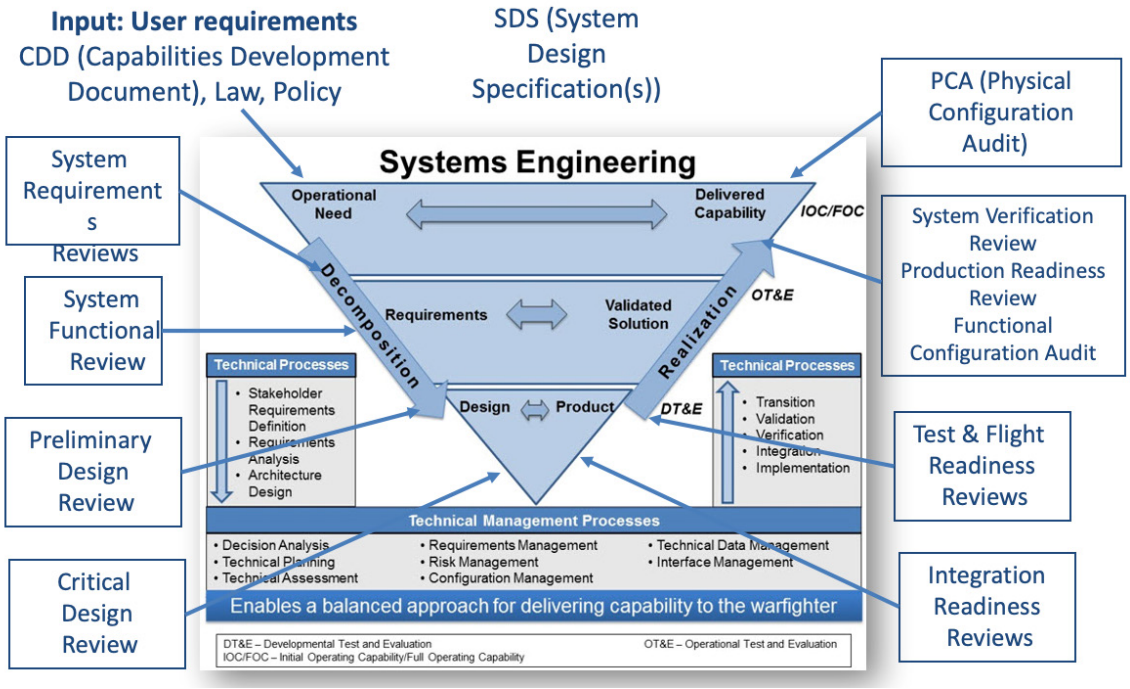


Figure 33. The Systems Engineering "Vee" referencing the Technical and Technical Management processes, and the Systems Engineering Technical Review (SETR) events. Source: College of Research and Engineering (2019e, 72).

Another important contribution of systems engineering is the definition of system technical baselines. Systems engineering technical baselines overlap with those of systems architecture. The functional baseline corresponds to the system functional architecture; the allocated baseline corresponds to the allocated architecture, and the product baseline corresponds to the physical architecture. The systems engineering process also defines other baselines that are not generally considered to be architecture centric, like the requirements baseline, but some frameworks for systems architecting, like the MagicGrid framework, do provide a formal way to document requirements. Once a baseline is established, stakeholders and designers control the baseline by managing changes to the baseline.

Figure 34 shows the mapping between technical baselines (green), SETR events (orange), and their context with respect to the system hierarchy.

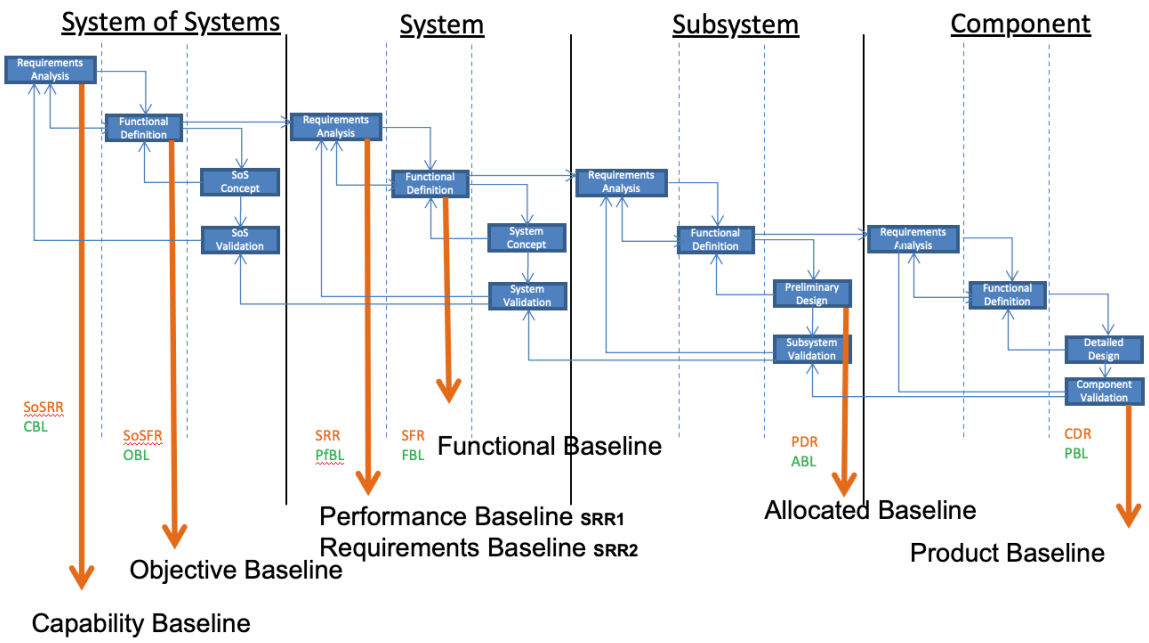


Figure 34. Systems architecting design process mapping to technical reviews and system baselines. Source: College of Research and Engineering (2019c, 43)

Table 2 shows the relationship between SETR events, and their corresponding system baselines a little bit more cleanly and expands the SETR event acronym definitions.

Table 2. Table mapping systems engineering technical reviews to their respective system baseline.

Systems Engineering Technical Review (SETR)	**Systems Baseline**
System of Systems Requirements Review (SoSRR)	Capability Baseline
System-of-Systems Functional Review (SoSFR)	Objective Baseline
System Requirements Review 1(SRR1)	Performance Baseline
Systems Requirements Review 2 (SRR2)	Requirements Baseline
System Functional Review (SFR)	Functional Baseline
Preliminary Design Review (PDR)	Allocated Baseline
Critical Design Review (CDR)	Product Baseline

System baselines are useful because they facilitate the formal documentation of system specifications and help control or management specification changes at the appropriate level. System baselines provide a hierarchical way to organize and structure system specifications with respect to the baseline level. Figure 35 shows a graphical example of how system specifications are organized according to the system baselines and SETR events.

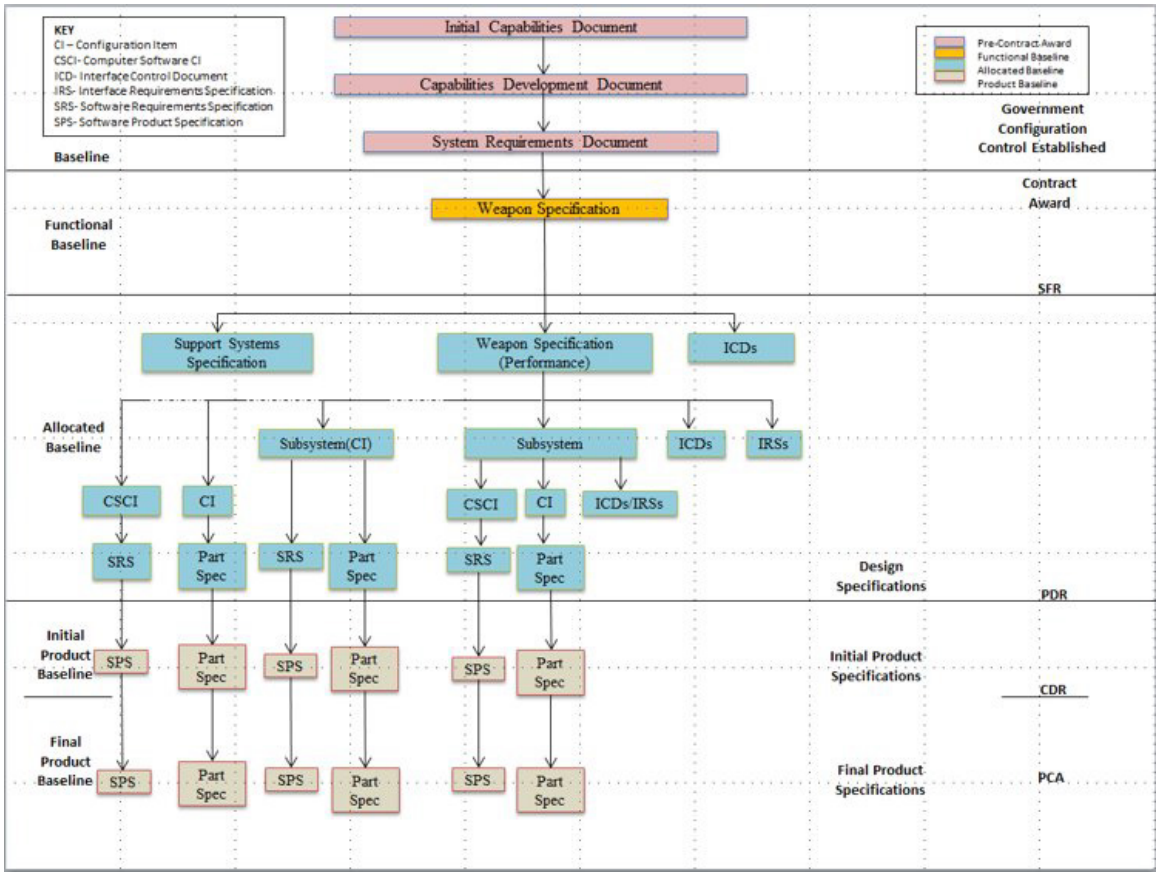


Figure 35. System technical baselines specification tree. This image is from the author's personal collection and is not publicly accessible. Source: College of Research and Engineering (2019d, 60).

Figure 36 shows the various types of specifications, their definition, and the baseline that the correspond to.

Specification	Content	Baseline
System Spec	Defines mission/technical performance requirements. Allocates requirements to functional areas and defines interfaces.	Functional
Item Performance Spec	Defines performance characteristics of CIs and CSCIs. Details design requirements and with drawings and other documents form the Allocated Baseline.	Allocated "Design To"
Item Detail Spec	Defines form, fit, function, performance, and test requirements for acceptance. (Item, process, and material specs start the Product Baseline effort, but the final audited baseline includes all the items in the TDP.)	Product "Build To" or "As Built"
Process Spec	Defines process performed during fabrication.	
Material Spec	Defines production of raw materials or semi-fabricated material used in fabrication.	

Figure 36. Specification types. Source: Systems Management College (2001, 75).

Figure 37 shows the evolution process of system specifications and how they are refined to the more concrete, lower-level, specifications that support building the system.

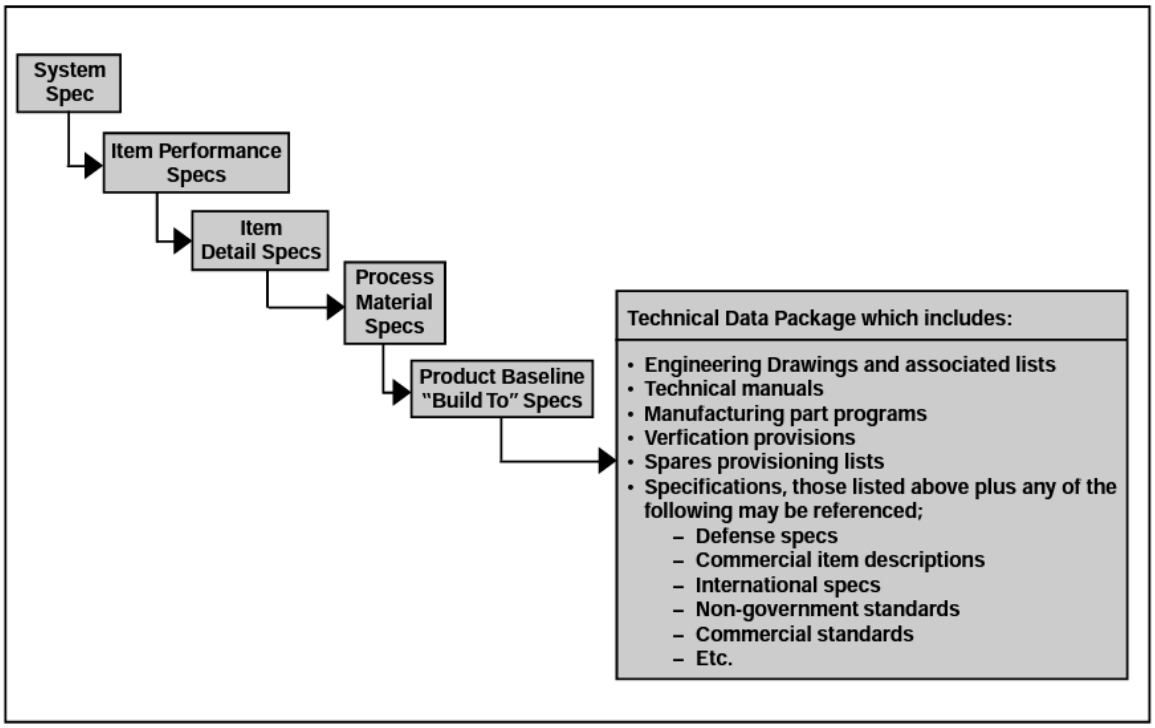


Figure 37. The specification process leading to the technical data package. Source: Systems Management College (2001, 76).

6. System Design Considerations

Systems engineering is a discipline that considers the life cycle of the system being developed. To do so, systems engineering must make a wide array of design considerations that will ideally support all aspects of the system life cycle. The design considerations support a *concurrent design* (Buede and Miller 2016) meaning that all phases of the system life cycle are considered. These design considerations are derived from the diverse set of technical and management disciplines' needs and objectives. See Table 3.

Table 3. Examples of Design Considerations from Blanchard and Fabrycky and the Defense Acquisition University—considerations are not matched for similarity. Adapted from Blanchard and Fabrycky (2011) and DAU (n.d.a).

	System design considerations by Blanchard and Fabrycky (2011, 38)	Design considerations by the Defense Acquisition University (Defense Acquisition University n.d.a)
1.	Compatibility/Interoperability	Accessibility (Section 508 Compliance)
2.	Supportability/Serviceability	Affordability (Systems Engineering Trade-Off Analyses)
3.	Flexibility and Expandability	Anti-Counterfeiting
4.	System/Product Sustainability	Commercial-Off-the-Shelf (COTS)
5.	Environmental Sustainability	Corrosion Prevention and Control
6.	Producibility/Constructability	Critical Safety Item (CSI)
7.	Product/System Quality	Demilitarization and Disposal
8.	Reliability and Maintainability	Diminishing Manufacturing Sources and Material Shortages (DMSMS)
9.	Recyclability/Disposability	Environment, Safety, and Occupational Health (ESOH)
10.	Human Factors and Safety	Human Systems Integration (HSI)
11.	Affordability/Profitability	Insensitive Munitions
12.	Performance/Functionality	Intelligence (Life Cycle Mission Data Plan)
13.	Social/Political Feasibility	Interoperability and Dependencies
14.	Technology Feasibility.	Item Unique Identification (IUID)
15.		Modular Open Systems Approach (MOSA)
16.		Operational Energy
17.		Packaging, Handling, Storage, and Transportation (PHS&T)
18.		Producibility, Quality, and Manufacturing (PQM) Readiness
19.		Reliability and Maintainability (R&M) Engineering
20.		Spectrum Management
21.		Standardization
22.		Supportability
23.		Survivability and Susceptibility
24.		System Security Engineering

Developing M&S products may require some special design considerations, such as *data validity* or *data availability*. From the table above, other good examples may be *interoperability of data flows*, *Performance/Functionality*, *Flexibility, and expandability.* These examples are subjective and not exhaustive. The determination of which design considerations to choose must be evaluated and selected by the stakeholders and designers; once selected these design considerations can be used as goals or objectives that can steer the project and design to the desired (cost, schedule, and performance) balance and technical realization of the M&S product.

7. Measures of Effectiveness, Measures of Performance and Key Performance Parameters, and Key System Attributes

It is important to cover Measures of Effectiveness (MoEs) and Measures of Performance (MoPs) because these measures align with the system's purpose and intended use, and they serve as validation benchmarks for the system. A system that is *effective* is a system that satisfies or accomplishes its operational concepts, which are translated into MoEs; this is an indicator that the system is *valid*, or that it is the *right* system. This means that the system achieves *what* it must do. The measures, particularly the performance measures provide a quantitative value for *how well* the system must perform its effectiveness functions.

Simply stated, Measures of Effectiveness (MoEs) have to do with system-level emergent behavior that directly addresses mission operational concepts. MoEs address the *what* the system must do, or what objective or outcome it must achieve. The mission operational concept space is bound by contextual parameters provided by the environment and accompanying systems. This considers the system's cooperation and interoperability with other accompanying systems that are part of the overall mission and the environmental envelope parameters in which the system is expected to perform.

A Measure of Performance (MoP) is considered to be lower on the system behavior hierarchy. MoPs address the *how well* a system must perform given the context and parameters. A combination of MoPs may be combined to result in the (higher-level) emergent MoE.

An example of an MoE and MoP may be as follows:

The system is an aircraft fighter/interceptor. One of its main operational concepts is to intercept enemy aircraft, particularly enemy bombers. Its MoE may be stated as "intercept" enemy aircraft." This MoE behavior can be instanced or reduced to an event, its parameters or conditions, and some probability that the desired outcome is achieved. A high-level MoE could be the "number of enemy aircraft shot down" given the mission scenario. The outcome may be stated as a *probability* that it is achieved. Some probability ranges or intervals can be established to evaluate the system satisfying the MoE, for example, 60% probability that an engagement event results in successful shooting down of an enemy aircraft; 30% results in disengagement or mission abort, and 10% results in taking damage from the enemy defenses. This behavior may be decomposed to identify some lower-level behaviors that contribute to the MoE behavior, such as the system's speed, rate-of-climb, range, weapons range and accuracy, radar enemy detection ability, etc. These lower-level behaviors may be classified as the MoPs and are more quantitative in value and can be stated as system parameters.

a. MoE and MoP definitions and characteristics from literature

The following provide direct definitions of MoEs and MoPs:

(1) MoEs

> Measures of Effectiveness are designed to correspond to accomplishing mission/system objectives and achieving desired results. They quantify the results obtained by a system and may be expressed as probabilities that the system will perform as required. MoEs measure changes in a system's capability, operational environment, or behavior related to achieving a goal, achieving an objective, or producing an effect, but task performance is not measured. MoEs may be further decomposed into Measures of Performance (MoP) and Measures of Suitability (MoS). (AcqNotes 2023)

> Measure of Effectiveness (MOE) is a measure of the ability of a system to meet its specified needs (or requirements) from a particular viewpoint. This measure may be quantitative or qualitative and it allows comparable systems to be ranked. These effectiveness measures are defined in the problem-space. Implicit in the meeting of problem requirements is that threshold values must be exceeded. (AcqNotes 2023)

> Measures of Effectiveness should use assessment Indicators that are relevant, measurable, responsive, and resourced so there is no false impression of task or objective accomplishment. They should be easily understood and evaluated. Complicated MOE is more difficult to evaluate and can lead to confusion and a lack of understanding of the real issue. A mix of quantitative and qualitative indicators is used to evaluate MOEs to mitigate the risk of misinterpretation and overcome the limits of raw data in understanding complex situations. (AcqNotes 2023)

AcqNotes (2023) provides the following is a list of MoE indicators:

- Relates to performance
- Objective
- Simple to state
- Testable
- Complete
- Clear
- States any time dependency
- States any environmental conditions
- Can be measures quantitatively (if required, may be measured statistically or as a probability)
- Easy to measure
- Select only MoEs that measure the degree to which the desired outcome is achieved
- Use the same MoEs to measure more than one condition when appropriate
- Structure so that they have measurable, collectible, and relevant indicators
- Write as statements (not questions)
- Maximize clarity (AcqNotes 2023)

(2) MoPs

> A measure of a system's performance is expressed as speed, payload, range, time-on-station, frequency, or other distinctly quantifiable performance features. Several MOPs and/or Measures of Suitability (MOSs) may be related to the achievement of a particular Measure of Effectiveness (MOE).(AcqNotes 2023)

> Measures of Performance (MOP) are necessary for correlation to specific MOEs in order to determine the optimal levels of effort for objective achievement. A careful analysis of the relationship between MOPs and MOEs reveals the need to shift COAs if the current plan is inefficient, or have adverse effects. (AcqNotes 2023)

The following quotes are from a report produced by Green (2001) which provides some additional characteristics that are not often emphasized in other literature.

> "MOEs" provide quantifiable benchmarks against which the system concept and implementation can be compared….
>
> effectiveness measures are critically important because they are the criteria that drive the system solution that is found. (Green 2001, 1)

Green (2001) noticed that the definition of functions and measures are bound by the system boundaries, therefore system boundaries play an important role in defining MoEs and their context. Green (2001,1) identified that "an open system interacts with the environment beyond its boundaries," thus noticing the system context level that frames the system's interactions with its environment and accompanying systems. With respect to the system Green (2001,1) states that "a system is a unified collection or set of objects that exhibit a unique set of behaviors when combined together in their operating environment." This statement relates to the concept of a system's (internal) functional architecture and the physical elements and structure that enable it. Green's (2001) statement also the relates to the concept of the emergent system-level behavior and value delivered at the system boundary when the system interacts with its exterior. During the top-down design phase, Green (2001,1) noticed that the systems engineering technical processes "capture the systems [sic] behavior and define the effectiveness measures, the criteria that means success or failure." This is done at various levels of the system's decomposition. Therefore, he correlates the important relationship between a system's effectiveness or performance measures and the corresponding system boundary that enables the behavior tied to those measures. Green (2001, 2) states that "the logic behind bounding the system early is driven by the performance measures development process," and that "the system boundaries define the set of system parameters that drive system performance. A change in the boundaries changes the parameter set and the resulting system behavior and its performance."

Here are some concise statements that Green (2001) makes about MoEs and MoPs; these are useful for further grasping the essence of MoEs and MoPs:

- Effectiveness measures are derived from first principles. They are mission and scenario dependent. (Green 2001, 3)
- Effectiveness measures must be both measurable and testable. This means they are quantitative in nature. (Green 2001, 3)

- MOPs quantify the set of selected parameters. Examples include sensor detection probability, sensor probability of false alarm, and probability of correct identification. (Green 2001, 3)
- [A measure of Effectiveness (MOE) is a] measure of how a system performs its functions within its environment. An MOE is generally an aggregation of MOPs. Examples include survivability, probability of raid annihilation, and weapon system effectiveness. (Green 2001, 3)
- Effectiveness measures must be independent at the level of analysis evaluation. In other works, MOPs should be independent but they can be aggregated into MOEs. (Green 2001, 3)
- If it can't be expressed as a *probability* [emphasis added] it probably is not an effectiveness measure. (Green 2001, 4)

(3) System Attributes

System Attributes is a concept that is closely related to MoEs and MoPs. This concept is important to know about in general but is not essential for supporting this thesis's research. only simple definitions are be provided.

> Attributes are the properties (characteristics, configuration, qualities, powers, constraints, and state) of the components and of the system as a whole. (Blanchard and Fabrycky 2011, 3)

> Typical quality attributes of an architecture are for instance performance, reliability, availability, security, producibility, disposability, and modifiability. They are tightly coupled to the physical architecture. Some of the quality attributes are also relevant for the functional architecture. (Weilkiens et al. 2015, 272–273)

> Every system has a set of properties that are largely a function of the system as a whole rather than just its constituent parts. There are dozens of properties such as security, reliability, safety, resistance to electromagnetic interference, usability and resilience. … Alternatively, they are called *quality attributes*, *non-functional requirements*, *-ilities*, and *specialities*. In this [Knowledge Area] KA, the term specialty engineering refers to the collective engineering of all quality attributes of a system….Quality attributes are often interdependent rather than orthogonal; e.g., making a system more secure may make it harder to use but also safer [in other words, enhancing one attribute may come at a trade-off, impairing another competing system attribute]. (Pyster and Olwell 2023)

System attributes are emergent qualities of the system as a whole and are enabled by the system's architecture. These attributes are generally known as the "-ilities" of a system. They may be attributed to a functional aspect of a system's behavior, such as

lethality, or they may be attributed to a physical or other characteristic of the system, such as *maintainability* or *affordability* (considering life cycle costs). Other attributes that don't neatly associate to an "-ility" maybe size, weight, system cost, speed, etc.

D. SYSTEMS ENGINEERING VALIDATION, VERIFICATION AND ACCREDITATION

The purpose of this section is to provide a concise summary of systems engineering VV&A concepts and definitions; doing so will provide a reference for comparison against M&S VV&A concepts and definitions—note the similarities between systems engineering VV&A concepts and definitions and those of M&S VV&A. Table 3. Shows a simple mapping of System engineering VV&A concepts and how they relate to M&S VV&A concepts.

Table 4. Similarities between Systems Engineering and M&S VV&A concepts.

Systems Engineering VV&A	Relationship	M&S VV&A
Operational validity	Similar to	Model Operational validation
Conceptual validity	Similar to	Conceptual model validation
Requirements validity, and Design validity	Similar to	Specification verification
Verification	Similar to	Computerized model verification

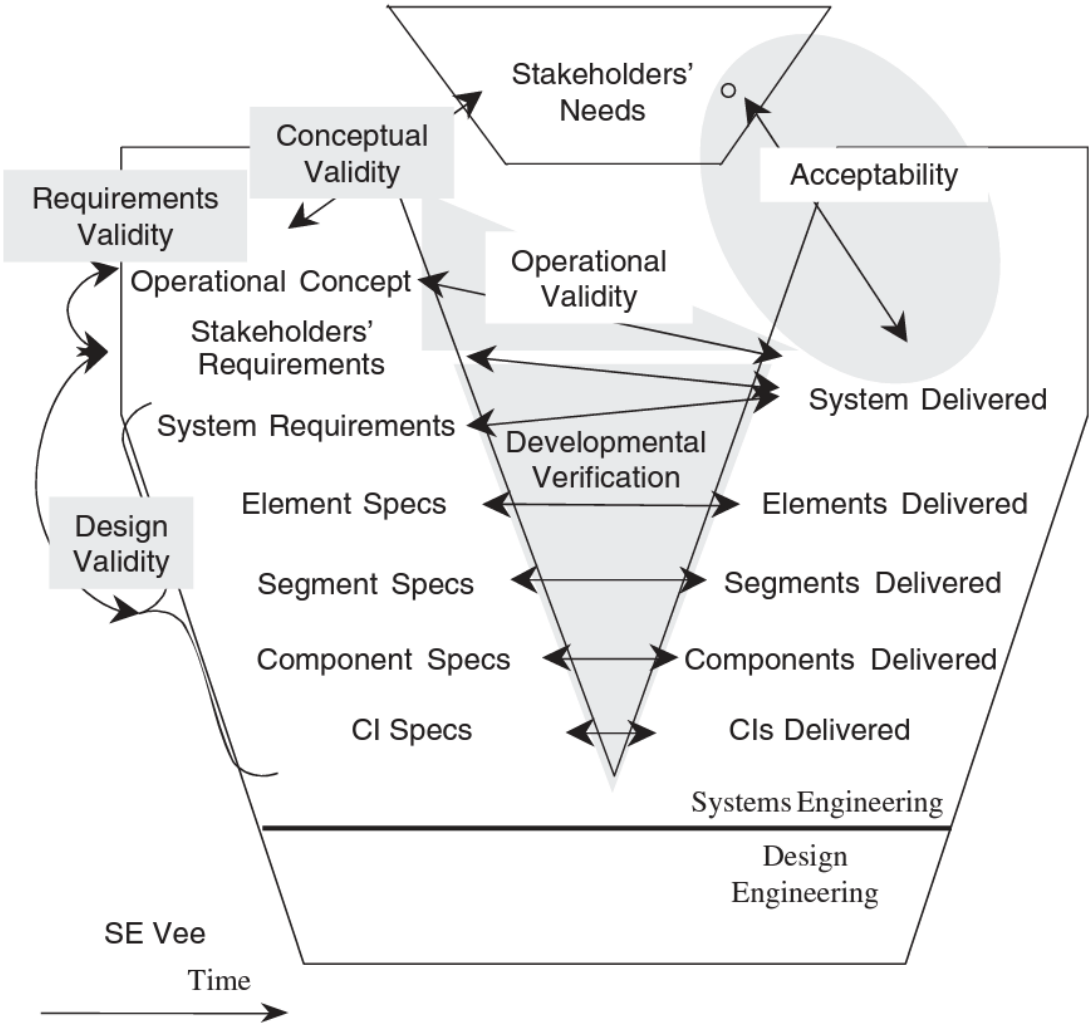


Figure 38. Systems engineering "Vee" validation, verification, and acceptance (VV&A) concept mappings. Source Buede and Miller (2016, 344).

The following definitions are provided by Buede and Miller (2016, 342–345). These definitions are consistent with other definitions and concepts found in systems engineering theory and provide a suitable base of comparison for VV&A concepts and definitions used in M&S.

- *Acceptance* is a stakeholder function for agreeing that the designed system, as tested or otherwise evaluated by the stakeholders, is acceptable….acceptance is driven by the stakeholders, with the knowledge of the results of validation and verification activities that preceded it.

- *Validation* is the process of determining that the systems engineering process has produced the *right system*, based upon the needs expressed by the stakeholder.
- The most reliable and early statement of the stakeholders' needs is the operational concept. Therefore, *operational validity* is the matching of the capabilities of the designed system to the operational concept; this naturally occurs late in the integration phase after the designed system has been verified.
- *Conceptual validity* is the correspondence between the stakeholders' needs and the operational concept. Conceptual validity needs to be established at the outset of the design process via interactions among the systems engineers and the stakeholders…[if] stakeholders' needs change…the operational concept must change with those needs.
- *Requirements validity* is the correspondence between the operational concept and the stakeholders' requirements. In requirements validity the operational concept is assumed to be an accurate reflection of the stakeholders' needs; the validation occurs by establishing that the stakeholders' requirements have neither introduced new issues nor left issues out of the operational concept, thus causing the design of a different system that envisioned in the operational concept.
- *Design validity* assumes that the Stakeholders' Requirements Document (StkhldrsRD) is a valid statement of the stakeholders' needs and addresses the congruence between the StkhldrsRD and the derived requirements. The derived requirements begin with the Systems Requirements Document (SysRD), evolve to subsystem and component specifications, and culminate in CI [configuration item] specifications.
- *Verification* is the matching of CIs [configuration items], components, subsystems, and the system to their corresponding requirements to ensure that each has been *built right*. … In order for verification to be successful, the originating and derived requirements must be testable; that is the requirements must be single statements that are unambiguous, understandable, and verifiable. (Buede and Miller 2016, 342–345)

Another concept that over compasses the conceptual validity, requirements validity, and design validity is the concept of *early validation*—not explicitly shown in Figure 38. Designers should be looking to exploit opportunities to perform early validation which will help steer them towards realizing the *right system*. This is why it is important to seek continuous stakeholder feedback and to formally capture and document information pertaining to each of the validation processes—conceptual, requirements, and design validation. The following passage is provided by Buede and Miller (2016):

> Conceptual, requirements, and design validity are called *early validation*, the determination that the right problem is being defined at the current level of abstraction, given the validity of the problem definition at a higher level of abstraction. (Buede and Miller 2016, 342)

Much effort is placed on discussing validation, which deals with the *right* system and the *right* concepts that addresses the *right* problems. Verification deals with the correctness of the system with regards to its design specification. In this case, correctness doesn't have to do with the purpose or application of the system; correctness only deals with the system being built *right*. Figure 39 shows the verification action pattern where a system element, either a function or physical characteristic is verified against its design specification or requirement, shown as the "reference." Verification is some type of method: test, analysis, inspection, or demonstration.

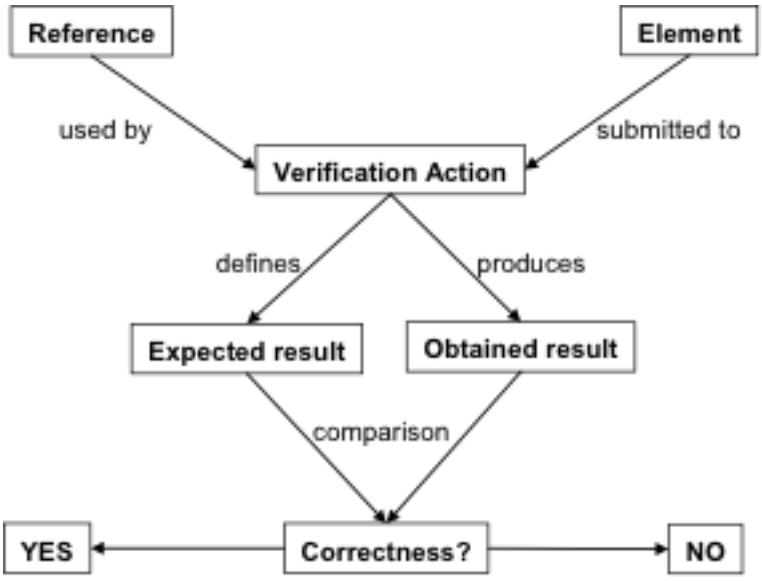


Figure 39. Verification Action from the Systems Engineering Body of Knowledge. Source: Snoderly and Faisandier (2023).

E. MODEL-BASED SYSTEMS ENGINEERING

Model-based systems engineering is a key enabler for carrying out the research work that supports main thesis question, "can Model-Based Systems Engineering (MBSE) be leveraged to develop a re-usable framework and process for Validation Verification and

Accreditation (VV&A) of Modeling and Simulation (M&S) models?" A summary of the key aspects of MBSE is presented in this section so that readers new to MBSE can have a fundamental understanding of MBSE and be able to appreciate and evaluate this thesis' research methodology.

For convenience, the definition of MBSE provided in Chapter I., "Introduction," is presented again here followed by an alternate definition of MBSE that may be more supporting of other "engineering domain models" (McBee 2022).

> MBSE is the formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases. (INCOSE 2007)

The following quote is from Nate McBee (2022), the senior manager of Ansys Government Initiatives (AGI) Product Management, who posted an Ansys blog titled "Model-Based Systems Engineering (MBSE) Explained":

> Model-based systems engineering (MBSE) is a methodology that focuses on creating and exploiting digital system and engineering domain models as the primary means of exchange of information, feedback, and requirements, as opposed to document-centric systems engineering. It involves the entire process of capturing, communicating, and making sure that all the digital models we use to represent a system are coordinated and maintained throughout the entire life cycle of the system….
>
> MBSE was developed to replace static documents with "intelligent" digital models that contain everything important about the system — the requirements, the architecture, and the interfaces between the pieces of the system. Instead of paper documents that were at best organized into folders, these digital models are connected by a "digital thread" that can be followed to understand the entire design. (McBee 2022)

McBee's (2022) definition provides a perspective that seems to be more inclusive of "engineering domain models." Based on Ansys's product line, and analyzing their technical marketing graphic, presented in Figure 40, it can be inferred that "engineering domain models" refers to the mechanical, electrical, fluids, software, and safety models mentioned on the right box (in Figure 40). These are physics-based models governed by physical equations.

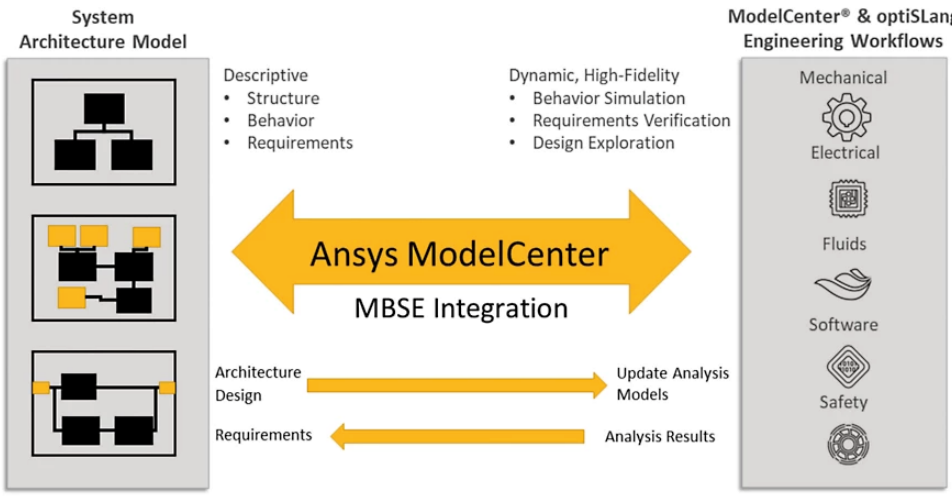


Figure 40. Ansys MBSE domain perspective. Source: McBee (2022).

Figure 40 showcases a noteworthy concept. It is believed that this is the current maturing trend of MBSE where a descriptive system architecture model and some "engineering domain models" are integrated so that seamless transfer of data can happen, thus supporting inputs/outputs necessary for a continuous thread analysis. The descriptive architecture model is used to define the system elements and element relationships while the engineering domain model or set of models are specialized to deal with a particular domain of engineering such as structural mechanics, fluid dynamics, thermodynamics, etc.

> The overarching systems architect model (SAM) serves as an "authoritative source of truth" for everyone working on the project. This digital model has a central location that can be accessed by all the engineers working on the project, but cannot be modified independently of others, preserving the single source of truth. Any changes made are automatically propagated throughout the model and checked for internal consistency and accuracy by the software. (McBee 2022)

For the purposes of this thesis research, the definition and perspective of MBSE is limited to the descriptive system architectural model. Thus, the presented concepts deal only with this definition and provide sufficient background and information to support the work done for this thesis.

1. The Three Pillars of MBSE –by Delligatti

To further define MBSE, Delligatti's (2013) three pillars, *modeling language*, *modeling tool*, and *modeling method*, will be presented along with the stated MBSE value proposition according to Delligatti (2013).

a. Modeling Language

> **Modeling Language**: [is] a semiformal language that defines the kinds of elements you're allowed to put into our model, the allowable relationships between them, and—in case of a graphical modeling language—the set of notations you can use to display the elements and relationships on diagrams…
>
> each modeling language is standardized medium for communication; the rules defined in a given language give the model's elements and relationships *unambiguous meaning* [emphasis added]. The capability to construct and read *well-formed models* [emphasis added] is at the heart of the MBSE approach…
>
> A modeling language dines a grammar: a set of rules that determines whether a given model is well formed or ill formed. (Delligatti 2013, 5)

The Systems Modeling Language (SysML) is the most commonly used graphical language used by systems engineers practicing MBSE, today. Other graphical modeling languages include: "UML, UPDM, BPMN, MARTRE, SoaML, IDEFx" (Delligatti 2013, 5). However, there also exist text modeling languages such as "Verilog" and "Modelica" (Delligatti 2013, 5). This thesis shall focus on SysML because it was the modeling language used to support the research methods.

b. Modeling Method

> A **modeling method** is something like a road map; it's a documented set of design tasks that a modeling team performs to create a system model. More precisely, it's a documented set of design tasks that ensures that everyone on the team is building the system model consistently and working toward a common end point. Without such guidance, there will be wide variance in the breadth, depth, and fidelity that each member of the team builds into the system model. (Delligatti 2013, 6)

Delligatti (2013,6) mentions that each model needs a "purpose" that addresses the intended use of the model and the reason why a model is being created in the first place.

Having a purpose can help define a "scope" (Delligatti 2013, 6) for the modeling details of what should and shouldn't be modeled (or is not necessary), and when a model is considered to be "complete"—meaning that the model satisfies the intended purpose (Delligatti 2013).

The modeling method used for this thesis research is Dassault Systemes' MagicGrid framework method (originally by NoMagic). Other modeling methods include INCOSE's Object-Oriented Systems Engineering Method (OOSEM)., Weilkiens' System Modeling (SYSMOD) method, IBM's Telelogic Harmony-SE method, etc. (Delligatti 2013).

c. Modeling Tool

> **Modeling tools** are a special class of tools that are designed and implemented to comply with the rules of one or more modeling languages, enabling you to construct well-formed models in those languages….
>
> Modeling tools are distinct from diagramming tools such as Visio, Schematic, Smart Draw, ProcessOn and others. With a diagramming tool, you create diagrams—shapes on a page. There is no model underlying those diagrams that ensures automated consistency between them. In contrast, with a modeling tool, you create a model—a set of elements and relationships between elements, and optionally a set of diagrams that serve as views of the underlying model….
>
> When you modify and element on a diagram within a modeling tool, you're actually modifying the element itself in the underlying model. The modeling tool then instantaneously updates all the other diagrams that display the same element. (Delligatti 2013, 7–8)
>
> The system model serves as a central repository for design decisions; each design decision is captured as a model element (or a relationship between elements) in a single place within the system model [emphasis added]. With the MBSE approach, all diagrams and autogenerated text artifacts are merely views of the underlying system model; they are not the model itself. (Delligatti, 2013, 3).

2. The MBSE Approach, "Same Same, but Different"

It should be emphasized that with an MBSE approach, "system engineers perform the same [system engineering] life cycle activities and produce the same set of

deliverables" (Delligatti 2013, 3). However, with the MBSE approach, "the primary artifact of those activities is an integrated, coherent, and consistent system-model…all other artifacts are secondary—automatically generated from the system model using that same modeling tool" (Delligatti 2013, 3).

Weilkiens et al. (2015, 30) makes a similar claim that the "key artifacts of the system architecture are stored in a system model." This is in contrast to a document-based approach where the "key artifacts are stored in documents and models are only used to contribute information to the documents, for instance, SysML diagrams in a text document" (Weilkiens et al. 2015, 30). Other terms similar in meaning to *Model-Based Systems Engineering* have been used, such as *model-driven* and *model-centric*. Model-centric seems to be a very close synonym to the term *model-based*. Weilkiens et al. (2015, 30) state that, "some say that *model-based* is a softer version of *model-driven*, that is, the model is an important asset, but not the source." However, Weilkiens et al. (2015, 30) also point out that "others say that both terms are synonyms." There does not appear to be any formal or official definitions for what any of these other similar terms to MBSE mean; they do not appear to cause ambiguity or confusion problems among the MBSE community and are generally regarded to be synonymous. This thesis document gravitates toward using the term *model-based*.

Using graphical languages to model the system design is not new. Graphical abstractions of the system have been done manually, pre-existing modeling computer programs. Use of graphical languages in the past have enabled designers to describe the system's physical structure and behavioral or functional logic at various levels of abstraction and decomposition; allowing the designers to visually model the system's architecture, at the higher-level, and at the lower-level subsystem or component *design-to* specification level. It is hard to trace the history and the use of graphical languages, but Weilkiens (2007) provides us a bit of insight with the following statement, "the activity model of SysML is compatible with the EFFBD popularly used in systems engineering" (Weilkiens 2007, 257). Weilkiens (2007, 257) then provides the following note with respect the EFFBD, "this [SysML activity] diagram is based on the *Functional Flow Block Diagram (FFBD)* developed by TRW in the 1950s; it represents complex flows in a very

simple way. EFFBD diagrams add data flow information to a flow." The "E" stands for "enhanced" or "extended." This research does not intend to absolutely prove the origin or history of graphical modeling languages, but an educated assumption is made that graphical languages were first used manually; diagrams were drafted by hand and as technology evolved, computer programs started taking over the drafter's role and provided modelers with more features and capabilities that allowed elements to be modeled in *one-place* and *one-time* in the underlying model which facilitated consistent re-use of model elements in diagrams or visualizations (views).

The computer modeling tool takes on the burden of "remembering" and maintain the complex set of existing model elements and relationships, ensuring automated consistency of changes to the model. Delligatti (2013, 9) states that "MBSE delivers its [return on investment] ROI when change happens—when new design decisions are made and stakeholders' needs evolve through the system life cycle. And change, of course, inevitably happens." It is the capability of the modeling tool that enables MBSE's return on investment; the pay-off is when the model needs to be changed, but the tool takes care of consistently propagating the impacts of the changes and updates the rest of the model accordingly. Without the tool, this effort and consistency would be on the human's shoulders—a very tall order when a model is large in scale and very complex.

It is the author's opinion that descriptive models provide the most value during the *preliminary-design*, when the system architecture is defined. During preliminary design both the physical and functional architecture are defined along with trace allocations of the physical elements to the corresponding functional elements. This system architecture is decomposed down to the level of detail where designers can provide unambiguous *design-to* specifications. The *design-to* specifications are used to guide the *detailed-design* which produces the *build-to* or *buy-to* specifications of the *product baseline*. Without good *design-to* specifications, it is very hard for engineers to 1) make sure that their portion of the detailed design will ultimately meet the stakeholders needs as some needs will not have been adequately captured in the preliminary design, or captured at all, and 2) inadequate or missing interface requirements will force the engineers to discover these when performing

their phase of design and integration which sometimes involves a test-build-test recursive pattern until the kinks are worked out.

3. Background on the Systems Modeling Language

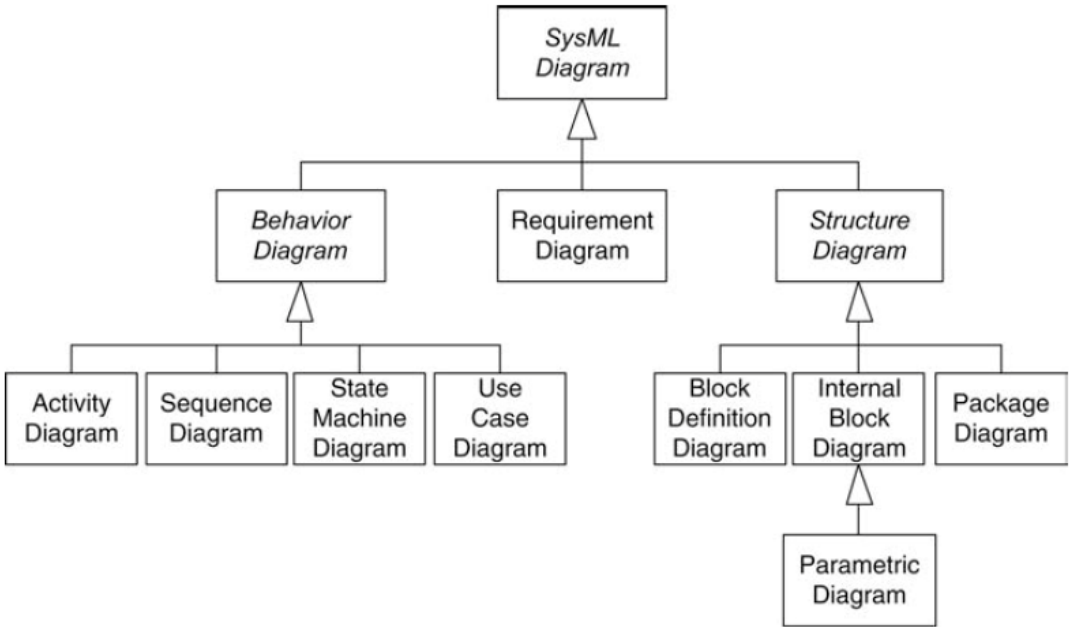
Only the basics of SysML are presented in this section; enough to support the reader's understanding what SysML is and understand its role with respect to this thesis' research.

a. SysML Diagrams

Keeping in mind what Delligatti (2013, 3) stated, "with the MBSE approach, all diagrams and autogenerated text artifacts are merely *views* of the underlying system model; they are not the model itself," It is important to redirect this statement and emphasize the value of diagrams. Diagrams present the primary *human-readable* source of information. The diagram allows model users to directly edit the model through modifying the diagram; adding, deleting, modifying elements and relationships directly from the diagram *frame*, commonly known as the "canvas." This is usually the way modeling elements are added and edited in the model—through the diagram canvas. Dr. Vaneman repeatedly states in his lectures and discussions that "diagrams are the common currency" (Warren Vaneman, class discussion, spring quarter 2023) of modeling tools—this is specifically significant when there is a need to exchange data from one modeling tool to another, which may include exchanging data across different modeling languages such as SysML, Life cycle Modeling Language (LML), Specification Description Language (SDL), etc. This is often desired when different teams using different tools and languages must collaborate and share modeling data. Beyond the claim that diagrams are the "common currency," the author believes that it is more precise to say that the *semantics* or meaning conveyed by the diagram is what is the "common currency." Correct semantical interpretations allows a modeler to translate from one modeling language more accurately to another, even if the languages cannot be translated with one-hundred percent carryover of semantics.

An overview of SysML can be covered by presenting the exhaustive set of diagrams that belong to SysML. SysML will not be covered in its entirety; to do so would require covering all modeling elements and their aspects. There are nine types of SysML diagrams,

their taxonomy can be observed in Figure 41; these nine diagrams happen at the lowest indenture of the taxonomy, except for the internal block diagram, which is counted along with its specialized form, the parametric diagram.



The hollow, triangular arrowheads represent a *generalization*. An example of how this is read is an activity diagram "is a type of" a behavior diagram which in turn "is a type of" SysML diagram. This syntax is part of the SysML language and allows us to convey abstractions of classification. Source: Delligatti (2013, 15).

Figure 41. SysML diagram taxonomy.

Delligatti (2013, 15–16) provides the following descriptions for each type of Sys ML diagram:

- The **block definition diagram** (BDD) is used to display elements such as blocks and value types (elements that define the types of things that can exist in an operational system) and the relationships between those elements. Common uses for a BDD include displaying system hierarchy trees and classification trees.
- The **internal block diagram** (IBD) is used to specify the internal structure of a single block. More precisely, an IBD shows the connections between the internal parts of a block and the interfaces between them.
- The **use case diagram** is used to convey the use cases that a system performs and the actors that invoke and participate in them. A use case

diagram is a black-box view of the services that a system performs in collaboration with its actors.

- The **activity diagram** is used to specify a behavior, with a focus on the flow of control and the transformation of inputs into outputs through a sequence of actions. Activity diagrams are commonly used as an analysis tool to understand and express the desired behavior of a system.
- The **sequence diagram** is used to specify a behavior, with a focus on how the parts of a block interact with one another via operation calls and asynchronous signals. Sequence diagrams are commonly used as a detailed design tool to precisely specify a behavior as an input to the development stage of the life cycle. Sequence diagrams are also an excellent mechanism for specifying test cases.
- The **state machine diagram** is used to specify a behavior, with a focus on the set of states of a block and the possible transitions between those states in response to event occurrences. A state machine diagram, like a sequence diagram, is a precise specification of a block's behavior that can serve as an input to the development stage of the life cycle.
- The **parametric diagram** is used to express how one or more constraints—specifically, equations and inequalities—are bound to the properties of a system. Parametric diagrams support engineering analyses, including performance, reliability, availability, power, mass, and cost. Parametric diagrams can also be used to support trade studies of candidate physical architectures.
- The **package diagram** is used to display the way a model is organized in the form of a package containment hierarchy. A package diagram may also show the model elements that packages contain and the dependencies between packages and their contained model elements.
- The **requirements diagram** is used to display text-based requirements, the relationships between requirements (containment, derive requirement, and copy), and the relationships between requirements and the other model elements that satisfy, verify, and refine them.

Figure 42 shows a condensed view of each of these diagrams; depicting some of the basic elements and configurations one might expect to find when viewing each of the respective diagrams.

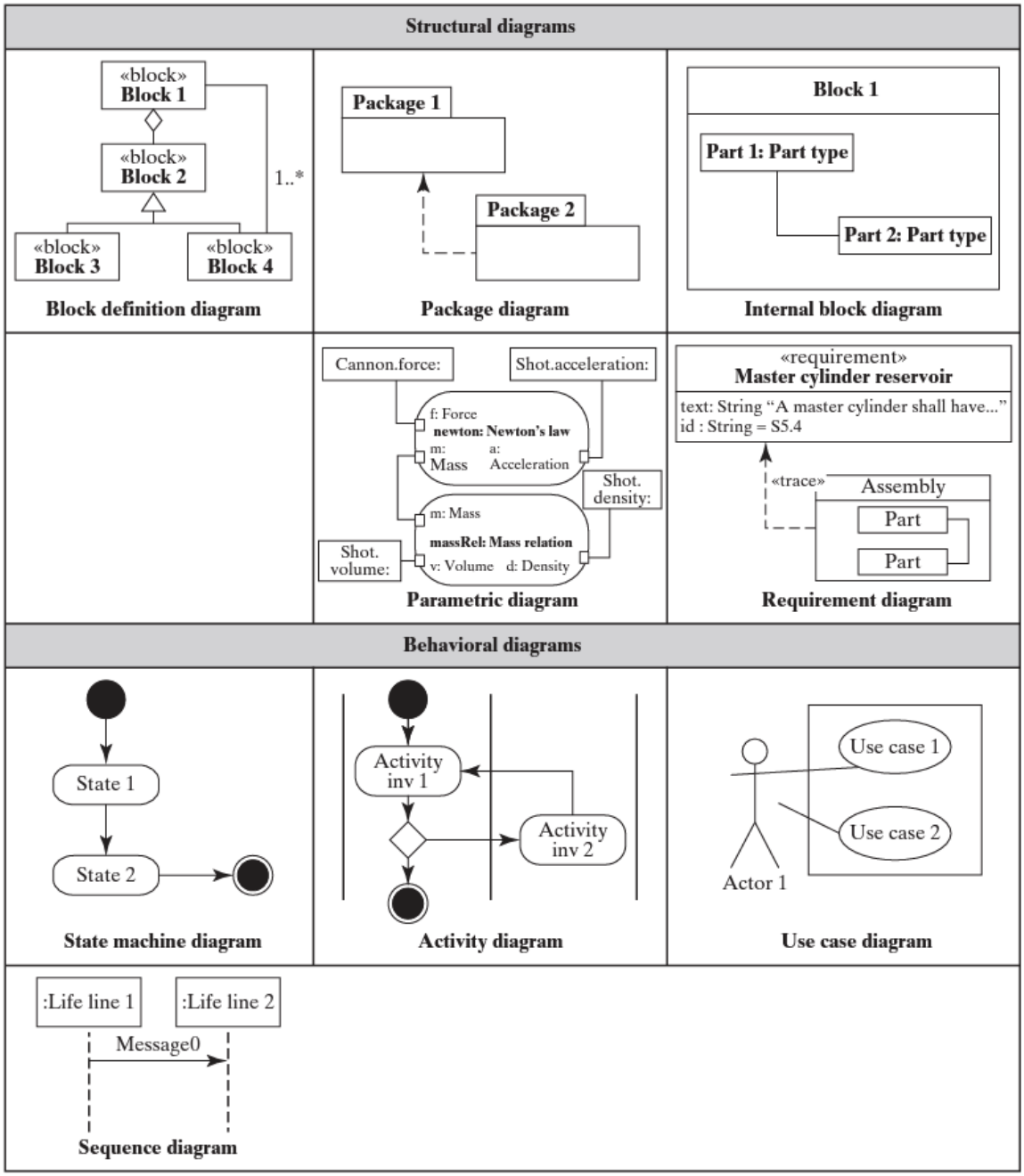


Figure 42. Summary of SysML diagrams. Source: Crawley, Cameron, and Selva (2016, 46).

Figure 43 provides a concise diagram for depicting the main types of logical relationships that are represented in SysML: decomposition/composition (different from aggregation), generalization/specialization, and class/instance logical relationships that

allow definition of elements and re-use of elements through instances or uses of the elements of definition.

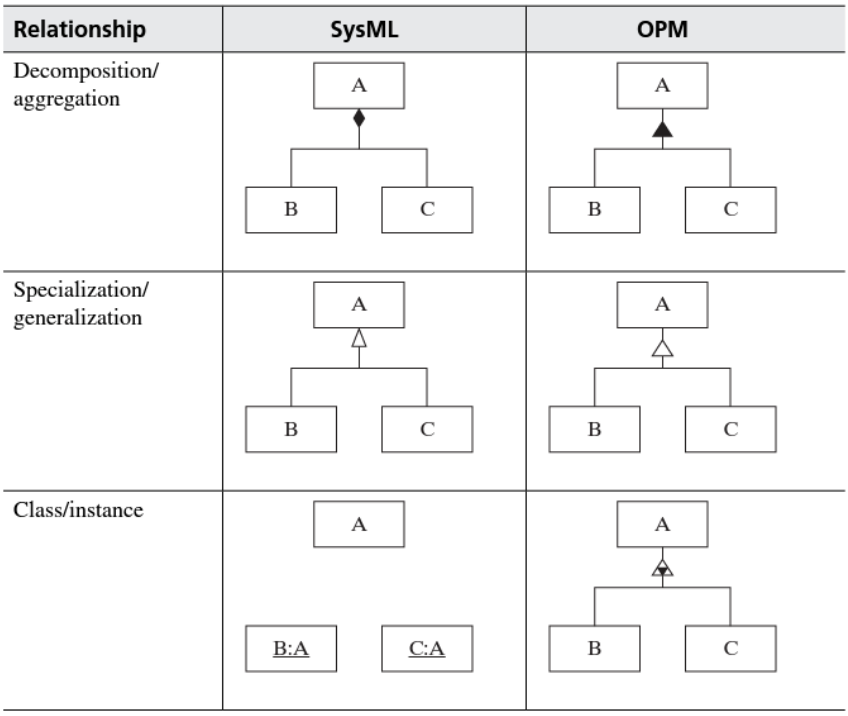


Figure 43. SysML and OPM logical relationships showing hierarchies of composition and abstraction, and logical representation of classes (elements of definition) and instances (elements of usage). Source: Crawley, Cameron, and Selva (2016, 48).

b. SysML Elements

This section will not go into all the details of SysML model elements. The usage of SysML elements will be explained more in the methodology section when describing the modeling decisions that were made. Figure 44 showcases the type of modeling elements that can be used in each of the nine diagram kinds.

Diagram Kind	Allowable Model Element Types
Block definition diagram	package, model, modelLibrary, view, block, constraintBlock
Internal block diagram	block
Use case diagram	package, model, modelLibrary, view
Activity diagram	activity
Sequence diagram	interaction
State machine diagram	stateMachine
Parametric diagram	block, constraintBlock
Requirement diagram	package, model, modelLibrary, view, requirement
Package diagram	package, model, modelLibrary, view, profile

Figure 44. Model element types allowed in each of the nine SysML diagram kinds. Source: Delligatti (2013, 20).

It is worthwhile to briefly introduce and describe the SysML *block* because of its fundamental role in building models and some of the nuances of its applications. As described by Delligatti (2013, 26), "a **block** is the basic unit of structure in SysML. You can use a block to model any type of entity within your system of interest or in the system's external environment." Blocks can be used to represent *definitions* or *instances* of system entities. When a block is used as an *element of definition*, it represents a type of entity, not its instance or *usage*, and defines a set of properties that are common to all instances of the entity defined by the block (Delligatti 2013). A block can be used to represent an instance of a system entity. In this case, the instance is typed by the block that describes the class or element of definition of that element to which that instance belongs too. The instance has all the class properties or attributes that are defined for that block. For example, a block can be used to define a wheel, usually this will be done in a block definition diagram (bdd) potentially showing that a wheel has a composition relationship to a car that is composed of four wheels and other major car components. Then, in an internal block diagram (ibd) of the car, one can see usage of the wheel block—we will see four instances of the wheel block being used to represent each of the four wheels in a car. In this situation it is more

precise to say that those four wheels are *part properties* of the car block. This is not a very rich explanation but introduces some of the nuances of SysML blocks with respect to their application in a model. For more information on detailed descriptions of SysML elements see Delligatti (2013), Holt and Perry (2019), Friedenthal, Moore, and Steiner (2015), and the latest SysML specification document (SysML.org n.d.).

In lieu of detailed SysML model element descriptions and specifications, this section presents the modeling elements being used (generically) in their respective diagrams with comments that identify the modeling elements. Most of the comments/labels are self-explanatory to a degree; others require more explanation. However, the identification of each of these elements will help the reader search for additional information from the references provided or other external references. Only the diagram types that were used as part of this thesis' methodology are presented. These diagram types include the: block definition diagram, internal block diagram, requirements diagram, use case diagram, and activity diagram.

Figure 45 shows a summary of a block definition diagram's modeling elements along with notations that help identify and provide more information about the elements used. The modeling elements showcase configurations that would be used for modeling, even though these are used generically and don't represent any real system.

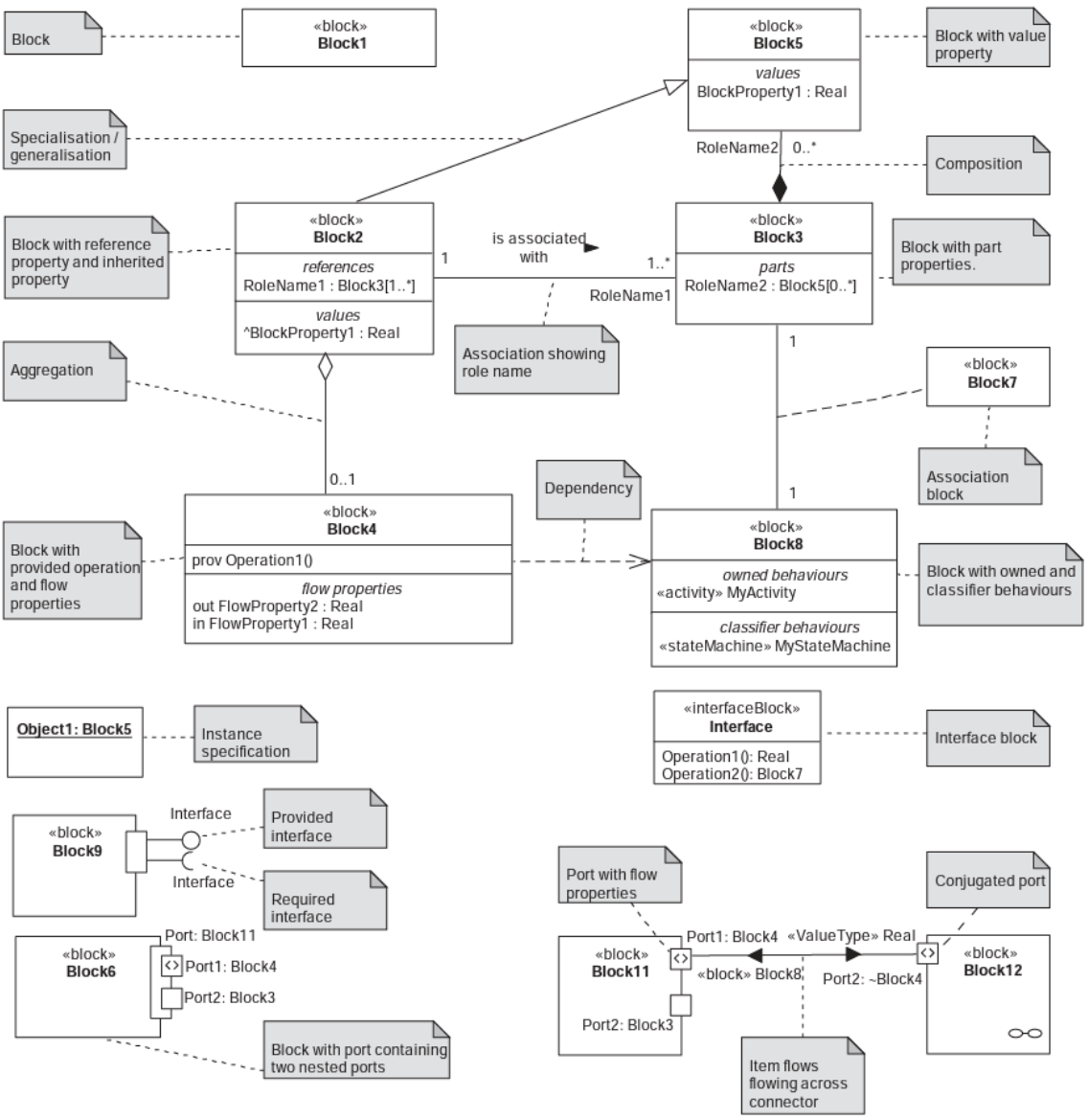


Figure 45. Block definition diagram notation summary. Source: Holt and Perry (2019, 141).

Figure 46 shows a summary of an internal block diagram's modeling elements along with notations that help identify and provide more information about the elements used. The modeling elements showcase configurations that would be used for modeling, even though these are used generically and don't represent any real system. The "Shared Part" notation attached to "Block 11" should also have mentioned that the dashed-line

block represents a *reference property*, belonging to another block elsewhere in the system. All other blocks are internal (belong) to "Block 10."

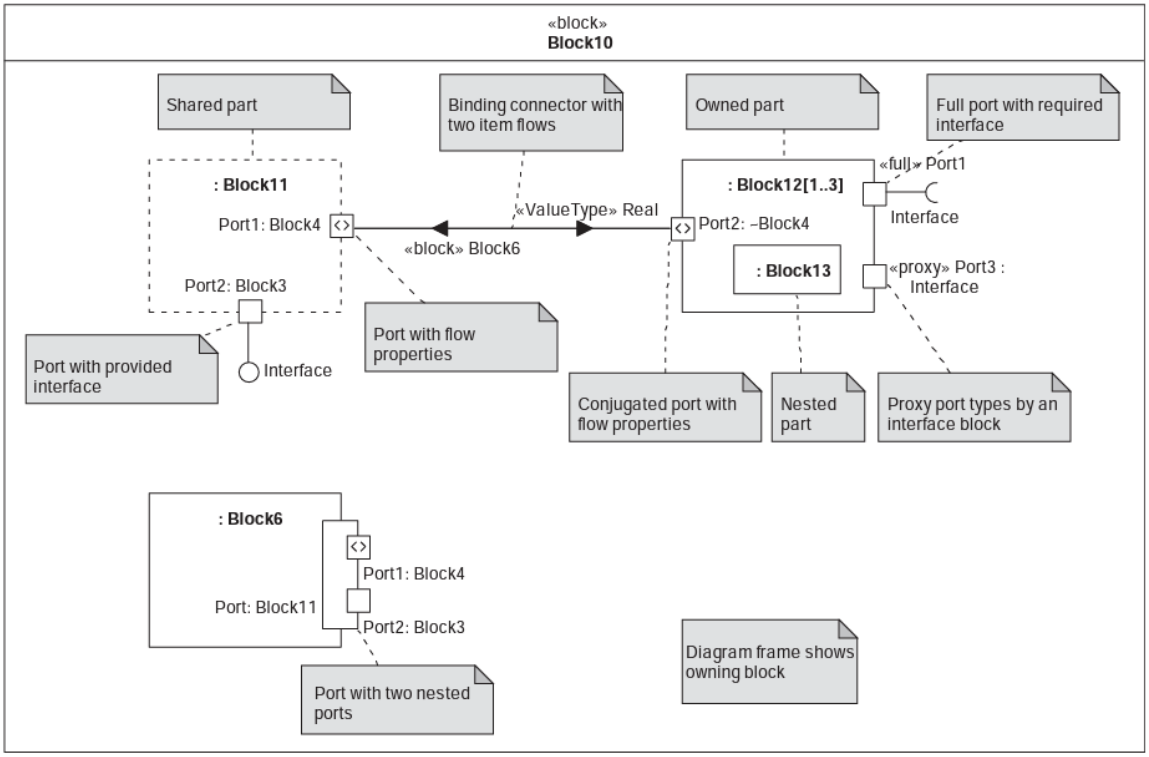


Figure 46. Internal block diagram notation summary. Source: Holt and Perry (2019, 151).

Figure 47 shows a summary of a requirement diagram's modeling elements along with notations that help identify and provide more information about the elements used. The modeling elements showcase configurations that would be used for modeling, even though these are used generically and do not represent any real system.

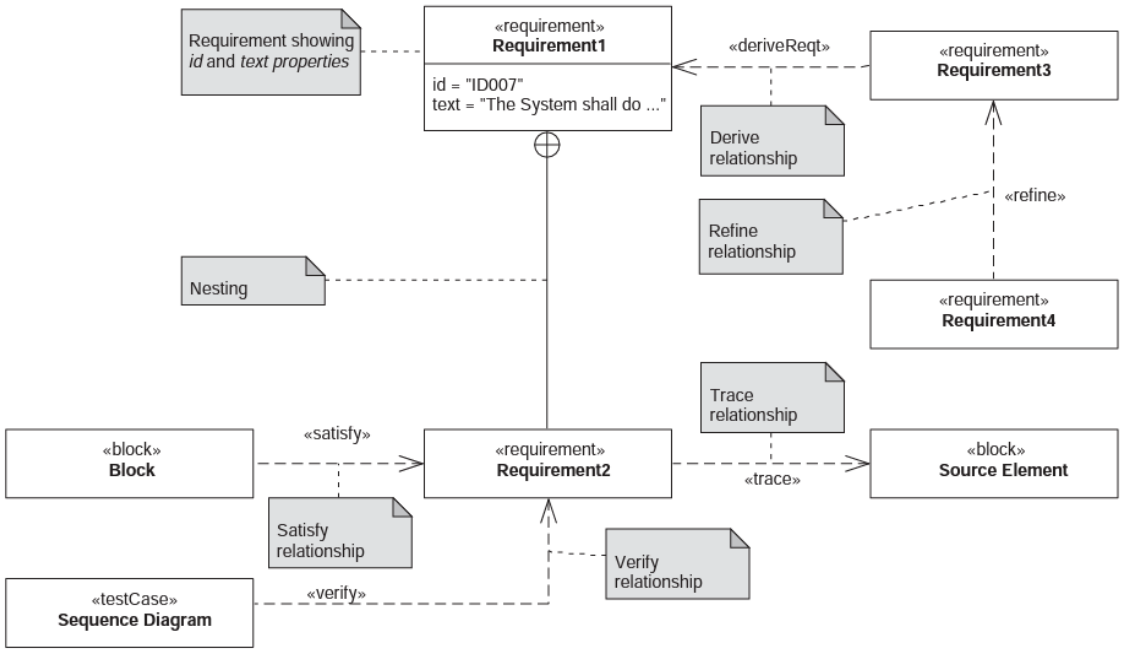


Figure 47. Requirement diagram notation summary. Source: Holt and Perry (2019, 181)

Figure 48 shows a summary of an activity diagram's modeling elements along with notations that help identify and provide more information about the elements used. The modeling elements showcase configurations that would be used for modeling, even though these are used generically and don't represent any real system. It is not obvious from this image that there are two types of flows: control flow and object flow. *Control flow* represents the logic that determines the sequence of actions, while *object flow* shows the inputs/outputs of such actions and the object flow pathways.

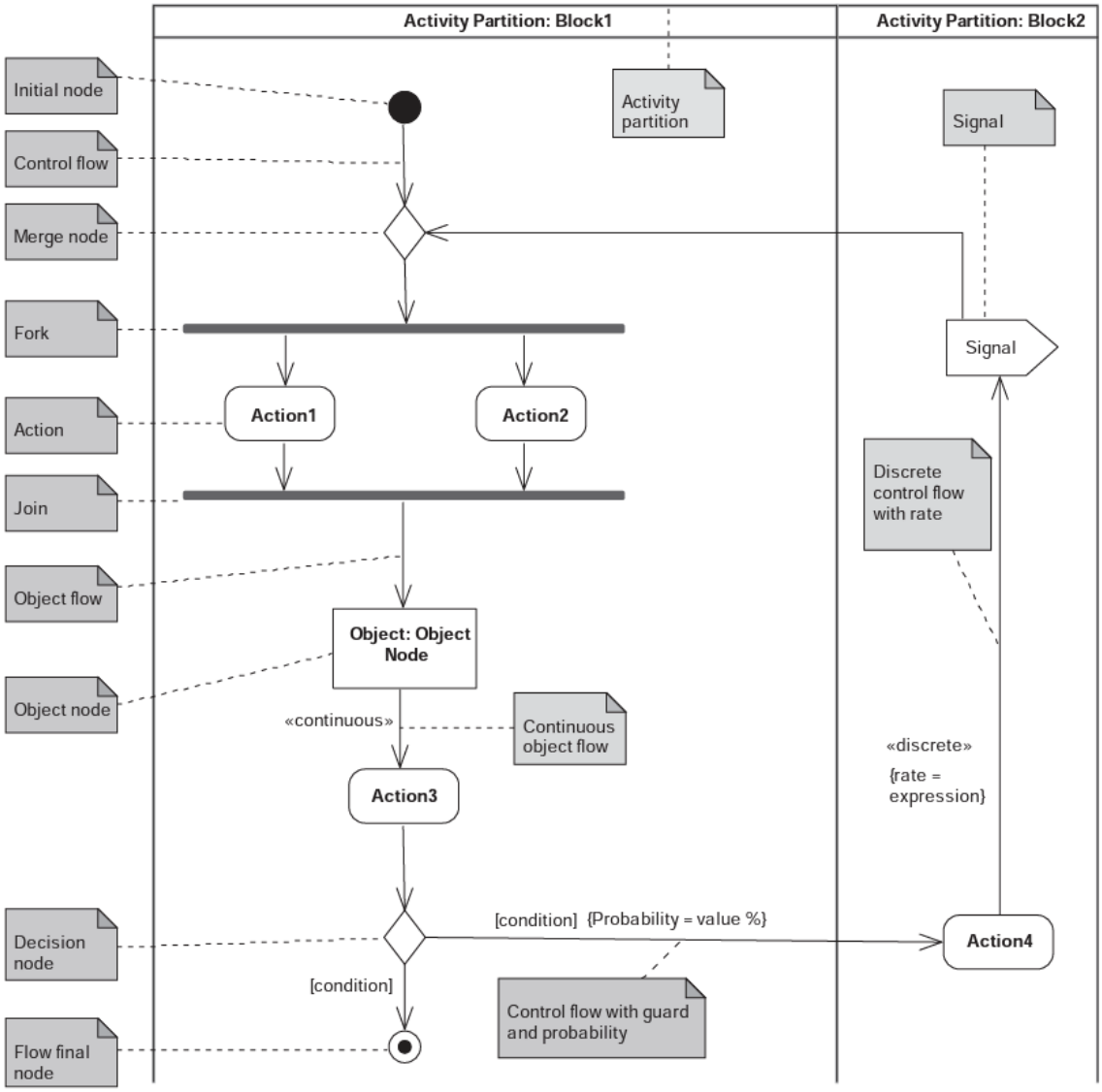


Figure 48. Activity diagram notation summary. Source: Holt and Perry (2019, 213)

Figure 49 shows a summary of a use case diagram's modeling elements along with notations that help identify and provide more information about the elements used. The modeling elements showcase configurations that would be used for modeling, even though these are used generically and don't represent any real system. The *include* use case relationship conveys that when the "*source"* use case ("Use Case 1") is invoked/executed, it requires the included use case ("Use Case 3"), meaning that the included use case will also be executed at some point when the main use case executes (Delligatti 2013, 85). In

contrast, the *extend* use case relationship conveys that if the source/main use case ("Use Case 1") is invoked/executes, the extended use case ("Use Case 4") may be "*optionally"* executed (Delligatti 2013, 87).

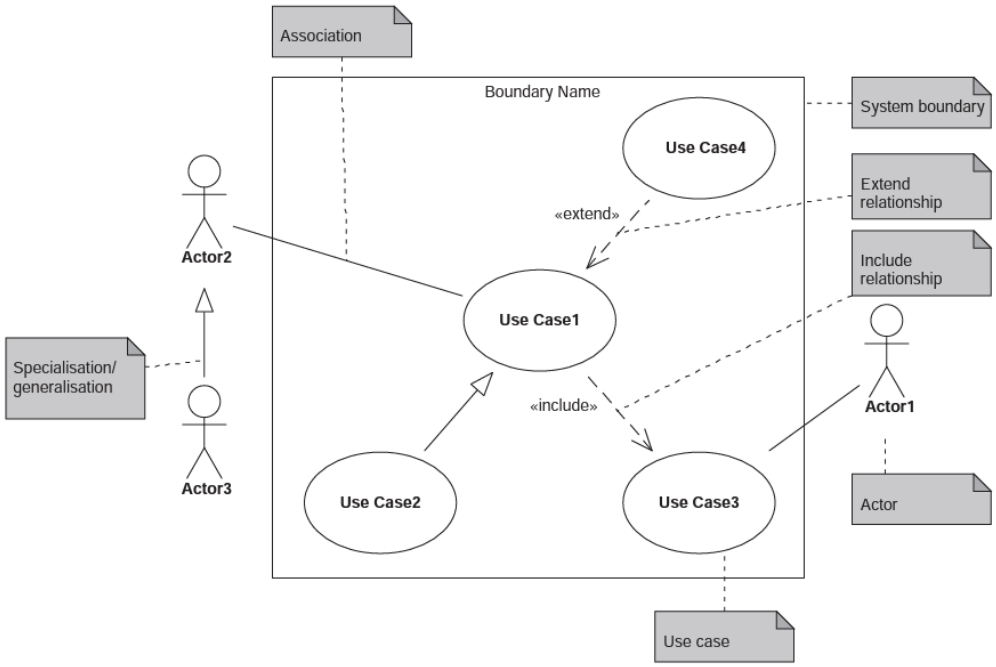


Figure 49. Use case diagram notation summary. Source: Holt and Perry (2019, 222)

For summary notations on the: package diagram see Holt and Perry (2019, 165); parametric diagram see Holt and Perry (2019, 171); state machine diagram see Holt and Perry (2019, 190); sequence diagram see Holt and Perry (2019, 199).

F. THE MAGICGRID FRAMEWORK

The MagicGrid Framework is an MBSE modeling framework that provides an ontological structure of the system architecture and its context, i.e., it identifies and classifies architectural entities and their existential relationships in an organized contextual arrangement. As stated earlier when discussing the Zachman framework and DoDAF, this framework is more structural than it is procedural. The MagicGrid framework does not specify a modeling process or dictates *how* modeling should be done. However, this

framework structure helps guide the modeling process by providing general specifications of *what* needs to be modeled and in what context (grid) of the architecture. Depending on the situation and subject to be modeled, the modeler or modeling team should decide in what logical order to modeling should be done. In reality, it could be concurrent and recursive across various grids, as opposed to going grid by grid.

1. Why MagicGrid?

MagicGrid provides a generic framework that can be applied universally to any type of system. It is not a custom framework, but it is very effective because it ontologically addresses essential concepts of a system's architecture that are common to most or all systems. MagicGrid is also compatible with the development of system technical baselines and their architecture. According to Aleksandraviciene and Morkevicius (2018; 2021), the MagicGrid framework is:

> designed to guide engineers through the modeling process and answer their questions, like "how to organize the model?," "what is the modeling workflow?," "what model artifacts should be produced in each step of that workflow?," "how these artifacts are linked together?," and so on. (Aleksandraviciene and Morkevicius 2021, 12)

MagicGrid indeed facilities answering these modeling questions. The inception of MagicGrid stemmed from the need of various industry sectors, such as "defense, automotive, aerospace" and others, to have an "unambiguous approach for developing system models using SysML" (Aleksandraviciene and Morkevicius 2021, 11).

MagicGrid is easy to follow. *The MagicGrid Book of Knowledge* (Aleksandraviciene and Morkevicius 2018; 2021), developed by Dassault Systemes, has provided a very useful follow-along, tutorial style guide that is easy to follow. Completing the guide teaches the MagicGrid framework, but incidentally, it also teaches beginners how to use the modeling tool, i.e., Cameo System Modeler (CSM) or Magic Systems of Systems Architect (MSOSA). Additionally, both these tools provide MagicGrid version 1 and 2 templates that make modeling using the MagicGrid framework even easier. Another benefit is that this framework is taught at the Naval Postgraduate School (NPS) by Professor Mark Rhoades, therefore creating a growing community of MBSE educated

practitioners who are familiar with this method. Although this framework is easy to adopt without any special training or education.

One of the major benefits of this framework is that it clearly distinguishes the delineation between the *problem domain* and the *solution domain*. By introducing the problem domain, this framework pushes the stakeholder to: (1) define their essential needs, (2) identify what change, outcome, or impact are they hoping to achieve in the world—independent of solution; helps identify system purpose, (3) define *what* system must do to achieve the desired value in solution-neutral way, and (4) identify *concepts* that can satisfy the solution-neutral functional architecture. These tasks and their products are fundamental to addressing the *right* problem and providing the design specifications of the *right* system. This is one of the primary reasons this framework was selected. The MBSE subject of this modeling effort is the Cost Readiness Impact Model (CRIM). Before any modeling or architectural design was formally done, CRIM had already been implemented. Meaning that there was already an existing system. In this situation, most modeling efforts in NAVAIR focus on capturing *what exists*, primarily the *physical architecture* or the *product baseline*. However, very little emphasis is placed on capturing the behavior or functions of the system components or the emergent functions of the system. Much less emphasis is placed on capturing the *solution-neutral* functional architecture that addresses the essence of *what* the system needs to do not *how* the system must do it. Addressing the fundamental needs, and solution neutral functions is critical for a valid system to be developed and for tracing and proving its validity. The MagicGrid framework provides spaces for reverse engineering CRIM and capturing its higher-level architectures and system baselines.

There are two versions of the MagicGrid framework, version 1, and version 2, shown in Figures 50, and 51, respectively. Version 2 explicitly adds the Safety & Reliability pillar to the ontology. This additional pillar may prove very useful for manned systems or systems in general that must operate safely and reliably lest they pose a risk. Safety and Reliability are emergent attributes of the system; they may be referred to as the "-ilities." In MagicGrid version 1, these still may be addressed through safety and reliability requirements and parametric specifications that are "shuffled" or "baked-in" into other pillars. However, as "-ilities" go, these may be the most worthwhile to factor out the

other pillars and be consolidated into their own, very visible pillar. MagicGrid version 1 does include some proposed extensions of MagicGrid, such as "specialty engineering," "integrated testing," and "analysis" (Aleksandraviciene and Morkevicius 2018). An additional pillar that may be potentially useful in the MagicGrid Framework would be a "Validation & Verification" pillar. Validation is more strongly related to the "Problem" domain rows, especially at the system level—recalling from Crawley et al. (2016), who presents the concept of the (functional) value pathway, internal to the system, and the concept that system delivers its ultimate function and value through the system boundary. Verification has to do with correctness of the implemented design with respect to its *design-to* and *build-to*, *code-to*, or *buy-to* specifications and would likely be present in the "Solution" domain rows. If the subject being modeled was an M&S subject the V&V pillar would be the de facto "M&S V&V" pillar.

	PILLAR						
DOMAIN			**Requirements**	**Behavior**	**Structure**	**Parameters**	**Specialty Engineering**
	Problem	**Black Box**	B1-W1 Stakeholder Needs	B2 Use Cases	B3 System Context	B4 Measurements of Effectiveness	
		White Box		W2 Functional Analysis	W3 Logical Subsystems Communication	W4 MoEs for Subsystems	
	Solution		S1 System Requirements	S2 System Behavior	S3 System Structure	S4 System Parameters	**Integrated Testing**
			SS1 Subsystem Requirements	SS2 Subsystem Behavior	SS3 Subsystem Structure	SS4 Subsystem Parameters	
			...	...	...	...	**Analysis**
			C1 Component Requirements	C2 Component Behavior	C3 Component Structure	C4 Component Parameters	
	Implementation		I1 Physical Requirements	Software, Electrical, Mechanical			

Figure 50. The MagicGrid Framework v.1. Source: Aleksandraviciene and Morkevicius (2018, 11).

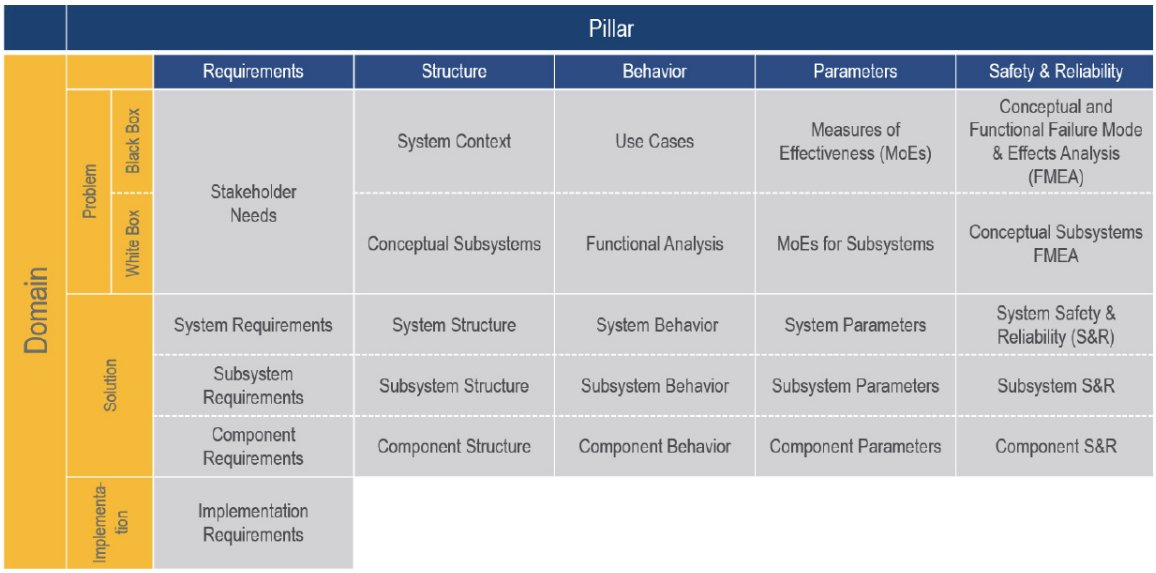


Figure 51. The MagicGrid Framework v.2. Source: Aleksandraviciene and Morkevicius (2021, 12).

The MagicGrid Framework is used extensively in the methodology. Therefore, this section will not cover further details regarding the MagicGrid Framework. For anyone interested in the MagicGrid Framework, either or both MagicGrid *Book of Knowledge* guides are recommended. Also, if someone is beginning to model, these guides are equally useful in introducing the usage of the modeling tools, i.e., Cameo Systems Modeler (CSM), or Magic Systems of Systems Architect (MSOSA).

G. MODELING AND SIMULATION

This section provides a general overview of models and of simulation models. The term "model" is quite general and can be an ambiguous term without any other context, like what is it a model of? Or what is the model for? For those without much knowledge of models, when they hear the term "model," they could be conjuring a fair amount of mysticism in their mind thinking that the model is some sort of magical black box, or they make a significant mental simplification of the model to the point where they generalize their perception of what the model is so much that they may be indifferent about it in comparison to other different models.

A main objective of this section is to create a rudimentary mental mapping of models that may come to mind when the term "model" is mentioned. This section will later focus on descriptive (schematic) models, and simulation models. The descriptive models are covered only briefly because there is another chapter section covering MBSE models—which are descriptive, and this type of model was used for this thesis research methodology. Simulation models are covered because the subject under study, the Cost Readiness Impact Model (CRIM) employs two types of stochastic simulation models: one is the "Driver Tree" simulation model, and the other one is a multi-echelon, multi-indenture discrete event, Monte Carlo simulation model, which uses Systecon's SIMLOX commercial simulation software. The Driver Tree model is very similar to a reliability (fault) tree, where each upper tree level value is equivalent to a (linear) parametric equation of a combination of some of the lower tier's values—based on the Naval Aviation Maintenance Program (NAMP) (Commander Naval Air Force 2022). The author cannot say with complete certainty that all Driver Tree equations are linear. This detail is not essential for answering the thesis question, but having this knowledge would support more concretely showcasing the ability of this thesis's proposed M&S VV&A framework and method to capture such details.

The descriptive models that apply to this thesis research involve mainly schematic models. Most of the data gathered by the author involved documents that described this thesis research subject, CRIM, in a graphical way. The author synthesized the information provided by these schematic models and even textual documentation—in a sense the "natural" language is a model too.

1. Taxonomy of Model Types

Simply stated, a *model* is an abstraction of some real-life subject. Buede and Miller (2016, 70) state that "a *model* is any incomplete representation of reality." The subject is usually something that exists physically, but a model can even be something meta-physical, like a concept. The model may only cover specific components of the subject and may only be suitable for narrow applications. To refine the notion of what a model is, Weilkiens

(2015, 27) provides a few simple, but fundamental ideas that can be universally applied to most or all models.

- Mapping—A model is a mapping of something else [the real-life subject of interest].
- Reduction—A model only reflects parts of the original thing.
- Pragmatic—A model fulfills a specific function and is used in place of the original for this purpose. (Weilkiens et al. 2015, 27)

With respect to mapping and reduction, some subject parts are more significant than others. Depending on parts level of criticality with respect to physical features (for a physical model) or with respect to the real-life system behavior (for a logical or analytical model), the modeling requirements may be set to meet the abstraction needs of the model so that the model meets those needs and fulfils its purpose. Weilkiens (2015) offers another insightful notion with respect to mapping and reduction and their relation to *abstraction*.

> The mapping and reduction features imply another often-mentioned feature of models: abstraction. *Abstraction* is the process to reduce the information about a concept to the relevant parts for a particular purpose. An abstraction is the result of an abstraction process. (Weilkiens et al. 2015, 28)

Perhaps models can be classified into two main general categories, *qualitative* and *quantitative*. The next classification tier could be borrowed from Kossiakoff et al. (2020, 264), who classifies models in three main categories: "schematic models," also known as "descriptive models," "mathematical models" and "physical models." This thesis literature survey does not make a great effort to synthesize a well-defined ontology or even a taxonomy of models. However, a simple taxonomy example is all that is needed to illustrate the types and variety of recognizable models—this also provides a contextual basis from which to make model type comparisons. Table 5 shows such a taxonomy which heavily borrows from Kossiakoff (2020).

Table 5. Taxonomy of Model types (example)

Top-Level Classes	Kossiakoff's main categories	Kossiakoff's sub-categories.
Qualitative	Schematic	Cartoons
		Architectural models (artistic sketches to technical blueprints
		System Block Diagrams
		System Context Diagrams
		Functional Flow Block Diagrams (FFBDs)
		Integrated DEFinition Language 0 (IDEF0) Diagrams
		[Logical Simulation (e.g. activity diagram flow simulation or state-machine simulation)] (added by author)
	Physical	Scale Models
		Mock-ups
		Prototypes
Quantitative	Mathematical (Numerical)	Approximate Calculations
		Elementary Relationships (like physics equations or geometric relations)
		Statistical Distributions
		Graphs
		[Numerical computation Simulation] (added by author)

a. Simulation Models

Kossiakoff et al. (2020) provide a suitable, general description for a simulation:

> Simulation is a general type of modeling that deals with the dynamic behavior of a system or its components. It uses a numerical computation technique for conducting experiments with a software model of a physical system, function or process….
>
> There are many different types of simulations, and one must differentiate static from dynamic simulations, deterministic from stochastic (containing random variables), and discrete from continuous. (272)

Kossiakoff et al. (2020, 272) group simulations into four categories: "operational, physical, environmental and virtual reality simulation." Kossiakoff et al. (2020) also provide examples for each type of simulation:

1. Operational Simulation examples: games [including computer-aided], system effectiveness simulation, mission simulation.
2. Physical Simulation [physics-based]: Aircraft, Automobiles, and Space Vehicles; Hardware-in-the-loop; engineering.
3. Environmental Simulation: Mechanical stress testing, crash testing, wind tunnel testing.
4. Virtual reality simulation: Spatial simulations, video games, battlefield simulation.

b. Descriptive Models

A *descriptive model* is qualitative by nature, meaning that it captures the descriptive qualities of the real-life subject. Descriptive qualities of a subject can be characteristics such as the form or elemental structure of the subject, logical behavior, such as the subject's functions, or even other attributes such metrics or values the subject may possess. When a descriptive mode is "interrogated" or used, the descriptive model conveys to the user this information. Abstraction helps distill and simplify the subject's parts and their fundamental identities and qualities so that they can be analyzed more easily without sacrificing losing the essential nature of the real-life subject. That is *if* the model is valid or built so that it accurately represents reality, the domain of interest, and achieves its intended use or purpose.

c. Qualitative and Quantitative Model Validity (Brief discussion, related to M&S VV&A)

As far as validation is concerned, Buede and Miller (2016, 74) state that, "for a qualitative model, conceptual validity is the most important," this is in contrast to a quantitative model where *operational validity* "is the key—does the model's output behavior represent that of the real world for the questions being asked?"

To continue contrasting: data validity, for a *quantitative model*, addresses whether "the appropriate inputs were employed in building, testing, and using the model" (Buede and Miller 2016, 74) For a *qualitative model*, this is different. Buede and Miller (2016, 74–75) state that "data validity, for a qualitative model, addresses whether the right individuals were

involved in creating the model and whether they obtained access to the best set of information about the real world during the creating process."

2. Quantitative Modeling Uncertainty and Errors

The ability to build accurate models is hindered by uncertainty, abstraction/simplification and errors introduced along the way. This section seeks to cover a few basic concepts on uncertainty and error to provide some context and understanding of these elements.

a. Systems Theory Uncertainty Philosophy

This brief section is meant to introduce a slightly differing philosophy on the classification of model uncertainties. Gansch and Adee (2020) state that:

> Previous work only distinguishes between aleatory and epistemic uncertainty. However we consider that in the field of highly automated driving vehicles, a further distinction into ontological uncertainty is valuable. (Gansch and Adee 2020)

The author very much appreciates the classification of Gansch and Adee and prefers it over historical descriptions of uncertainty, but to avoid confusion and to maintain consistency with the previous definitions of uncertainty, the author tries to remain within the established definitions of epistemic and aleatory uncertainty. However, the descriptions given by the author can be easily translated to match those of Gansch and Adee (2020).

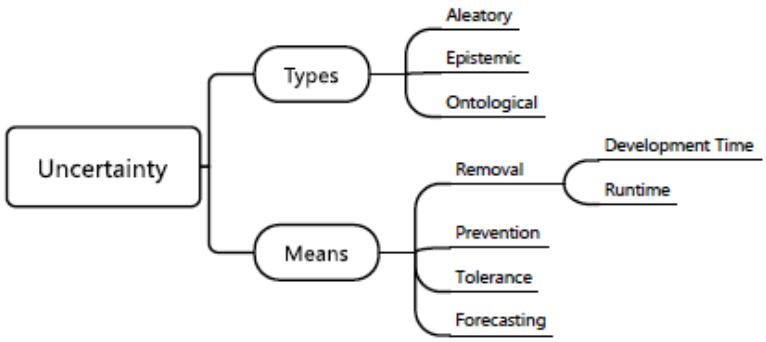


Figure 52. Uncertainty taxonomy and means of uncertainty mitigation. Source: Gansch and Adee (2020, 1348).

Gansch and Adee (2020) introduce the concept of "ontological uncertainty," which appears to split from epistemic uncertainty so that linguistic ambiguity can be removed and give a more precise meaning to identify the uncertainty entity as ontological or epistemic. "Ambiguity exists linguistically due to our inability to express a meaningfully declarative sentence. Hence, we may have ambiguity with the truth condition of a proposition" (Modarres 2016, 2). Figure 52 shows Gansch and Adee's (2020) uncertainty taxonomy; the "Means" describe ways uncertainty can be mitigated or reduced.

Gansch and Adee (2020, 1348) state that "segregating the epistemic and ontological uncertainty in a model can be challenging, as they constitute blurred boundaries. A general rule of thumbs [sic.] is a distinction between model accuracy (epistemic) and model correctness (ontological)."

The following are definitions and further descriptions of each type of uncertainty, as defined by Gansch and Adee (2020):

(1) Ontological Uncertainty:

> Ontological uncertainty can be defined as a condition of complete ignorance in the model of a relevant aspect of the system. ….
>
> This can also be referred to as the unknown-unknown, the state of we do not know that we do not know. (Gansch and Adee 2020, 1348)

(2) Epistemic Uncertainty:

> Epistemic uncertainty is associated to the lack of knowledge about the system model and the inexact encoding of physical system to models. …
>
> An important aspect of epistemic uncertainty is that we lack in knowledge but we are aware of it. This can also be referred to as the *known-unknown* of the model. Another characterization of epistemic uncertainty can be attributed to conditional entropy, i.e., the difference of information contained in the model with respect to the physical system. A model is an abstracted encoding of a system, hence a mere projection of reality. Due to abstraction, not every detail is encoded in the model and thereby conditional entropy is induced. (Gansch and Adee 2020, 1347)

(3) Aleatory Uncertainty:

Aleatory uncertainty can be regarded as randomness of a process represented by a system model…

Aleatory uncertainty is considered irreducible for a given choice of a probabilistic model and is quantified by probabilistic distributions. (Gansch and Adee 2020, 1347)

3. Uncertainty Classification Example

To better understand the definitions of uncertainty provided, Gansch and Adee (2020) provide an example. They present a physical system composed of two planets, which have been simplified into point masses. Gansch and Adee (2020) then propose two ways of modeling this system: a deterministic model and a probabilistic model. Figure 53 shows a representation of this example.

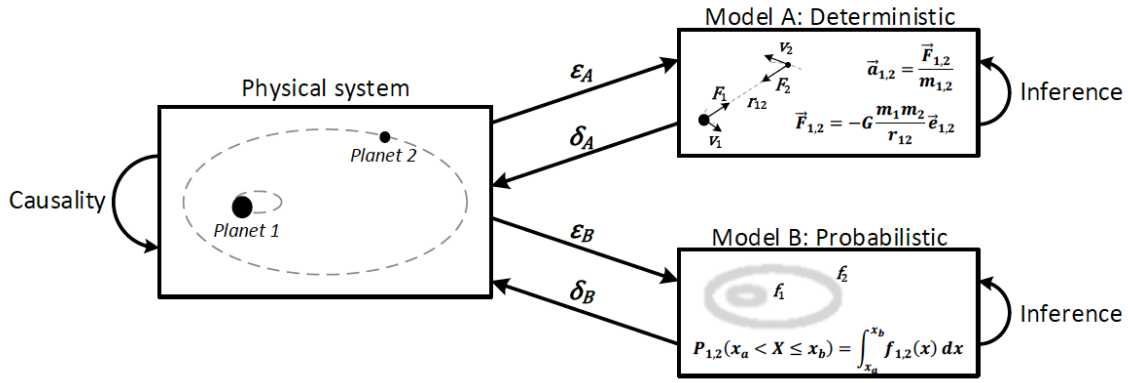


Figure 53. Modeling uncertainty example. Source: Gansch and Adee (2020, 1347).

Table 6 has been constructed by the author to summarize and organize Gansch and Adee's (2020) example.

Table 6. Modeling uncertainty example: discussion. Adapted from Gansch and Adee (2020).

Model A: Deterministic Model	Two planets are considered and simplified to point masses. Newtons laws of gravity are considered. The deterministic model represents the causality of motion of the bodies. A future state of the system can be determined with the given the starting conditions (initial position and velocity).
Model B: Probabilistic Model	Two planets are considered and simplified to point masses. A probabilistic model is built by repeated observation of the systems positions. A future state of the system can be inferred from the obtained probability density functions of the observations.
NOTE	Models can be a mixture of deterministic and probabilistic elements.
Epistemic Uncertainty	For idealized point masses, the model is completely accurate and there is no uncertainty in the model, it is solely deterministic. Real planets are more complex than point masses. If the planets are depicted with a body, the laws of physics still hold, but a model with idealized point masses do not accurately describe the complex physical system anymore—one may ask: *to what degree?* or *to what purpose?* "A more accurate model based on a spatial integral over the attracting forces would allow a more accurate description and reduce the epistemic uncertainty. However, this would require complete information of the actual mass distribution, which might not be available" (Gansch and Adee 2020, 1347).
Aleatory Uncertainty	The probabilistic model provides a probability that the system is in a certain given spatial frame, given the conditions. However, the exact configuration of the system cannot be known with one-hundred percent certainty. The probabilistic model can also have epistemic uncertainty based on how many observations were made. The probabilities are calculated based on past observations. "The gap between the actual and observed probabilities represents the epistemic uncertainty of the probabilistic model. …Hence, our knowledge increases and the epistemic uncertainty decreases with every observation" (Gansch and Adee 2020, 1347). "With an infinite amount of observations, the exact probabilities to find either of the two bodies withing a spatial fame can be inferred from the obtained probability density functions" (Gansch and Adee 2020, 1347). Gansch and Adee (2020) probably assume that the system is in an ideal "steady state" or the approximation/simplification of treating this system as a steady state system is exact enough to get exact probabilities of inferring the future system state.
Ontological Uncertainty	The models were built around the assumption of two planets. If there were *three* planets in the system, the observation of the system would contradict "the prediction by the models due to the influence of a third planet" (Gansch and Adee 2020, 1348). It is assumed that Gansch and Adee (2020) are referring to the deterministic model; theoretically, the probabilistic model should capture the overall observed system behavior, which includes the unknown third planet. In this example, the deterministic physics-based model would need to include the properties and effects of a third planet. **Fun Fact**: The deterministic model in this case is based on Newton's theory of gravity. Because of Newton's law of gravity, during the 1800s, scientists believed that there must be another planet (planet Vulcan, which orbits closely to the sun and was proposed by French Mathematician Le Verrier) that accounted for Mercury's unexplained orbital path. Previously, in 1846, Le Verrier predicted Neptune's existence mathematically before it was observed—Le Verrier instructed German astronomer Johan Gottfried Galle where to look based on his calculations (Ruskin 2017). Neptune's existence and gravitational effects explained the puzzling effects of Uranus's orbit (Ruskin 2017). In November of 1915, Albert Einstein "demonstrated that the curvature of spacetime, and not a hidden planet, convincingly accounted for Mercury's odd orbit. Ruskin (2017) quotes Thomas Levenson, the head of MIT's science writing program: 'The sun with its great mass creates its dent in space-time…until, as Einstein finally captured in all the abstract majesty of his mathematics, the orbit of [Mercury] precesses away from the Newtonian ideal" (Ruskin 2017).

a. Descriptive Model's Uncertainty

Primarily, a qualitative model should seek to avoid modeling omissions due to any *epistemic* or knowledge gaps; according to Gansch and Adee (2020), this would equate or extend to *ontological* gaps. Such gaps may occur when critical or even relevant parts of the real-life subject are missed because modelers did not have the knowledge. Figure 54 illustrates the situation that a modeler may face with respect to knowledge gaps. This is strongly related to the data validity concepts of a qualitative model that were stated previously in the section above, "Qualitative and Quantitative Model Validity (Brief discussion, related to M&S VV&A)."

- "Unknown-Unknown": A modeler may not know that a sub-domain of overall subject of interest exists, and therefore also know any details about it—represented by the term "unknown-unknown." This description fits Gansch and Adee's (2020) definition for ontological uncertainty.

- "Unknown-Known" :Of the existing overall knowledge of the subject of interest, there may exist some relevant or critical information. It is somewhere, perhaps know by some subject matter expert or captured in some document or database, but the modeler has no awareness of it, and this information is obviously not captured in the model. This type of knowledge could be represented by the term "unknown-known," quite an oxymoron, but it is appropriate for the description given here. This description could be used generally to describe the situation where data or information *actually* exists, but from our outward perspective, we don't know—it must be discovered or brought to our attention.

- "Known-Unknown" : This term represents the situation where the modeler may know that a part of the real-life subject exists, but the modeler's level of knowledge is not sufficient to model the necessary details of that part. This is analogous to a modeler having an abstract *black-box* understanding of what they are supposed to model, but the model's intended use requires that the modeler be able to model the *white-box* details (to the level

required). Gansch and Adee (2020) would classify this uncertainty as *epistemic uncertainty*, where all the details of the system are not fully *encoded* into the model.

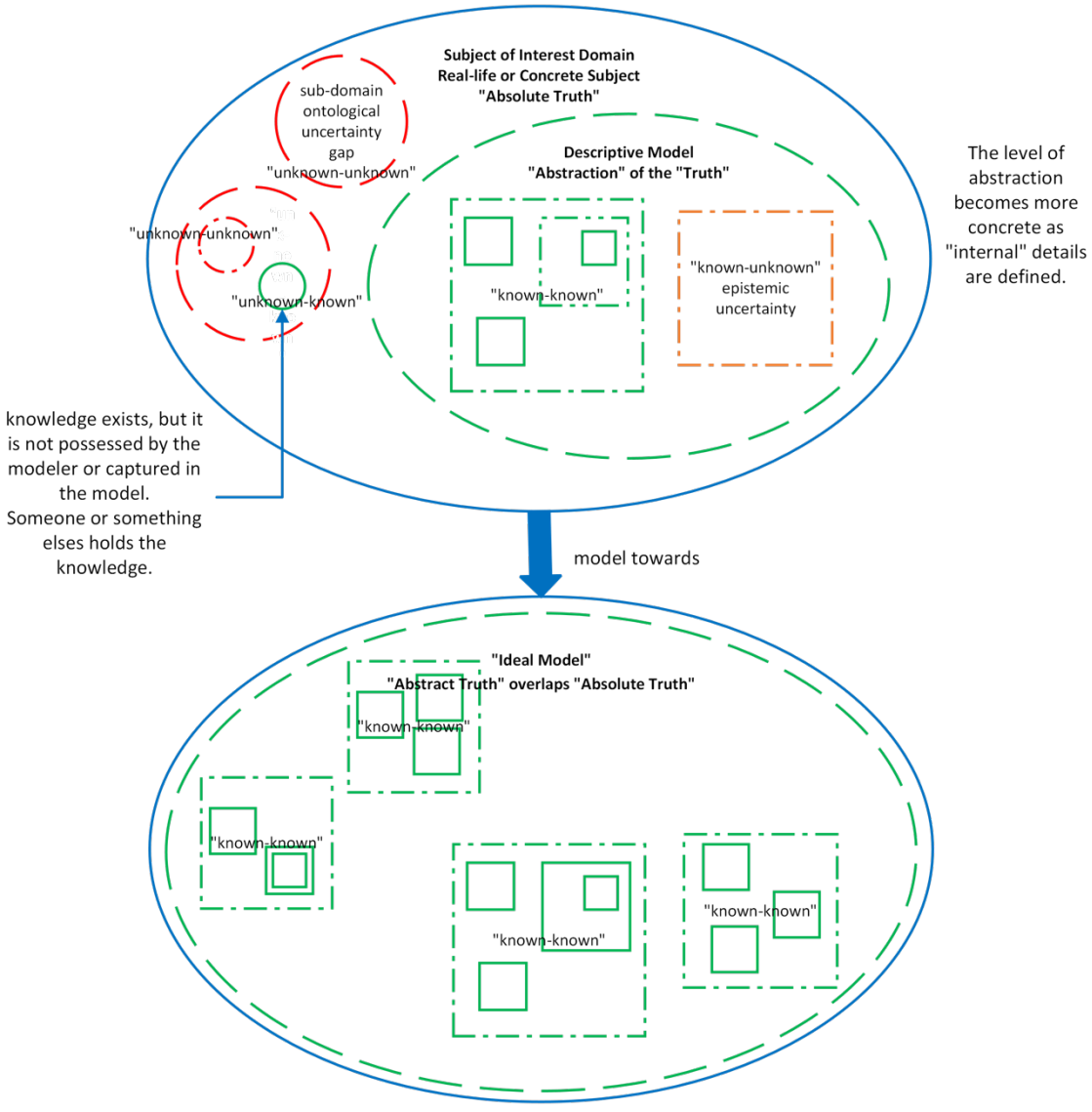


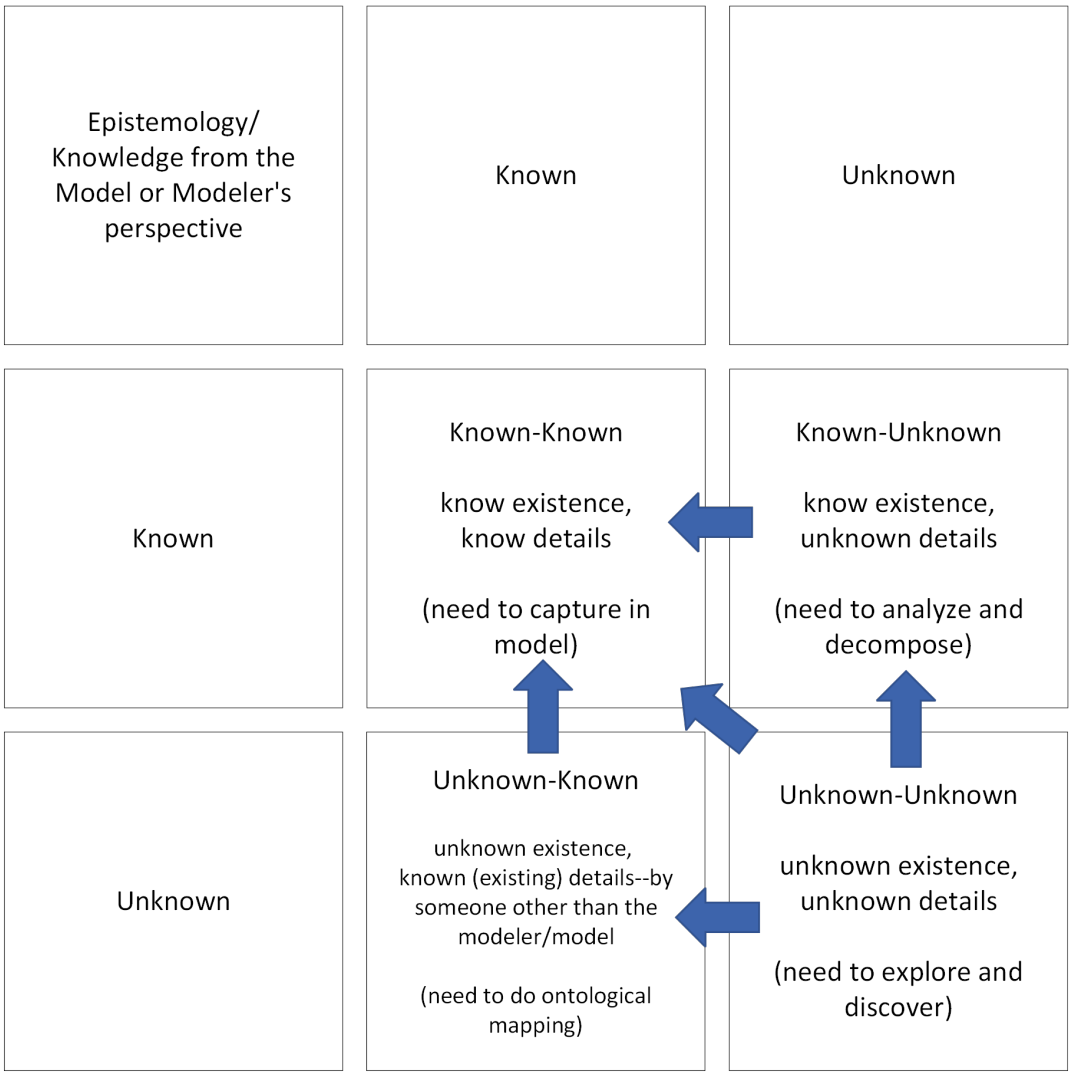
Figure 54. Conceptual representation of a model or modeler's knowledge about the system.

Figure 55 is a complementary figure that organizes the epistemic concepts presented here.

The "known-known," "known-unknown," and "unknown-unknown" terms are adapted from former United States Secretary of Defense Donald Rumsfeld's (February 12, 2002) news briefing response to the "lack of evidence linking the government of Iraq with the supply of Weapons of mass destruction to terrorist groups," which have led some to reference this matrix structure as the "Rumsfeld matrix" (Wikipedia 2023a). The following quote is from Rumsfeld's DOD news Briefing response:

> Reports that say that something hasn't happened are always interesting to me, because as we know, there are known knowns; there are things we know we know. We also know there are known unknowns; that is to say we know there are some things we do not know. But there are also unknown unknowns—the ones we don't know we don't know. And if one looks throughout the history of our country and other free countries, it is the latter category that tends to be the difficult ones. (Wikipedia 2023a)

Rumsfeld may have brought recent attention to this concept, but the idea of "unknown unknows was created in 1955 by two American psychologists, Joseph Luft and Harrington Ingham in their development of the Johari window," which is an analysis technique long used by national security and intelligence professionals; these terms and their concepts are also used in project management and strategic planning disciplines (Wikipedia 2023a).



Arrows show the direction knowledge should flow to support successful or valid modeling efforts.

Figure 55. Model or modeler's epistemic uncertainty management concept mapping, adapted from the "Rumsfeld matrix" and the Johari window.

These terms are not widely known or used, but the author has seen an uptick of their use among the NAVAIR community. For example, one instance was used to address project risks, another instance to refer to system design requirements and architecture, and another instance to categorize the uncertainty of program office-air (PMA) funding planning. However, sometimes these terms are defined and used poorly, and they cause as much confusion as they do enlightenment. The original Johari window "is a technique

designed to help people better understand their relationship with themselves and others" (Wikipedia 2023b). The Johari window is based on a different perspective than the matrix provided in Figure 56, with the only similar quadrant being the "blind spot" quadrant where information is not known to self but known to others—this is the unknown-known situation described earlier.

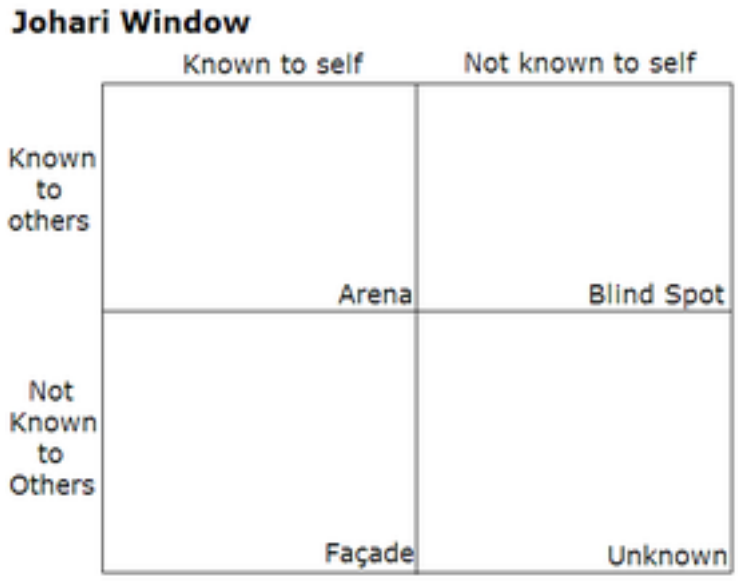


Figure 56. The Johari Window. Source: Wikipedia (2023b).

b. Quantitative Model Uncertainty—Aleatory and Epistemic Uncertainty

This section considers a stochastic model that contains aleatory or probabilistic uncertainty in its inputs, but the model structure and processes are deterministic. Figure 57 shows a taxonomy for classifying uncertainties. Beisbart and Saam (2019, 120) provide a generic term for *uncertainty*, stating that "an *uncertainty* is an imprecision in a value," but for the purposes of better understanding simulation models, definitions for *epistemic uncertainty*, and *aleatory uncertainty* are provided and some characteristics regarding each is discussed.

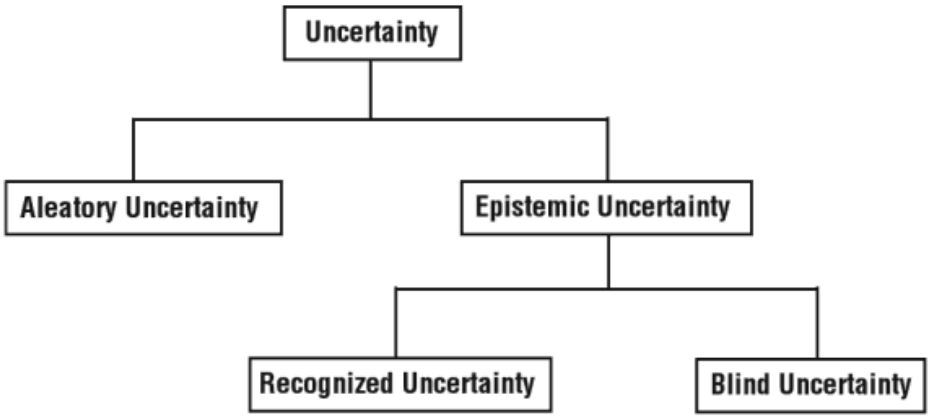


Figure 57. Taxonomy of uncertainties by their classification. Source: Oberkampf and Roy (2010, 54).

Epistemic Uncertainty is "uncertainty due to our lack of knowledge of the system, surroundings, environments, and scenarios" (Oberkampf and Roy 2010, 20). Epistemic uncertainty is also referred to as "reducible uncertainty, knowledge uncertainty, and subjective uncertainty" (Oberkampf and Roy 2010, 53). Oberkampf and Roy (2010, 53) state that "the fundamental source of epistemic uncertainty is incomplete information or incomplete knowledge of any type that is related to the system of interest or its simulation."

Epistemic uncertainty may lead to modelers not completely modeling all the real-life subject or system's logical processes and assumptions. Omission of such modeling components may lead the model to not behave like the real system and therefore not be an accurate representation in its form, behavior, or its results.

The second type of uncertainty is that of *aleatory uncertainty*, which is "uncertainty due to inherent randomness in the system, surroundings, environments, and scenarios" (Oberkampf and Roy 2010, 20). Aleatory uncertainty is also referred to as stochastic uncertainty, variability, inherent uncertainty, uncertainty due to chance, and Type A uncertainty" (Oberkampf and Roy 2010, 52). Oberkampf and Roy (2010, 52) state that "the fundamental nature of aleatory uncertainty is randomness, e.g., from a stochastic process." With respect to how aleatory uncertainty is used to describe characteristics of the system and its values or metrics, Oberkampf and Roy (2010, 52) state that "the mathematical

representation or characterization, most commonly used for aleatory uncertainty is a probability distribution."

Oberkampf and Roy (2010, 53) state that "epistemic uncertainty is a property of the modeler or observer, whereas aleatory uncertainty is a property of the system being modeled or observed." Epistemic uncertainty is carried through from the modeler and "built" into the model through omission of the elements or processes that are present in the real system but absent in the model. Observation of the real system allows us to identify random processes of the system that yield random result values.

Uncertainty is related to a model's nondeterministic nature. "A model is deterministic if, for a given entry, the results are always identical," and "a model is stochastic or random when its results are apparently determined by one or more random variables, which represent an uncertain process. In other words, the results of the simulation will vary from one execution to another, even if the starting data is identical" (Cantot and Luzeaux 2013, 59). Figure 58 illustrates this concept, where S_i represents the current system state. In a deterministic system process, the system, S_i, transforms into a known state, S_j, with one-hundred percent certainty. In a stochastic system, the current state of the system, *Si,* may transition into two future variations, S_j or S_k, and this transition is random, we don't know with one-hundred percent certainty which form the current system will take, but we may know the probability that it will take one form vs. the other—e.g., the probability that S_i will transition to S_j is 80%, therefore, the probability that S_i will transition to S_k is 20%.

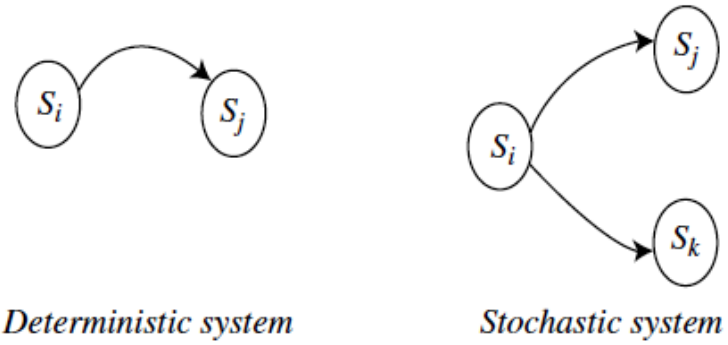


Figure 58. Deterministic vs. stochastic evolution of a system's state. Source: Cantot and Luzeaux (2013, 60).

In addition to the stochastic or random system definition provided by Cantot and Luzeaux (2013), another reason that the system can yield random results is not because the model's logical rules or mathematical relationships are stochastic, in fact, they may be entirely deterministic, but because a probabilistic input is random, it varies with each simulated event, therefore, the mechanistic model is processing a different value each for each simulated event, which is why results also vary with each event. There is no variation or change within the defined model logical rules. Oberkampf and Roy (2010, 69–70) provide a concise description of a nondeterministic mathematical model:

> the mathematical model of the system, which includes the influence of the surroundings on the system, is considered nondeterministic in the sense that: (a) the model can produce nonunique system responses because of the existence of uncertainty in the input data for the model, (b) the analysis may consider multiple possible environments and scenarios that the system may experience, and (c) there may be multiple alternative mathematical models for the same system of interest. The term nondeterministic is used instead of stochastic because nondeterminism can be due to either aleatory or epistemic uncertainty or, more commonly, a combination of both. The mathematical models, however, are assumed to be deterministic in the sense that when all necessary data for a designated model are specified, the model produces only one value for every output quantity. That is there is a one-to-one correspondence from input to output of the model.

One key thing to note is that Oberkampf and Roy (2010, 70) provide a more precise description of "nondeterministic," where nondeterminism can be due to both aleatory

and/or epistemic uncertainty, therefore the term stochastic is mainly reserved form aleatory uncertainty due to randomness.

Figure 59 provides a simple graphical illustration to contextualizes the basic parts of a nondeterministic simulation model. In the model depicted, a probability distribution function is assigned to each input parameter; the distribution is based on statistical observations of that particular system's input metric data. The model has logical process or mathematical relationships that are deterministic in mechanism but may possess epistemic uncertainty, meaning that logical processes or mathematical relationships that represent real-life system behavior may be absent in the model due to lack of knowledge. Thus, the model may not accurately function or behave like the system it is based on. The model then processes that randomly generated input values and generates output results. Results are also probabilistic in nature; each output metric is also represented by a probability distribution.

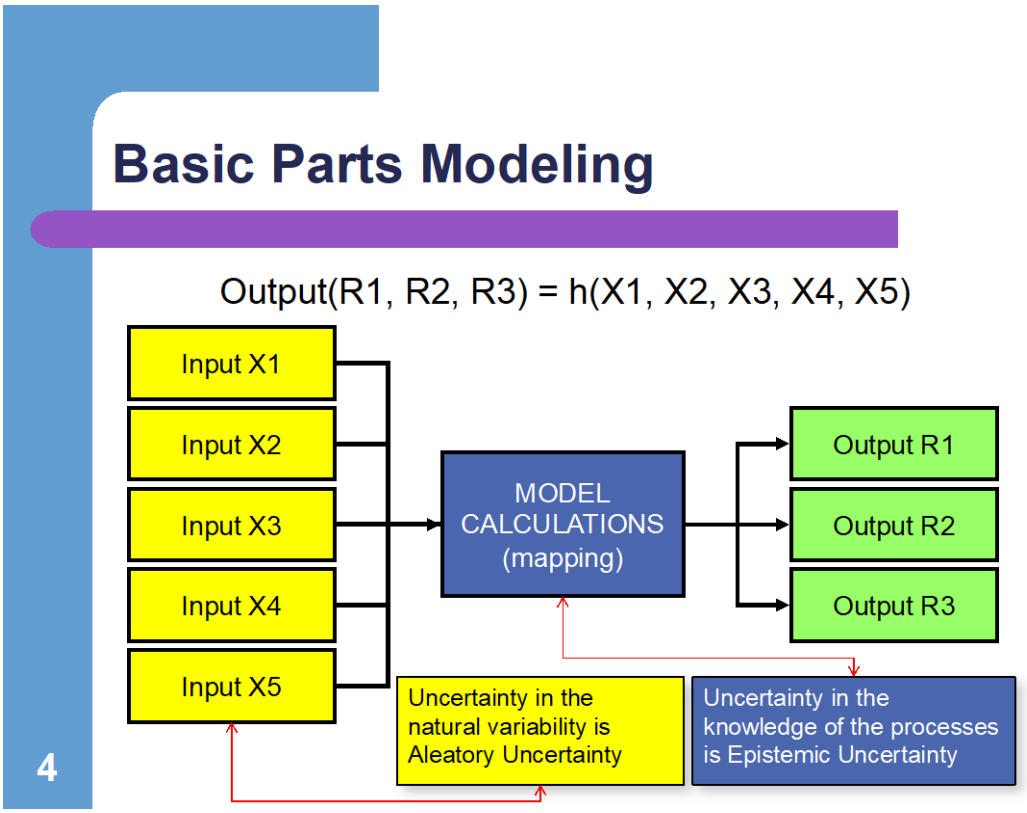


Figure 59. Stochastic model parts and sources of uncertainty. Source: Rhoades (2020, 4).

According to Oberkampf and Roy (2010, 70), "the most common strategy for incorporating uncertainty directly into the computational analysis involves three basic steps." Figure 60 illustrates these steps. These steps are: (1) characterizing the sources of uncertainty, (2) propagation of uncertainty, and (3) analysis of the model-output (Oberkampf and Roy 2010):

1. *Characterizing the sources of uncertainty*: In this step uncertainties are characterized as either aleatory or epistemic (Oberkampf and Roy 2010). If an uncertainty is characterized as aleatory, it is characterized by a probability distribution function (PDF) or a cumulative distribution function (CDF) (Oberkampf and Roy 2010). If uncertainty is characterized as epistemic, then it is characterized as an "interval-value quantity with no likelihood information specified" (Oberkampf and Roy 2010, 70).

2. *Propagation of the Uncertainty*: In this step, "values from the uncertain input quantities specified in the previous step are propagated through the model to obtain uncertain output quantities" (Oberkampf and Roy 2010, 71).

3. *Analysis of the model output*: In this step the ensemble of calculations produced by sampling for the system response quantities (SRQs) of interest is interpreted (Oberkampf and Roy 2010).

Oberkampf and Roy (2010, 71) further describe the characteristics of the system response quantities (SRQs):

> The general form of the ensemble of calculations is a family of CDFs or, equivalently, a family of complementary cumulative distribution functions. Multiple probability distributions are produced because of the existence of epistemic uncertainty. Each probability distribution represents the results from sampling all of the aleatory uncertainties, from *one* sample from all of the epistemically uncertain quantities. Each one of the probability Distributions represents a possible probability distribution of the SRQs.

It should be noted that the definitions for *epistemic uncertainty* provided by Oberkampf and Roy (2010) and by the author differ with respect to context. Oberkampf and Roy (2010) imply that within a model, there are various model processes (or parameters) that exist that are employed, but not all of them are employed completely or at the same time, so in order to get a fuller model answer, the user should select the combination of processes or parameters of interest—for a subset, or run all combinations for an exhaustive set of the domain. The author considers epistemic uncertainty to include omissions of the system's behavior, processes, qualities, or characteristics that were not encoded into the model, either because of simplification/abstraction omissions (reductions) or because the existence of such details is not present in the model. For example, a simple mathematical model for a ball's velocity in free-fall may be described by the following equation:

$$V_f = V_i + gt$$

where, V_f is the velocity (final) at time t, V_i is the initial velocity at $t = 0$, and g is the earths gravitational acceleration; downward is the "positive" direction; only the force of

gravity ($F = ma = mg$) is acting on the ball, thus $a = g$. One glaring simplification is the omission of the effects of air resistance. In the given equation, given enough fall time, the ball's velocity will increase for as long as it falls. However, in the real-world, the ball will reach a terminal velocity when the air-resistance and gravitational force on the ball are equal and air resistance will slow down the ball all throughout its fall until it reaches its terminal velocity. In this example, the model's *epistemic uncertainty* is contributed by the omission of the effects of air resistance and induces an error in the ball's predicted velocity at a given time. This example is also useful with regards to V&V. The numerical calculation may be computed correctly and verified to be correct according to specification of the simple equation selected, but the numerical calculation may be overall invalid for the user's purpose.

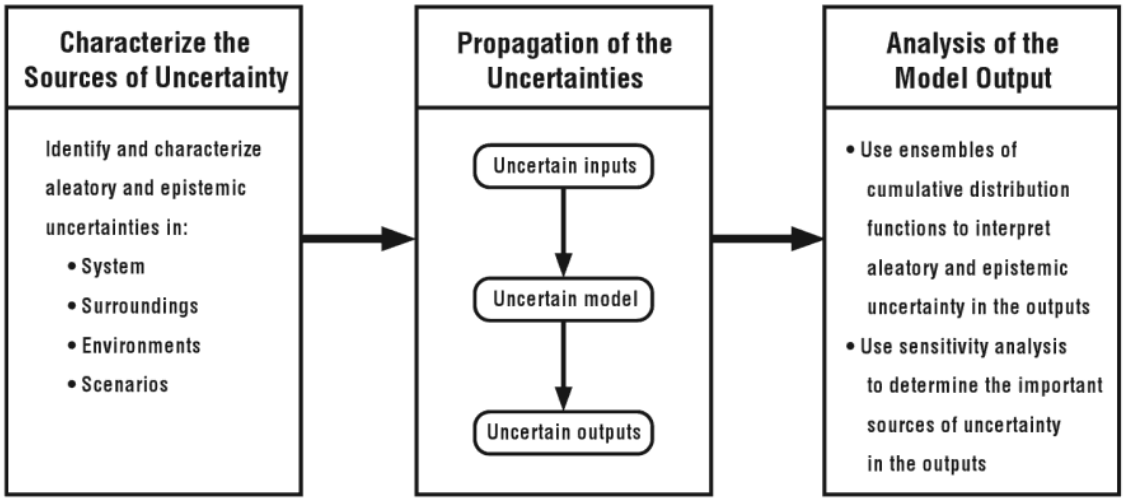


Figure 60. Basic steps in an uncertainty analysis. Source: Oberkampf and Roy (2010, 70).

c. Modeling errors, Accuracy and Credibility

Accuracy has to do with how close the output value(s) of the simulation are to the system's true values. To assess accuracy, *solution verification* must be performed on the results on the model. Beisbart and Saam (2019, 77) state that "the primary goal of solution verification is to estimate the numerical accuracy of a discrete solution to the mathematical

model of the system of interest for specific input data," and that solution verification is "dependent on the quality and completeness of code verification." If a model is "accurate" with respect to its intended use and purpose, the solution verification can support operational validation, where the simulation results are compared with measurements of the systems behavior or response. Accuracy of the model is impacted by errors that are introduced along the experimentation process or are inherently built into the model due to lack of knowledge on the system or coding errors. Beisbart and Saam (2019) provide a useful definition for error:

> An *error* is defined as the difference between the value obtained and the true value, and thus has both a magnitude and a sign; the true value depends on the context of use. (Beisbart and Saam 2019, 120)

According to Beisbart and Saam (2019, 77), solution verification "attempts to identify and quantify three sources of errors that can occur in the exercise of a simulation code," represented in Figure 61.

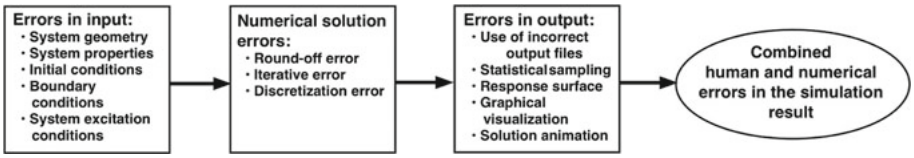


Figure 61. Three types of error sources in solution verification—from the perspective of a Partial Differential Equation simulation. Source: Beisbart and Saam (2019, 77).

Beisbart and Saam (2019) provide the following additional descriptions for the sources of error identified:

- Errors, blunders. Or mistakes made by the computational analysis in preparation of the input data for the simulation code;
- Numerical solution errors resulting from computing the discretized solution of the mathematical model; and
- Numerical errors, blunders, or mistakes made by the computational analysts in any processing or presentation of the output data that is produced by the simulation code. (Beisbart and Saam 2019, 77)

For numerical simulation results that are also probabilistic in nature, *statistical significance* may be used to determine the accuracy of the results. Sabine and Plumpton (1985) provide useful guidance for deciding on how to approach uncertainty in data sampling:

> We must decide whether the question involves interval estimation or hypothesis testing; also, how many random samples we require and whether small or large samples are suitable. We must also decide what assumptions are required for the theory we intend to use, and whether those assumptions are justified. (Sabine and Plumpton 1985, 85)

H. MODELING AND SIMULATION VALIDATION, VERIFICATION AND ACCREDITATION

1. M&S VV&A Background and Definitions

According to Oberkampf and Roy (2010), the Operations Research (OR) community, also referred to as the "systems analysis" or "modeling and simulation" community (Oberkampf and Roy 2010, 22), is the "first applied technical discipline that began to struggle" with the methodology and terminology of V&V. Oberkampf and Roy (2010, 22) state that the OR community analyzed systems or problems that could be "extraordinarily complex," involving "strong coupling of complex physical processes, human behavior in a wide variety of conditions, and computer-controlled systems that adapt to changing systems characteristics and environment" (Oberkampf and Roy 2010, 22). Although OR models heavily rely on mathematical analysis to quantify solutions for business or management type questions, these OR models typically do not deal with more fundamental scientific models such as computational fluid dynamics models which rely on mathematical relationships based on physics. Here are some examples of subjects that are tackled by the OR community (Oberkampf and Roy 2010, 22):

- Industrial production models
- Business or governmental organizations
- Marketing models
- National and world economic models
- Military conflict models

The subject under study and experimentation for this thesis's methodology, the Cost Readiness Impact Model (CRIM), is best classified as an OR type M&S product.

CRIM seeks to address management and operations related questions for NAVAIR, and it does so by modeling the necessary existing O&M physical (not to be confused with physics) processes that serve as the underlying logical framework.

It is important to define, understand, and compare variations of V&V definitions employed by various technical communities. Differences and variations in terminology may cause ambiguity, confusion or and even "unnecessary disagreements" (Oberkampf and Roy 2010a, 21). V&V definitions by the OR community, the International Electrical Engineering (IEEE) community the DOD community, Pace (2004) and Sargent (2013) are given below. For more definitions by the American Institute of Aeronautics and Astronautics (AIAA), the American Society of Mechanical Engineers (ASME) communities, and the Hydrology community see Oberkampf and Roy (2010). The purpose of presenting these various definitions is: (1) to show that the V&V perspectives are not always similar across communities and perhaps even amongst individuals of the same community; and (2) so that the reader may synthesize these definitions in their mind and form a more robust intuition of V&V definitions and the nature of V&V. For an M&S V&V project, it is recommended that the stakeholders agree upon which V&V terminology to use. In the authors' opinion, the definitions provided by the DOD, especially in the *Department of Defense Standard Practice Documentation of Verification Validation, and Accreditation (VV&A) for Models and Simulation*, MIL-STD-3022 w/CHANGE 1 (Department of Defense 2012), are suitable general definitions, and also specifically suitable for this thesis research.

a. OR community V&V definitions

These are the definitions given by Oberkampf and Roy (2010, 22) for the OR community, originally published by the Society for Computer Simulation:

- Model verification: substantiation that a computerized model represents a conceptual model within specified limits of accuracy.
- Model validation: substantiation that a computerized model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model.

Oberkampf and Roy (2010, 22) point out that "the main implication [of these definitions] is that the computerized model, i.e., the computer code, must accurately mimic the model that was originally conceptualized."

b. IEEE and related communities V&V definitions

Oberkampf and Roy (2010, 24) capture the IEEE definitions:

- Verification: the process of evaluating the products of a software development phase to provide assurance that they meet the requirements defined for them by the previous phase.
- Validation: the process of testing a computer program and evaluating the results to ensure compliance with specific requirements.

The definition of validation here appears to be an extended definition of verification, because "compliance with specific requirements" is subjective and could bypass the true purpose of validation, which in the context of simulation models, should be ground on reality. As Oberkampf and Roy (2010, 25) point out, these definitions are more suited for "software V&V" and not for "model V&V." The IEEE definitions are presented to illustrate the point that not all communities have the same perspective of V&V and that for different subjects or applications, V&V can be adapted for suitability.

c. The Department of Defense community V&V definitions

The following definitions are provided by Oberkampf and Roy (2010, 25), and based off the 1994 Department of Defense, Defense Modeling and Simulation Office (DMSO).

- Verification: the process of determining that a model implementation accurately represents the developer's conceptual description of the model.
- Validation: the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model.

d. Secretary of the Navy Instruction 5200.46 V&V definitions

The following definitions are provided by the Secretary of the Navy Instruction 5200.46 document (Secretary of the Navy 2019, 3). This document includes other useful M&S VV&A definitions and policy.

- Verification: The process of determining a model, simulation, and associated data accurately represent the developer's conceptual description and specifications.
- Validation: The process of determining the degree to which a model or simulation and its associated data are an accurate representation of the real world from the perspective of the intended uses of the model.

e. Department of Defense MIL-STD-3022 w/CHANGE 1 definitions

The following definitions are from the DOD "Documentation of Verification, Validation, and Accreditation (VV&A) for models and Simulations" (Department of Defense 2012, 3). These definitions expand over the SECNAV instruction V&V definitions by including the additional "federation of models" terminology. These definitions are more compatible with the consideration of a federation of models or a "system-of-systems" architecture of models.

- Verification: The process of determining that a model, simulation, or federation of models and simulations implementations and their associated data accurately represents the developer's conceptual description and specifications.
- Validation: The process of determining the degree to which a model, simulation, or federation of models and simulations, and their associated data are accurate representations of the real world from the perspective of the intended use(s).

f. Academic V&V definitions

The following definitions are quoted from Sargent (2013, 12):

- Model verification is defined as 'ensuring that the computer program of the computerized model and its implementation are correct.'
- Model validation is defined as the 'substantiation that a model within its domain of applicability possesses a satisfactory range of accuracy consistent with the intended application of the model.

The following definitions are quoted from Pace (2004, 164), where he provides the definitions in the form of questions regarding the model or simulation, and aspects of V&V:

- Verification: Did I build the thing right? Have the model and simulation been built so that they fully satisfy the developer's intent (as indicated in specifications)? Verification has two aspects: design (all specifications and nothing else are included in the model or simulation design) and implementation (all specifications and nothing else are included in the model or simulation as built).
- Validation: Did I build the right thing? Will the model or simulation be able to adequately support its intended use? Is its fidelity appropriate for that? Validation has to aspects: conceptual validation (when the anticipated fidelity of the model or simulation conceptual model is assessed) and results validation (when results from the implemented model or simulation are compared with an appropriate referent to demonstrate that the model or simulation can in fact support the intended use).

2. Sargent's Simplified Model Development Process and V&V Process Relationships

The simplified version of Sargent's "model development process" is presented in this subsection. Although simpler than Sargent's (2013) "real-world" and "simulation world" relationships with V&V, this simplified model offers most of the concepts necessary to understand the basic M&S V&V relationships. Figure 61, known by some as the "Sargent Circle" (Pace 2004, 164), shows Sargent's (2013) simplified version of the model development process and V&V process.

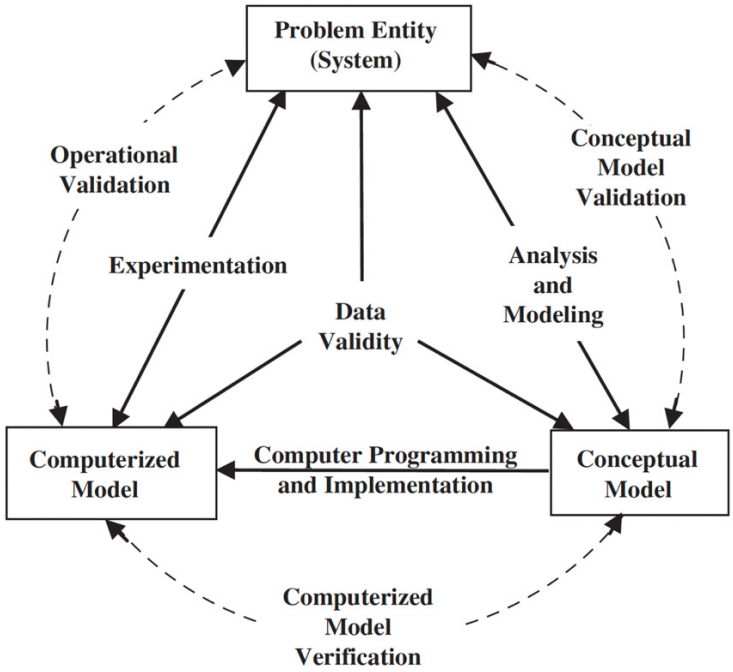


Figure 62. Sargent's simplified version of the model development process. Source: Sargent (2013, 14).

a. M&S Entities

The following definitions for **M&S entities** are directly from Sargent (2013, 14); these are the entities represented by rectangles in Figure 62:

- The problem entity is the system (real or proposed), idea, situation, policy or phenomena to be modeled.
- The conceptual model is the mathematical/logical/graphical representation (mimic) of the problem entity developed for a particular study.
- The computerized model is the computer implementation of the conceptual model. (Sargent 2013, 14)

The "Sargent Circle" is similar to, if not derived from, the "proper framework to review the credibility of a simulation," presented by Schlesinger (1979, 1) (Schlesinger 1979, 1). Oberkampf and Roy (2010, 23) use this framework and label it as the "phases of M&S and the role of V&V," but the construct is unchanged from Schlesinger (1979) and presented in Figure. 63.

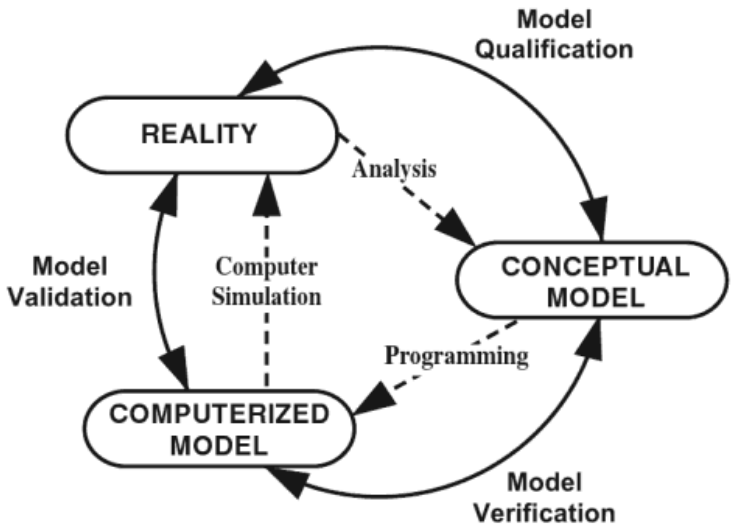


Figure 63. Phases of M&S and the role of V&V. Source: Schlesinger (1979, 1).

b. Summary of Alternative M&S V&V models

Oberkampf and Roy (2010) provide definitions for Figure 63, which are similar to the definitions defined by Sargent (2013):

- The conceptual model comprises all relevant information, modeling assumptions, and mathematical equations that describe the physical system or process of interest. The conceptual model is produced by analyzing and observing the system of interest. The SCS defined qualification as "Determination of adequacy of the conceptual model to provide an accept able level of agreement for the domain of intended application." The computerized model is an operational computer program that implements a conceptual model using computer programming. (Oberkampf and Roy 2010, 23)

Oberkampf and Roy (2010) offer an alternate, more linear, and sequential flow conceptual model, presented in Figure. 64.

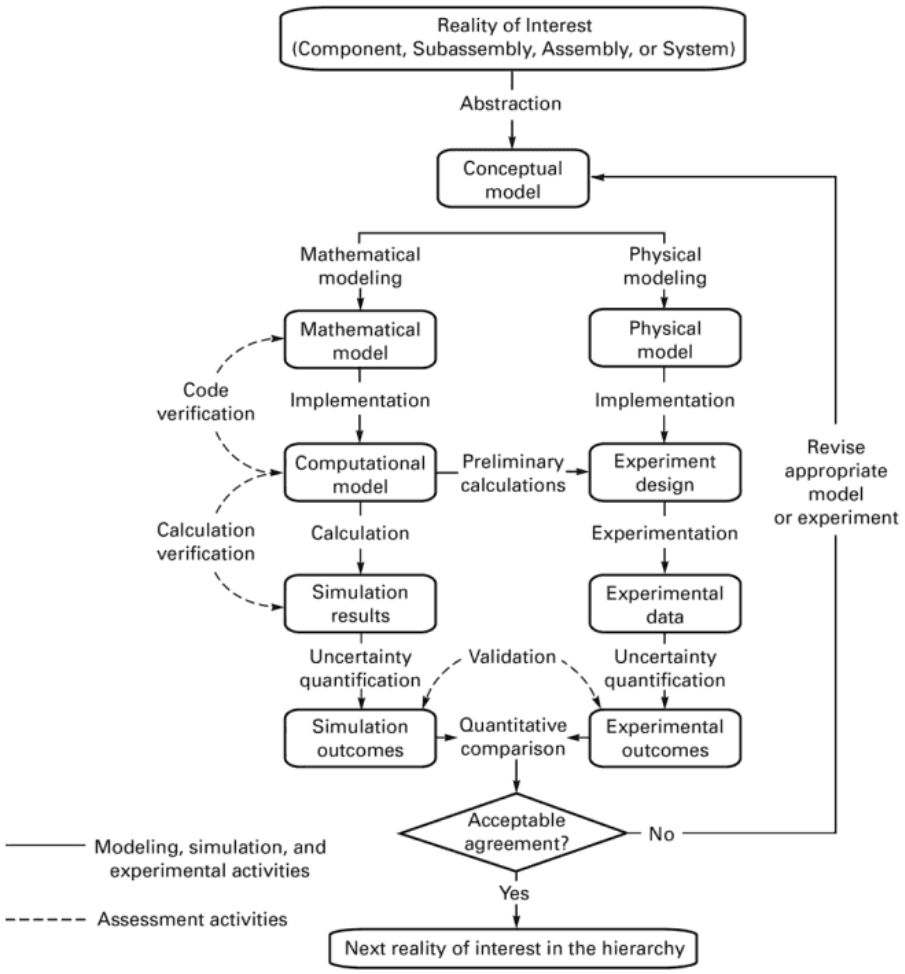


Figure 64. Verification and validation activities and products. Source: Oberkampf and Roy (2010, 30).

Beisbart and Saam (2019, 595) provide an "M&S life cycle" V&V management approach, where "at each M&S phase, appropriate V&V activities have to be defined, executed, and managed." This model is illustrated in Figure 65.

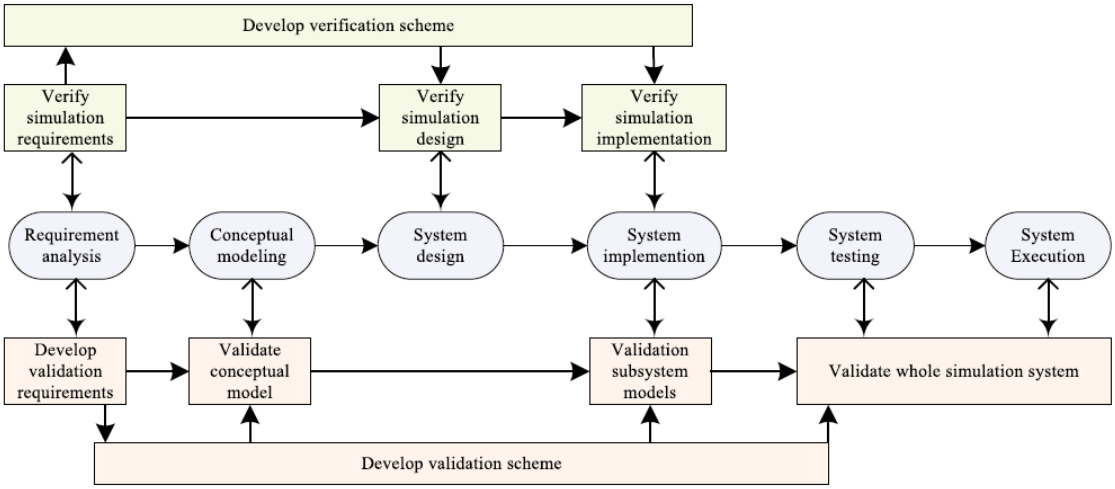


Figure 65. Simulation V&V process model. Source: Beisbart and Saam (2019, 595).

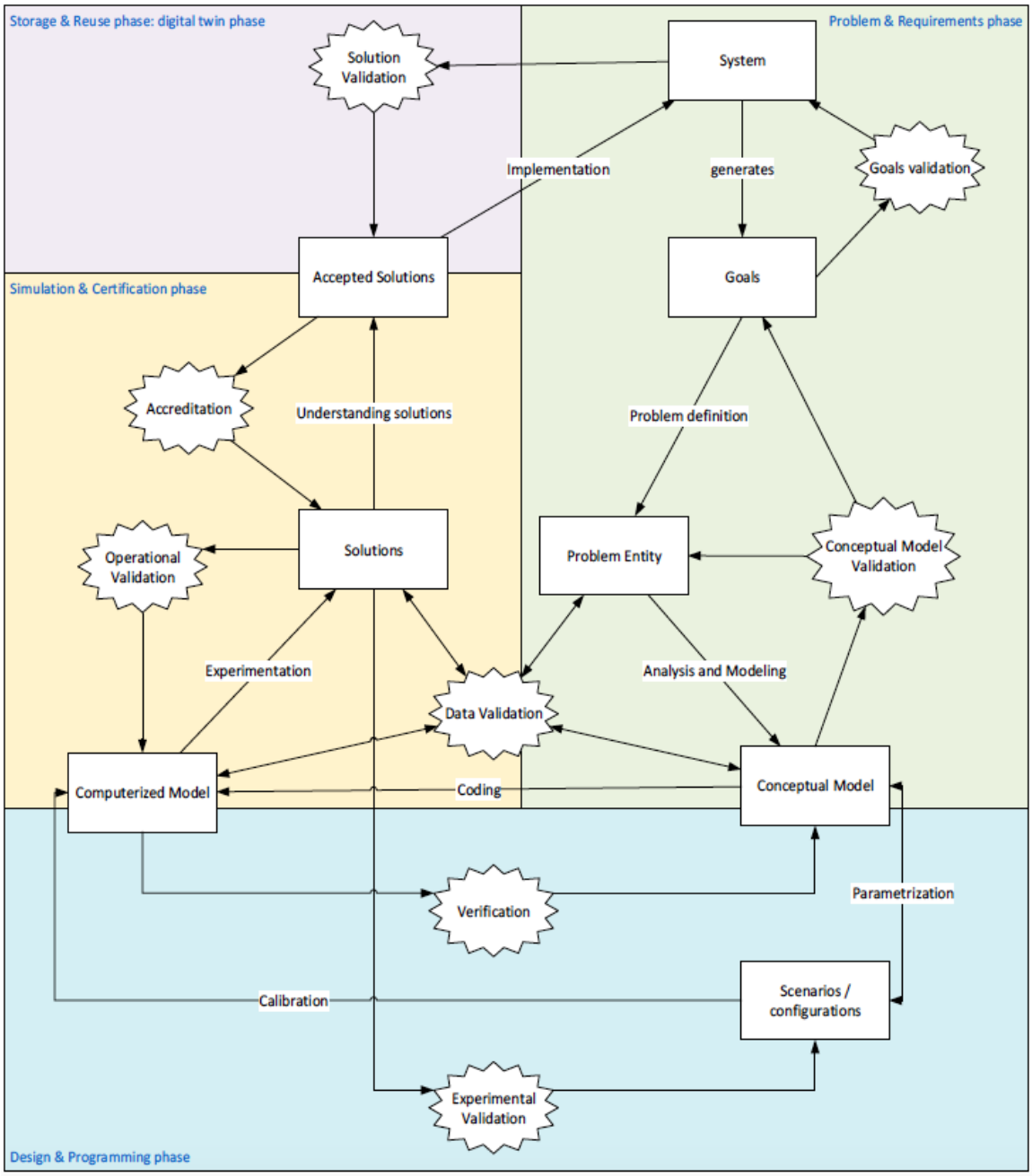
Beisbart and Saam (2019, 594–595) include the following main validation activities; the last two consist of the "so-called result validation":

- Develop simulation validation requirements
- Develop simulation validation scheme
- Validate conceptual model
- Validation subsystems models
- Validate the whole simulation system.

Beisbart and Saam (2019, 595) also include the main verification activities:

- Develop simulation verification scheme
- Verify simulation requirements
- Verify simulation design
- Verify simulation implementation

Finally, this last simulation model development and V&V conceptual model by Fonseca i Casas (2023), presented in Figure 66, is worthwhile including because it showcases the model entities, model development processes and V&V processes organized by development phases.



The squares represent the M&S entities/products, the stars represent the V&V actions, and the arrows represent the M&S development and implementation process. This M&S lifecycle is partitioned into four main M&S and V&V phases: (1) the problem and requirements phase, (2) the design and programming phase, (3) the simulation and certification phase, and (4) Storage and re-use phase: digital twin phase.

Figure 66. The M&S development and M&S VV&A life cycle. Source: Fonseca i Casas (2023, 5).

c. Data Validity

Sargent (1986) provides a definition for data validity and discusses the purpose for data validity with regards to building and validating models:

> *Data validity* is defined as ensuring that the data necessary for model building, model evaluation and testing, and conducting the model experiments to solve the problem are adequate and correct. (Sargent 1986, 237)

Data validity is placed in the center of "Sargent's Circle." This could have been done deliberately by Sargent to emphasize the centrality of valid data and because it is the source for building and validating models. The question regarding data validity is an important one. According to Sargent (2013, 17), *data validity* is "often not considered [generally] to be part of model validation," but it is very important because it impacts the performance of the model and the ability to validate the model. The author considers valid data to be an essential part of model validation. Sargent (2013, 17) states that data are primarily used for three uses or purposes; for model validation, only the first two uses are applicable:

1. For building the conceptual model: This type of data from the problem entity (system) drives the identification and development of appropriate system theories, mathematical models and logical relationships, estimating model parameter values, and developing and testing the model assumptions.
2. For validating the model: This type of data may be behavioral data obtained from the problem entity (system) which is needed to support the operational validity step. In this step, the problem entity's behavior is compared with the computerized model's behavior. These data are usually system input/output data. If problem entity (system) data are not available, then sufficient operational validity may not be obtained which also fails to achieve high model confidence.
3. For performing experiments with the validated model: these experiments provide data/results that can be used to make inferences about the real-life system.

What is needed to support these three uses of data mentioned above is that data is "appropriate, accurate, and sufficient," (Sargent 2013, 17) and that all "data

transformations, such as data disaggregation [or data aggregation], are made correctly" (Sargent 2013, 17). Data should be screened for outliers. Sargent (2013, 17) states that "Outliers should always be evaluated, and if correct, the reason for them occurring should be incorporated into the simulation model."

d. The Referent

Pace (2004) provides another definition for the *system data* which is derived from the *problem entity* (system) and is captured or measured for the three purposes stated by Sargent (2013): (1) building the conceptual model, (2) validating the model, (3) performing experiments. Pace's (2004, 165) definition of the system data is called the "referent." The following is a quote from Pace (2004):

> A recent study chartered by the Defense Modeling and Simulation Office (DMSO) VV&A Technical Director suggests that "the referent is the best or most appropriate codified body of information available that describes characteristics and behavior of the reality represented in the simulation from the perspective of validation assessment for intended use of the simulation" and provides guidance about referent identification selection, and description. (Pace 2004, 165)

The tragic irony is that, according to Pace (2004, 165), "the modeling and simulation literature has largely neglected the referent, in the past failing to provide guidance about how to select or describe information used as the *standard for comparison* in a validation assessment." In practice, Pace (2004, 165) states that "sometimes the 'best' information is too costly, would take too long to obtain, or has some other impediment, so 'appropriate' information that has adequate fidelity for the intended use of the model or simulation serves as the referent." What Pace (2004) appears to imply is that the *referent* is the ideal set of data for the purposes of M&S and M&S V&V. In reality, the (referent) threshold that stakeholders must achieve is the "appropriate" (Pace 2004, 165) level of information and data.

e. M&S Development Processes

The following definitions for **M&S development processes** or "phases," as generally referred to by Sargent (2013), are slightly modified versions of the original ones

provided by Sargent (2013, 14). Sargent (2013) does not explicitly regard these as "development processes," but this term is appropriate. Sargent definitions are syntactically woven into his text and are not clearly demarcated due to the fluid, interwoven structure in which they are written, thus the author has provided the following modified definitions—adding the term "process," but keeping the semantical essence the same as the original:

- The analysis and modelling phase is the process where the conceptual model is developed from the problem entity. As the name implies, the problem statement is studied, analyzed and understood. System theory and other scientific or technical concepts are used to assist with this task. (Sargent 2013, 14)
- The computer programming and implementation phase is the process where the computerized model is developed from the conceptual model. (Sargent 2013, 14)
- The experimentation phase is the process when inferences about the problem entity are obtained by conducting computer experiments on the computerized model. Experiments can be performed to give results that are used for operational validation purposes. It is recommended that the model undergo operational validation efforts and that stakeholders achieve a level of confidence or satisfaction with the model before employing experiments for real-world usage such as using model results to support managerial decisions. (Sargent 2013, 14)
- Data validity is defined as ensuring that the data necessary for model building, model evaluation and testing, and conducting the model experiments to solve the problem are adequate and correct. (Sargent 2013, 14)

These processes are represented by solid arrows in Figure 62. The order in which these process definitions appear implies their ideal (not universal) sequence for developing an M&S product. However, the nature of developing such M&S products is iterative and seeking for continuous improvements, therefore these processes may exist concurrently with bi-lateral feedback.

f. M&S Validation and Verification Processes

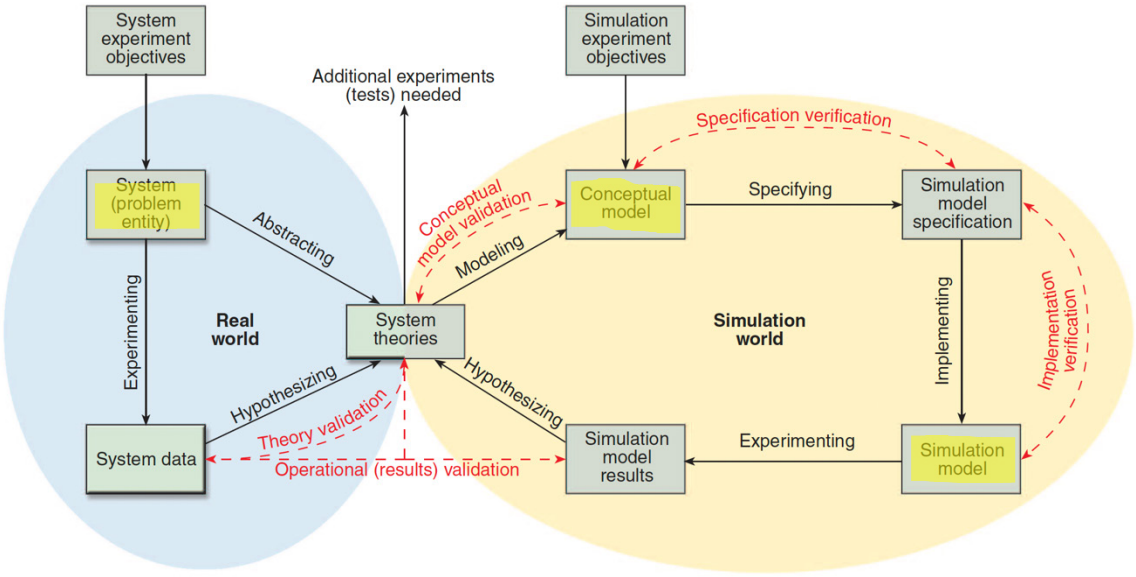
Finally, the M&S **validation and verification processes** and relationships are also defined by Sargent (2013, 14). These are listed below, worded just as they appear on Sargent's text:

- Conceptual model validation is defined as determining that the theories and assumptions underlying the conceptual model are correct and that the model representation of the problem entity is 'reasonable' for the intended purpose of the model.
- Computerized model verification is defined as assuring that the computer programming and implementation of the conceptual model are correct.
- Operational validation is defined as determining that the model's output behavior has a satisfactory range of accuracy for the model's intended purpose over the domain of the model's intended applicability.

It is recommended that when the conceptual model is developed that conceptual model validation follow suit before proceeding. It is likely that corrections or improvements to the conceptual model will be made after the initial conceptual model validation. This first validation may serve as the first "filter" making sure that the developers are developing the "right" model. The "right" conceptual model will serve as the basis for the design-to specifications of the implemented computerized model, which will go through verification ensuring that the computerized model is build "right" according to the conceptual model specifications. Operational validation is where the "rubber meets the road" because the computer model results are compared directly to the real-world data.

3. M&S V&V Framed through Sargent's Real-World and Simulation-World Paradigms

Sargent offers an expanded version of simplified model where entity details and relationships are made more explicit. It is worthwhile to also be familiar with this extended version. In this subsection, the new entities and processes and the relationships are covered. Sargent. The entities from the simplified version are highlighted for visual reference and comparison.



This diagram is taken from Pace's (2004) article; it was selected over Sargent's identical diagram because of this diagram's horizontal orientation and added colors in lieu of the original's black-grey-white tones, but the diagram is the same as the one developed by Sargent (2013). Source: Pace (2004, 165)

Figure 67. V&V relationships between the "Real-world" and "simulation-world" paradigms.

a. The Real-World Paradigm

Two paradigms are defined in Sargent' extended version: the Real-World and the Simulation-World. The Real-World paradigm is discussed first.

According to this framework, there "exists some *system or problem entity* in the real world, and an understanding of which is desired" (Sargent 2013, 14). Sargent (2013, 15) includes an entity called "System Experimenting Objectives." It is implied that these objectives are directly aligned to realize the "understanding" of the real world which is desired. However, it is unclear what the form of "experimenting" is meant here by Sargent (2013), but there are two likely possibilities: (1) the type of experimentation that can be done directly on the system such that the experimental conditions impact or invoke some sort of system behavior or response that is then observed, measured and analyzed for greater understanding – this type of experimentation may not be feasible in large complex systems; or (2) important and specific parts of the system are observed or measured to collect data and information which can then be used to understand the system

characteristics. It is likely that Sargent meant the latter definition of "experimenting," due to the necessity of data for building the model, validating the model, and performing computer simulated experiments.

The *experimenting* process is a development process that yields *system data/results* and is represented by a solid arrow in Figure 67.

b. System Theories: The Nexus between the Real-World and Simulation-World Paradigms

According to Sargent (2013, 14), s*ystem theories* "describe the characteristics and the causal relationships of the system (or problem entity) and possibly its behaviour (including data)." Sargent (2013, 14) also states that *system theories* "are developed by *abstracting* what has been observed from the system and by *hypothesizing* from the system data and results" and are validated by performing *theory validation* with respect to *system data/results*. Sargent (2013, 15) further expands on system theory development through abstraction by stating that "to develop valid systems theories, which are usually for a specific purpose, the system is first observed and then abstraction is performed from what has been observed to develop proposed system theories." With respect to *hypothesizing*, Sargent (2013, 15) states that "new proposed system theories may be hypothesized from the system data and the comparisons made, and also possibly from abstractions performed on additional system observations."

The two development processes used for deriving systems theories *abstracting* and *hypothesizing* are show by the solid arrows on Figure 66.

c. The Simulation-World Paradigm

> A simulation model should only be developed for a set of well-defined objectives.
>
> —Robert G. Sargent,
> Verification and Validation Of Simulation Models

The words in the epigraph above are likely to weigh heavily in the conscience of those who have experienced, many a time, situations where some type of implemented solution was developed and build without first identifying or defining the essential set of

stakeholder needs and requirements – at best it leads to a long windy road in development integration and testing, and at worst yielding a solution that is less than ideal or not the "right" solution—thus, not a valid solution. It is similar for M&S. An organization should understand operational purpose; its objectives at various levels and throughout the organization; its processes and the problems and challenges it must solve. However, experience reveals that organizations often struggle with these responsibilities especially when it comes to fully defining and understanding the problem-domain which serves as the basis for formulating stakeholder needs, requirements, and a solution-neutral functional description of *what* their M&S product needs to do to address the problem entity. Stakeholders in general should not begin to define the *how* aspects of an M&S product, meaning that stakeholders should not prematurely define an M&S implemented solution and how it will work and what form it will take before stakeholders understand the *what* the M&S product must achieve in relation to the stakeholder's needs and objectives.

The following are definitions provided by Sargent (2013, 15) with respect to the expanded **simulation-world entities**:

- The simulation model specification is a written detailed description of the software design and specification for programming and implementing the conceptual model on a particular computer system.
- The simulation model is the conceptual model running on a computer system such that experiments can be conducted on the simulation model.
- The simulation model data and results are the data and results from experiments conducted (experimenting) on the simulation model.

It is suggested that the *simulation model specification* be considered analogous to the *built-to* specifications with respect to system design, whereas the *conceptual model* would provide the *design-to* specifications. From a system perspective the simulation model software code would be part of the *product-baseline* (see the *Systems Engineering Technical Reviews, System Baselines and System Specifications* section of this chapter for more details).

The following are slightly modified **simulation-world development processes** definitions derived from the definitions provided by Sargent (2013, 15). Sargent does not explicitly call these out as "development processes," but the term is deemed appropriate

here and adds more explicit context. Sargent also does not provide clearly demarcated definitions in his text; thus, the following definitions are provided by the author, adding the term "development process" and minimally rephrasing the definitions, but these still retain the semantical essence of the definitions provided by Sargent (2013).

- Modelling is the development process in which the conceptual model is developed from the system theories.
- Specifying is the development process where the simulation model specification is developed from the conceptual model which considers the simulation objectives. The simulation model specification includes the software design, the programming and implementation specifications derived from the conceptual model.
- Implementing is the development process where the simulation model is developed and realized from the simulation model specifications.
- Experimenting is the development process where simulation model results/data are generated from conducting experiments in the simulation model. Inferences about the system can be made from simulation results.

The following are Sargent's (2013,15) definitions on the **simulation-worlds V&V process and relationships**:

- Conceptual Model Validation is defined as determining that the theories and assumptions underlying the conceptual model are consistent with those in the system theories and that the model representation of the system is 'reasonable' for the intended purpose of the simulation model.
- Specification Verification is defined as assuring that the software design and the specification for programming and implementing the conceptual model on the specified computer system is satisfactory.
- Implementation Verification is defined as assuring that the simulation model has been implemented according to the simulation model specification.
- Operational Validation Operational validation is defined as determining that the mode's output behavior has a satisfactory range of accuracy for the model's intended purpose over the domain of the model's intended applicability.

Sargent (2013) offers more elaborate discussions on *conceptual model validation*, *computerized model verification*, *operational validity* (validation), and documentation. Anyone performing M&S V&V should familiarize themselves more with these elaborate discussions provided by Sargent (2013). This document covers the basic elements and

definitions required to understand and utilize this thesis' proposed MBSE VV&A framework for M&S.

d. Sargent's Recommended V&V Procedure steps

Sargent (2013, 22) offers a generally applicable procedure for performing V&V on simulation models, the eight steps are as follows:

1. An agreement be made prior to developing the model between (a) the model development team and (b) the model sponsors and (if possible) the users that specifies the decision-making approach and a minimum set of specific validation techniques to be used in determining model validity.
2. Specify the acceptable range of accuracy required of the simulation model's output variables of interest for the model's intended application prior to starting the development of the model or very early in the model development process.
3. Test, wherever possible, the assumptions and theories underlying the simulation model.
4. In every model iteration, perform at least face validity on the conceptual model.
5. In every model iteration, at least explore the simulation model's behaviour using the computerized model.
6. In at least the last model iteration, make comparisons, if possible, between the simulation model and system behaviour (output) data for at least a few sets of experimental conditions, and preferably for several sets.
7. Prepare the verification and validation documentation for inclusion in the simulation model documentation.
8. If the simulation model is to be used over a period of time, develop a schedule for periodic review of the model's validity.

4. Validation Techniques

The following validation techniques were compiled by Sargent (2013, 16–17). According to Sargent they can be used "subjectively" or "objectively" (R G Sargent 2013). The "objective" case refers to techniques that use some type of "mathematical procedure or statistical test, for example hypothesis tests or confidence intervals" (Sargent 2013, 16). These techniques are a toolkit for the user who determines which ones should be used for

the various parts and levels of the model elements and even the model as a whole. Thus, these techniques are usually combined when performing V&V on the sub-models and overall model. Here is the full list and definitions of the validation techniques (Sargent 2013, 16–17):

1. Animation: The model's operational behaviour is displayed graphically as the model moves through time. For example, the movements of parts through a factory during a simulation run are shown graphically. (Usually, only a relatively short time interval can be observed, which may result in not all behaviours being observed.).
2. Comparison to other models: Various results (e.g., outputs) of the simulation model being validated are compared to results of other (valid) models. For example (1) simple cases of a simulation model are compared to known results of analytic models, and (2) the simulation model is compared to other validated simulation models.
3. Data relationship correctness: Data relationship correctness requires data to have the proper values regarding relationships that occur within a type of data, and between and among different types of data. For example, are the values of data collected on a system or model correct for some known relationship within some type of data such as an inventory balance relationship or a dollar relationship?
4. Degenerate tests: The degeneracy of the model's behaviour is tested by appropriate selection of values of the input and internal parameters. For example, does the average number in the queue of a single server continue to increase over time when the arrival rate is larger than the service rate?
5. Event validity: The 'events' of occurrences of the simulation model are compared to those of the real system to determine whether they are similar. For example, compare the number of fires in a fire department simulation to the actual number of fires.
6. Extreme condition test: The model structure and outputs should be plausible for any extreme and unlikely combination of levels of factors in the system. For example, if in-process inventories are zero, production output should usually be zero.
7. Face validity: Individuals knowledgeable about the system are asked whether the model and/or its behaviour are reasonable. For example, is the logic in the conceptual model correct and are the model's input–output relationships reasonable?

8. Historical data validation: If historical data exist (e.g., data collected on a system specifically for building and testing a model), part of the data is used to build the model and the remaining data are used to determine (test) whether the model behaves as the system does.
9. Internal validity: Several replications (runs) of a stochastic model are made to determine the amount of (internal) stochastic variability in the model. A large amount of variability among the replications may cause the model's results to be questionable, and if typical of the problem entity may question the appropriateness of the policy or system being studied.
10. Multistage validation: Naylor and Finger (1967) proposed combining the philosophy of science methods of rationalism, empiricism, and positive economics into a multistage process of validation. This validation method consists of (1) developing the model on theory, observations, and general knowledge; (2) validating the model's assumptions where possible by empirically testing them; and (3) comparing (testing) the input–output relationships of the model to the real system.
11. Operational graphics: Values of various performance measures, for example the number in queue and percentage of servers busy, are shown graphically as the model runs through time; that is, the dynamical behaviours of performance indicators are visually displayed as the simulation model runs through time to ensure that the performance measures and the model are behaving correctly.
12. Parameter variability-sensitivity analysis: This technique consists of changing the values of the input and internal parameters of a model to determine the effect upon the model's behaviour or output. The same relationships should occur in the model as in the real system. This technique can be used qualitatively—directions only of outputs—and quantitatively—both directions and (precise) magnitudes of outputs. Those parameters that are sensitive, that is, cause significant changes in the model's behaviour or output, should be made sufficiently accurate prior to using the model. (This may require iterations in model development.)
13. Philosophy of science methods: The three philosophy of science methods are rationalism, empiricism, and positive economics. Rationalism requires a model to be logically developed (correctly) from a set of clearly stated assumptions. Empiricism requires every model assumption and outcome to be empirically validated. Positive economics

requires only that the model outcomes are correct and is not concerned with a model's assumptions or structure (causal relationships or mechanisms).

14. Predictive validation: The model is used to predict (forecast) the system's behaviour, and then comparisons are made between the system's behaviour and the model's forecast to determine whether they are the same. The system data may come from an operational system or be obtained by conducting experiments on the system, for example field tests.
15. Structured Walkthrough: The entity under review is formally presented usually by the developer to a peer group to determine the entity's correctness. An example is a formal review of computer code by the code developer explaining the code line by line to a set of peers to determine the code's correctness.
16. Trace: The behaviour of a specific type of entity in a model is traced (followed) through the model to determine whether the model's logic is correct and if the necessary accuracy is obtained. (Most current simulation software provide for trace capability making the use of traces relatively simple.)
17. Turing test: Individuals who are knowledgeable about the operations of the system being modelled are asked whether they can discriminate between system and model outputs. (Schruben (1980) contains statistical tests for the Turing test.)

a. Using Graphical Models for Model Specification and Validation.

Sargent (1986), in 1986, proposed the concept that graphical models could be useful for model specification and model validation. Graphical models are a model of the simulation model and use a graphical language that has certain grammatical rules so that its semantics are understood by those familiar with the particular graphical language. It is interesting that Sargent's publication was in 1986 when he made the following statements:

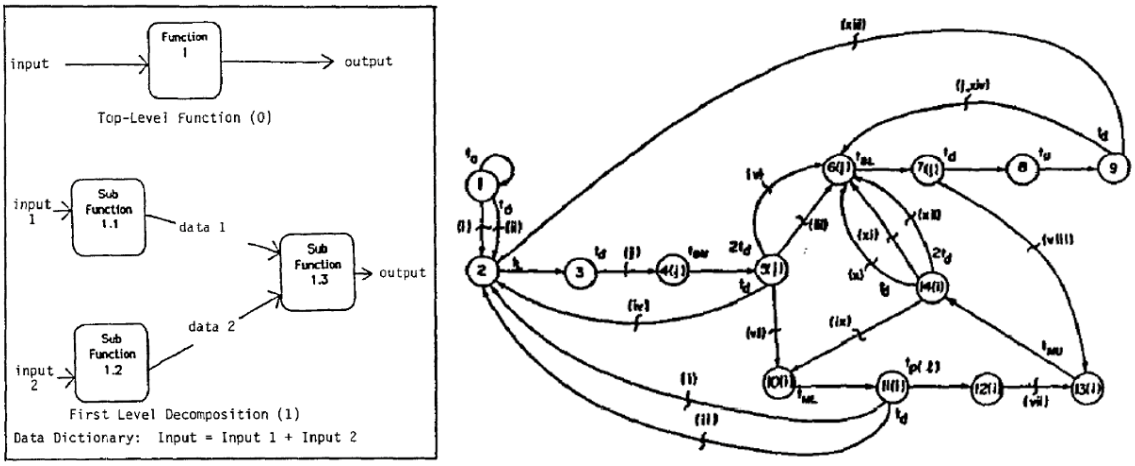
> The use of graphical models for model specification and in modeling is rapidly growing. This is occurring primarily because new interactive computer graphic software systems for developing graphical models of systems are rapidly becoming available at reasonable costs. …These new graphic software systems are being developed for an being used on personal computers, workstations, and graphic devices (terminals) that tie into mainframe computers….

> Preprocessors exist in some of the interactive computer graphic software systems to generate computer codes from the graphical models. (Sargent 1986, 237)

Sargent's statements are interesting because they sound similar to the statements of the MBSE community today. Regardless, the author independently arrived at the same concept, and implements it in this thesis's research methodology. Graphical models are not new and pre-date computers. For example, block diagrams and flow diagrams have been used manually before to represent the design logic of systems or products. "What is new," according to Sargent (1986), "are the wide variety of new graphical models becoming available and the (new) interactive computer graphic software systems being developed for graphical models."

Sargent (1986, 238) claims that "one group of graphical models which can be used in model validation is those for functional decomposition." Figure 68 presents a graphical description of the functional decomposition graphical models that Sargent (1986) is referring to. Sargent (1986) provides the additional quote:

> The diagrams form a hierarchical set, with lower levels expanding in more detail, on the functions and data of the upper level. The diagrams are numbered in a manner which shows their location in the hierarchy. This decomposition can continue, if desired, down to algorithms and to pseudo code. (Sargent 1986, 238)



Left: "DeMarco Diagram for Functional Decomposition," Right: "Event Graph of a Simple Flexible Manufacturing System." Source: Sargent (1986, 238–239)

Figure 68. Graphical model examples.

Sargent (1986, 239) states that "graphical models can be most helpful with conceptual model validity." Sargent (1986) specifies that the conceptual model can be modeled by the graphical model at various levels. A high-level will "identify and show the relationships among the subsystems or identify and show relationships among the high level system functions" (Sargent 1986, 239). A low-level or detailed model view "will contain the details of the model logic including its algorithms" (Sargent 1986, 239). The number of levels used will depend on the complexity of the system being modeled and the level of abstraction necessary to traverse the various levels. This may be subjective to the modeler. Sargent (1986, 239) makes an important distinction, "do not confuse the levels of representation of a [graphical] model with the amount of detailed *contained in a model*, i.e., its fidelity or accuracy." Out of the various validations techniques enumerated by Sargent (1986, 239), he considers *face validation* to be the "primary validation technique currently used to determine if the model representation is reasonable for the model's intended use." In *face validation*, "experts on the problem entity (e.g., the system) evaluate the [graphical] model [which is a representation of the conceptual model] to determine if they believe it [the conceptual model] is reasonable and correct for its intended purpose" (Sargent 1986, 239). Here is a caveat: the graphical model should be an accurate and

complete representation of the conceptual model. This is important because the experts are judging the conceptual model indirectly through the graphical model. Perhaps accuracy of the graphical model can be emphasized on the critical parts of the conceptual model where it is quintessential that experts be able to validate. With this said, not every part of the conceptual model needs to be modeled graphically, but the parts of interest of the conceptual model that are intended to be validated via the graphical model should be captured with great detail and precision in the graphical model, or at least with what is required for adequate validation.

To reiterate, the idea of using some descriptive graphical modeling language to help with the functional design, traceability, and V&V of software (in the form of M&S in our domain of interest) is not original to the author. Pace (2004, 163) states that "significant advances have resulted from increased use of the Unified Modeling Language (UML) and other descriptive paradigms facilitating formal consistency in model development, especially those using formal methods." The "advancements" that Pace (2004, 163) is talking about are in regard to software engineering maturity as a discipline which needs "formal, consistent, and repeatable processes in the production of quality software." Using descriptive languages like UML is helping software developers achieve said advancements.

This thesis' proposed methodology adopts this concept and expands the use of a descriptive model using SysML and Cameo Systems Modeler (CSM) to capture and describe CRIM. The MagicGrid Framework is used to organize the modeling effort and to provide logical and methodological approach to modeling.

b. Further V&V methods and recommended references

This section does not deep dive into M&S V&V techniques; that is outside the scope of this thesis research. This thesis research intends to identify, define, and reference V&V techniques at a high level so that someone undertaking an M&S V&V effort can survey what techniques may exist, but not necessarily at a detailed level. To achieve that, it is recommended that readers use the references provided in this document. Images from Fonseca i Casas (2023) are presented here because these do a great job at presenting

information concisely and are somewhat self-explanatory. It was deemed beneficial to include these as reference.

Figure 69 shows a classification of the various validation techniques with respect to their nature: static or dynamic; level of formality: informal or formal; and what their application is with respect the model: model logic or model output data.

	STATIC	DYNAMIC	
INFORMAL	✓ Face Validation ✓ Walkthroughs ✓ Audit ✓ Desk Checking ✓ Documentation consistency checking ✓ Inspections ✓ Reviews ✓ Traceability Assessment ✓ Interface Analysis (model interface; user interface)	✓ Visualization/Animation ✓ Turing Test ✓ Graphical Comparisons ✓ Acceptance Testing ✓ Object-Flow Testing ✓ Compliance Testing (authorization; performance; security; standards) ✓ Product Testing ✓ Alpha Testing ✓ Beta Testing	
FORMAL	✓ Induction ✓ Inference ✓ Logical Deduction ✓ Lambda Calculus ✓ Predicate Calculus ✓ Predicate Transformations ✓ Proof of Correctness ✓ Cause-Effect Graphing (State Transition analysis . . .) ✓ Goodness-of-fit ✓ Data Analysis (data dependency; data flow) ✓ Control Analysis (state transition; calling structure; analysis of concurrent process; control flow) ✓ Semantic Analysis ✓ Structural Analysis ✓ Syntax Analysis ✓ Fault/Failure Analysis	✓ Partition analysis ✓ Path Analysis ✓ Symbolic Execution ✓ Assertion Checking ✓ Inductive Assertions ✓ Fault/Failure Insertion Testing ✓ Interface Testing (data; model; user) ✓ Special Input Testing (boundary value; equivalence partitioning; degenerative; extreme input; invalid input; real-time input; self-driven input; stress; trace-driven input; fixed values)	TEST MODEL LOGIC
		✓ Black-Box Testing ✓ Bottom-Up Testing ✓ Top-Down Testing ✓ Debugging ✓ Execution Testing (monitoring; profiling; tracing) ✓ Field Testing ✓ Predictive Validation (historical data) ✓ Regression Testing ✓ Sensitivity Analysis ✓ Statistical Techniques ✓ Sub-model/Module Testing ✓ Symbolic Debugging ✓ Structural (White-Box) Testing (branch; condition; data flow; loop; path; statement) ✓ Comparison Testing ✓ Event Validity	TEST MODEL OUTPUT DATA

Figure 69. Validation test classification. Source: Fonseca i Casas (2023, 10).

Figure 70 shows a list of proposed documents associated with each of the modeling entities and offers an explanation for each of the documents proposed.

PRODUCT	DOCUMENTATION	EXPLANATION
SYSTEM	System documentation	All the documentation regarding the system operations. Tacit and informal information can also be used here to understand the system's true behavior.
GOALS	Goals document	A document that contains the goals that drive the development of the model.
PROBLEM ENTITY	Assumptions document	A document that contains the assumptions of the model.
CONCEPTUAL MODEL	Conceptualization of the model	The conceptual model constitutes a specification and includes all the assumptions (not the simplifications) detailed in the Hypotheses document.
SCENARIOS/ CONFIGURATIONS	Experimental design definition	A document containing the experimental design. A discussion regarding the quality of the RNG used in the experimentation can be added here if needed.
COMPUTERIZED MODEL	Code and technical documentation of the code	Along with the code, the technical documentation explains the code and the needed configurations of the computational infrastructure needed to execute and maintain the model.
SOLUTIONS	Solutions document	Description and data containing the solutions obtained from the model's execution.
ACCEPTED SOLUTIONS	Executive summary of the project	Conclusions of the modeling project and suggestions to perform the implementation on the system.

Figure 70. Proposed documents for each product of the modeling cycle. Source: Fonseca i Casas (2023, 6)

The tables presented in Figure 71 and Figure 72 provide details of "when to do the test, what test to do, how to do the test (if the test is Dynamic, needs the model execution, and if it is Informal, needs the experts on the system participation), and why, that is the nature of the assumptions that will be tested" (Fonseca i Casas 2023, 11).

WHEN	WHAT	HOW		WHY		
		Formal (F)/Informal (I)	Static (S)/Dynamic (D)	Systemic Structural	Systemic Data	Simplification
CONCEPTUAL MODEL VALIDATION	Audit	I	S	+		+
	Cause-Effect Graphing	F	S	+		+
	Desk Checking	I	S	+		+
	Documentation consistency checking	I	S	+		+
	Face Validation	I	S	+		+
	Induction	F	S	+		+
	Inference	F	S	+		+
	Inspections	I	S	+		+
	Interface Analysis	I	S	+		+
	Lambda Calculus	F	S	+		+
	Logical Deduction	F	S	+		+
	Predicate Calculus	F	S	+		+
	Predicate Transformations	F	S	+		+
	Proof of Correctness	F	S	+		+
	Reviews	I	S	+		+
	Traceability Assessment	I	S	+		+
	Walkthroughs	I	S	+		+
DATA VALIDATION	Events	F	D		+	
	Goodness-of-fit	F	S		+	
	Statistical Techniques	F	D		+	
VERIFICATION	Control analysis	F	S	+		
	Data Analysis	F	S	+		
	Debugging	F	D	+		
	Execution Testing	F	D	+		
	Fault/Failure Analysis	F	S	+		
	Interface Testing	F	D	+		
	Semantic Analysis	F	S	+		
	Structural Analysis	F	S	+		
	Symbolic Debugging	F	D	+		
	Syntax Analysis	F	S	+		

Figure 71. The When, What How Why (W3H) testing table, Part 1. Source: Fonseca i Casas (2023, 11)

WHEN	WHAT	HOW		WHY		
		Formal (F)/Informal (I)	Static (S)/Dynamic (D)	Systemic Structural	Systemic Data	Simplification
OPERATIONAL VALIDATION	Assertion Checking	F	D	+		
	Black-box Testing	F	D	+	+	+
	Bottom-Up Testing	F	D	+	+	+
	Comparison to other models	F	D	+	+	+
	Compliance Testing	I	D	+		
	Degenerative	F	D	+		
	Extreme conditions	F	D	+		
	Fault/Failure Insertion Testing	F	D	+		
	Fixed values	F	D	+		
	Graphical Comparisons	I	D	+	+	+
	Historical data	F	D	+	+	+
	Inductive Assertions	F	D	+		
	Object-Flow Testing	I	D	+	+	+
	Partition analysis	F	D	+	+	+
	Path Analysis	F	D	+		
	Sub-model/Module Testing	F	D	+	+	+
	Symbolic Execution	F	D	+		
	Top-Down Testing	F	D	+	+	+
	Turing tests	I	D	+	+	+
	Visualization/Animation	I	D	+	+	+
	White-Box Testing	F	D	+		
EXPERIMENTAL VALIDATION	Sensitivity analysis	F	D	+	+	
ACCREDITATION	Acceptance Testing	I	D	+	+	+
	Alpha Testing	I	D	+	+	+
	Beta Testing	I	D	+	+	+
	Product Testing	I	D	+	+	+
SOLUTION VALIDATION	Field Testing	F	D	+	+	+
	Regression Testing	F	D	+	+	+

Figure 72. The When, What How Why (W3H) testing table, Part 2. Source: Fonseca i Casas (2023, 12)

Beisbart and Saam (2019, 483) provide a summary table of frequentist statistical tools for (operational) model validation, shown in Figure 73. x_p represents the process and x_m the model. Beisbart and Saam (2019, 483) clarify the meaning of "Test is backwards," stating that this indicates that the null hypothesis is that the model is valid. Other clarifications are: "CLT" stands for Central Limit Theorem, and "Regression modeling is with the model predictions as the predictor variable and the process observations as the response variable" (Beisbart and Saam 2019, 483).

Name	Notes	Strengths	Weaknesses
Correlation	The correlation of x_m and x_p. High correlation implies a valid model	Familiar	Ignores intercept and slope
RMSE	The square root of the mean of $(x_m - x_p)^2$. Low *RMSE* implies a valid model	Units are the same as x_p	Tester has to define "low"
χ^2 test	The sum of $(x_m - x_p)^2/x_p$ has a known distribution when the model is valid. Low χ^2 implies a valid model	Formally tests validation	Test is backwards
T-test	The *t*-statistic has a known distribution when the model is valid. Low *t* implies a valid model	Formally tests validation	Test is backwards
MVS	Compare the means of the model and the process outputs and variances of the model and the process outputs using standard tests and test the null hypothesis that the regression slope is nonpositive. Similarity of means, similarity of variances, *and* nonnegative slope imply a valid model	Formally tests validation. Covers location, scale, and relationship	Test is backwards and Normality is assumed
Regression	Apply standard *t*-tests to the intercept ($H_0 : \beta_0 = 0$) and the slope ($H_0 : \beta_1 = 1$). Low *t* statistics imply a valid model	Formally tests validation. Tests the mean and point-to-point agreement	Test is backwards. Normality is assumed or *CLT* is invoked
WMT	Apply standard whole-model style *F*-test to the model ($H_0 : \beta_0 = 0; \beta_1 = 1$). Low *F*-statistic implies a valid model	Formally tests validation. Tests the mean and point-to-point agreement	Test is backwards. Normality is assumed or *CLT* is invoked
TOST	Prove that the mean of x_m is at the same time not too far above and not too far below the mean of x_p	Formally tests validation	Tester has to define "not too far". Normality is assumed or *CLT* is invoked
PTTE	Prove that the mean of x_m is at the same time not too far above and not too far below the mean of x_p	Formally tests validation. More powerful than *TOST* but tests different types of limits	Tester has to define "not too far". Normality is assumed
Fidelity	Apply standard *TOST* to the intercept ($H_0 : \beta_0 \neq 0$) and the slope ($H_0 : \beta_1 \neq 1$). High *t* statistics imply a valid model	Formally tests validation. Tests the mean and point-to-point agreement	Tester has to define "not too far". Normality is assumed or *CLT* is invoked

Figure 73. Testing simulation models using frequentist statistics. Source: Beisbart and Saam (2019, 483).

Beisbart and Saam (2019, 608) provides a structure or guide to perform a "validation experiment," which is described by the following quote, and illustrated in Figure 74.

> The top layer is the validation experiment as a whole. It consists of multiple-simulation configurations. In each of these, a point of the model's parameters space is selected, investigated, and then evaluated. In a stochastic model, for each configuration, a number of simulation runs (or replications) are executed. These sets of runs have to be analyzed, e.g., by calculation of mean values or confidence intervals. In a deterministic model, this layer is omitted, as it is not necessary to execute multiple runs for a single configuration. The bottom layer is formed by a single simulation run. (Beisbart and Saam 2019, 608)

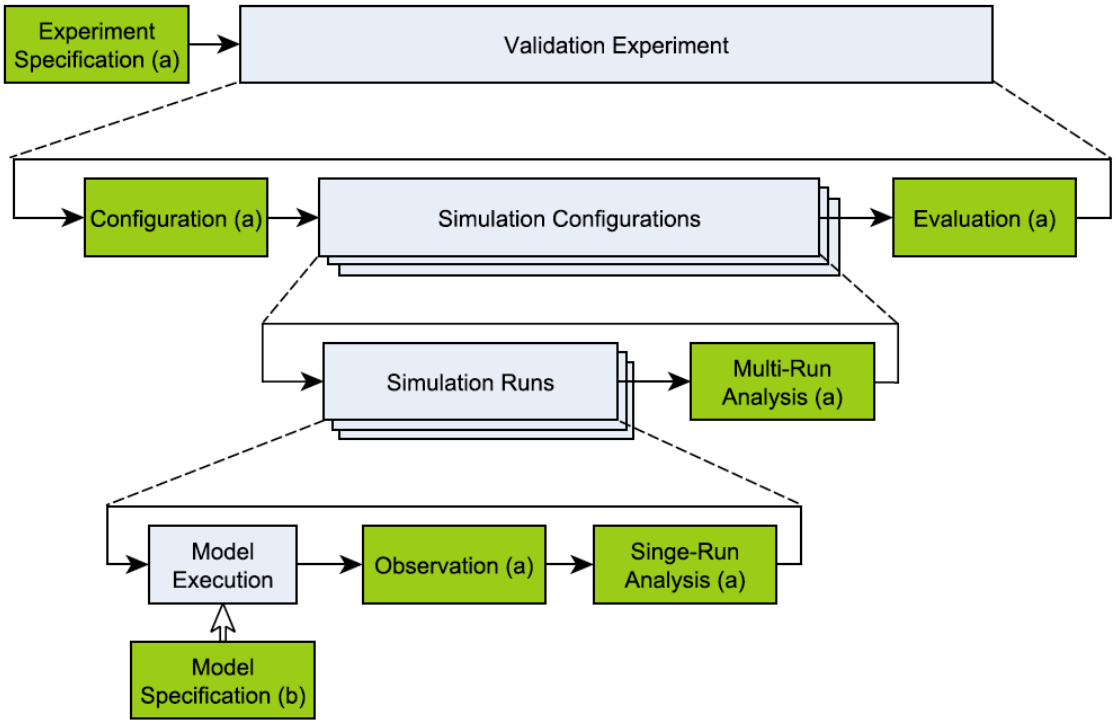


Figure 74. Framework for conducting validation experiments. Source: Beisbart and Saam (2019, 609).

5. Accreditation and Credibility

According to Pace (2004, 165), Accreditation is a "management decision that may include schedule and other considerations as well as technical V&V information." The

main question that must be addressed for an M&S product to be accredited is "*should it be used?*" (Pace 2004,165). Those with authority to make the accreditation decision are generally, or should be, those responsible for the consequences from the use of results from M&S products (Pace 2004; Oberkampf and Roy 2010b). The accreditation decision is often based on whether the M&S product meets its intended use (Pace 2004; Oberkampf and Roy 2010b) and satisfies "acceptability criteria," for example "capabilities that must be demonstrated for a particular application" (Pace 2004, 165). Pace (2004, 165) provides the following conceptual model that represents accreditation, shown here on Figure 75.

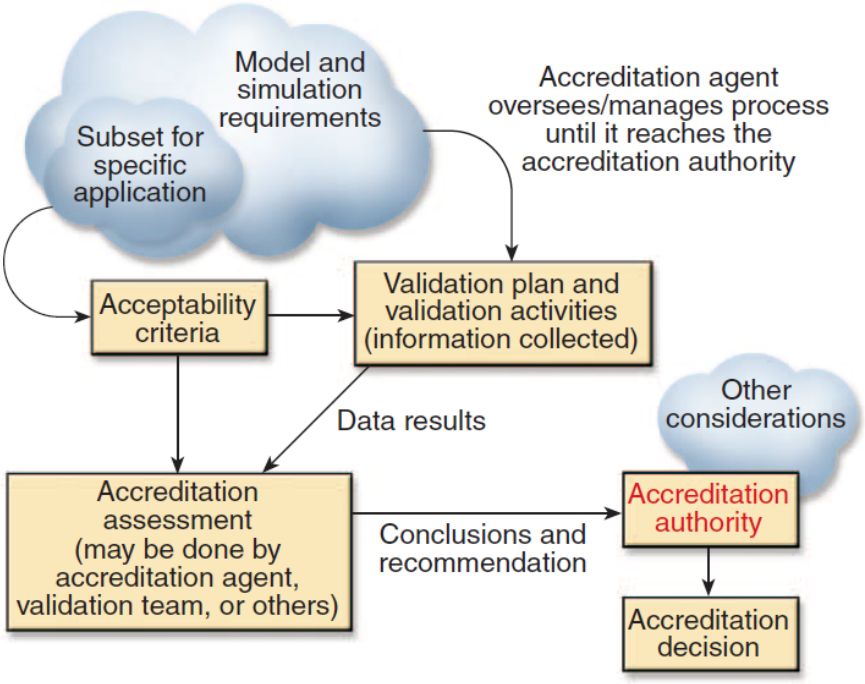


Figure 75. The accreditation conceptual mapping. Source: Pace (2004, 165).

Credibility is related to how stakeholders perceive the M&S results. Credibility may vary depending on the individual stakeholder and their familiarity or involvement with the M&S product. Regardless, the main question that is addressed by credibility is "*should results from the model or simulation be believed?*" (Pace 2004, 165). Pace (2004, 166) states that credibility is dependent upon "V&V information, accreditation status, the reputation of those who developed or use the model or simulation, and its history of use."

Pace (2004) also considers that bias or prejudice for or against the M&S product is also a factor.

With respect to the level of independence achieved while performing V&V, Oberkampf and Roy (2010, 49) provide a spectrum, shown in Figure 76. In reality, the overall V&V effort may be a mix of various levels of independent V&V performed—depending on who validates and verifies which parts of the model and that person or entity's degree of independence from the M&S developer.

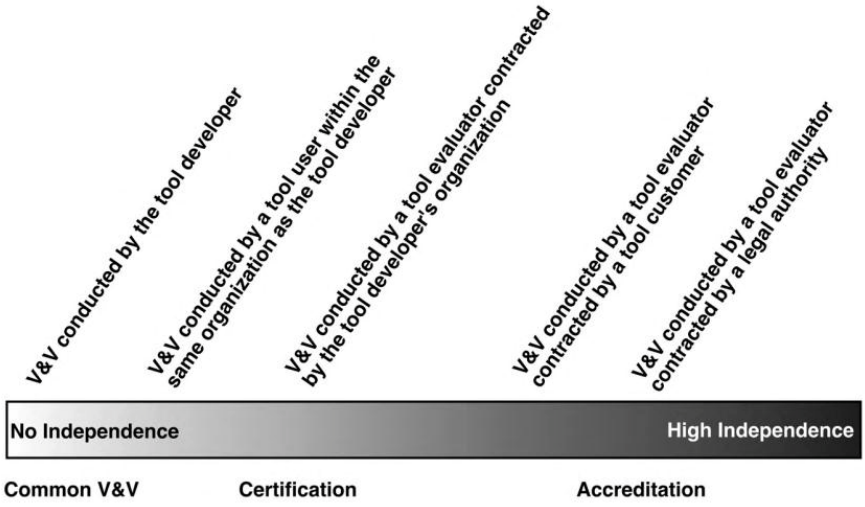


Figure 76. The V&V levels of independence spectrum. Source: Oberkampf and Roy (2010, 49).

I. COST READINESS IMPACT MODEL PAST V&V RESEARCH

Recent research has been performed by the Naval Postgraduate School (NPS) in support of CRIM VV&A efforts. An analytical evaluation of the SIMLOX outputs, using the statistical t-test was performed (Horton 2023). SIMLOX is a discrete event Monte Carlo simulation software that is used to simulate the multi-echelon, multi-indenture inventory processes of Navy and other aircraft. For this research, the SMILOX results for the FA-18 aircraft were studied. Horton (2023) collaborated with the Institute for Defense Analyses (IDA); the developers of the FA-18 SIMLOX model to perform a qualitative walkthrough of the model, evaluating the correctness of the multi-echelon, multi-indenture processes

for the FA-18 aircraft. Horton (2023) obtained modeling results and performed what is considered *operational (result) validation* of the SIMLOX model. For more details and results, see Horton's (2023) thesis research.

The thesis performed here, by the author, is associated to Horton's (2023) research. However, the author seeks to capture the full domain of CRIM M&S VV&A elements and present them in a framework that is useful and can support future CRIM VV&A efforts. Horton's (2023) operational validation work can be formally recorded by the author in context of CRIM's model entities and in context of other M&S VV&A elements.

J. MULTI-ECHELON, MULTI-INDENTURE INVENTORY SYSTEMS

Multi-Echelon system theory and implementation (practice) can be categorized under (or as a part of) the more general (or broader) domain of Inventory Control Systems, which is further categorized as a part of Operations or Business management. This background section will briefly cover fundamental subjects of multi-echelon systems, especially in context to military logistics applications. This background supports the reader's understanding of the theoretical concepts and terminology used in multi-echelon systems theory which serves as an abstract model representation and description of the real-life Naval aviation multi-echelon Operations and Maintenance (O&M) logistics system (the subject of the model) and the model subject of this thesis, the Cost Readiness Impact Model (CRIM).

Most of the theoretical concepts and definitions presented in this section are from Craig C. Sherbrooke's (2004) book "Optimal Inventory Modeling of Systems Multi-Echelon Techniques" (the first edition was published in published in 1992, the second edition was published in 2004). Mr. Sherbrooke (1934-2022) made a career solving real inventory problems and made a significant contribution to the logistics community through his work and publications. Amongst all the literature reviewed in support of this multi-echelon inventory systems background section, this book is quintessential in forming and describing fundamental multi-echelon concepts and is a useful basis for understanding other multi-echelon theory and methods.

a. Multi-Echelon Inventory Systems Theory Definitions

- Multi-echelon: Hierarchy of maintenance and repair locations
- Multi-indenture: Hierarchy of items representing the system.
- Indenture: item level/position in the multi-indenture hierarchy
- Failure – demand: the terms are used interchangeably.
- Availability: the percent of time that the system is operational
- Item: the term "item" is used to designate a specific type of part
- Unit: the term "unit" is for the quantity of the item.
- Stock level: the stock level on any item at any location can be thought of as the average number of units of the item in repair or resupply plus some safety level to protect against variability in the demand and repair process.
- Cannibalization – consolidation of aircraft "holes" or backorders to the smallest number of aircraft possible by remove-and-replace maintenance.

2. The System Approach

Sherbrooke (2004) makes a distinction between methods that use a *system approach* vs. an *item approach* for calculating spares. Sherbrooke (2004) does not directly give a definition for the system approach, but the definition can be inferred to be the method that seeks to optimize objective metrics at the system-level context. According to Sherbrooke (2004), the manager's perspective should be at the system level, and answering questions such as:

- What does the optimal system cost-effectiveness curve look like?
- What can we do to change our logistics support structure to achieve a desired availability for more efficiently?
- How can we insure [sic] that 95% of our scheduled aircraft flights will not be delayed for lack of spare parts?
- How much more money do we need to spend to move from 95% to something higher? (Sherbrooke 2004, 2)

The system approach "requires the analyst to determine stock levels for each item and location that take into account the stock levels for other items and locations, as well as the system target. In statistical estimation, the system approach tells us that we should not limit ourselves to data on a given item when we are making estimates of its mean and variance" (Sherbrooke 2004, 224)

Effects of using the system approach leads to different stockage decisions and significant improvements in system availability and cost. Sherbrooke (2004) states that the system approach can be employed for:

1. Generation of cost-availability curves for a system.
2. Determination of the optimal stock level for an item at a location, taking into account that it depends on stock levels at other locations and for other items, as well as considering the target availability or cost.
3. Estimation of mean demand for an item (Bayes or James-Stein techniques that utilize data from other items).
4. Estimation of item variance-to-mean ratios from a power curve or Bayesian techniques (using data from other items).
5. Evaluation of prediction techniques (using stockage model and availability for a group of items). (Sherbrooke 2004, 234)

Sherbrooke (2004, 147) states that "an important aspect of the system approach is to obtain the maximum effectiveness per resource units (dollars, weight, volume or whatever)." To measure this, an availability-cost curve must be generated; and example is shown in Figure 77. Sherbrooke (2004, 3) states that "the system approach presents the manager with an [optimal] availability-cost curve of his 'efficient' system alternatives….any points below the curve are 'inefficient' in that it is possible to find solutions on the curve with more availability or less cost; points above the curve are unobtainable."

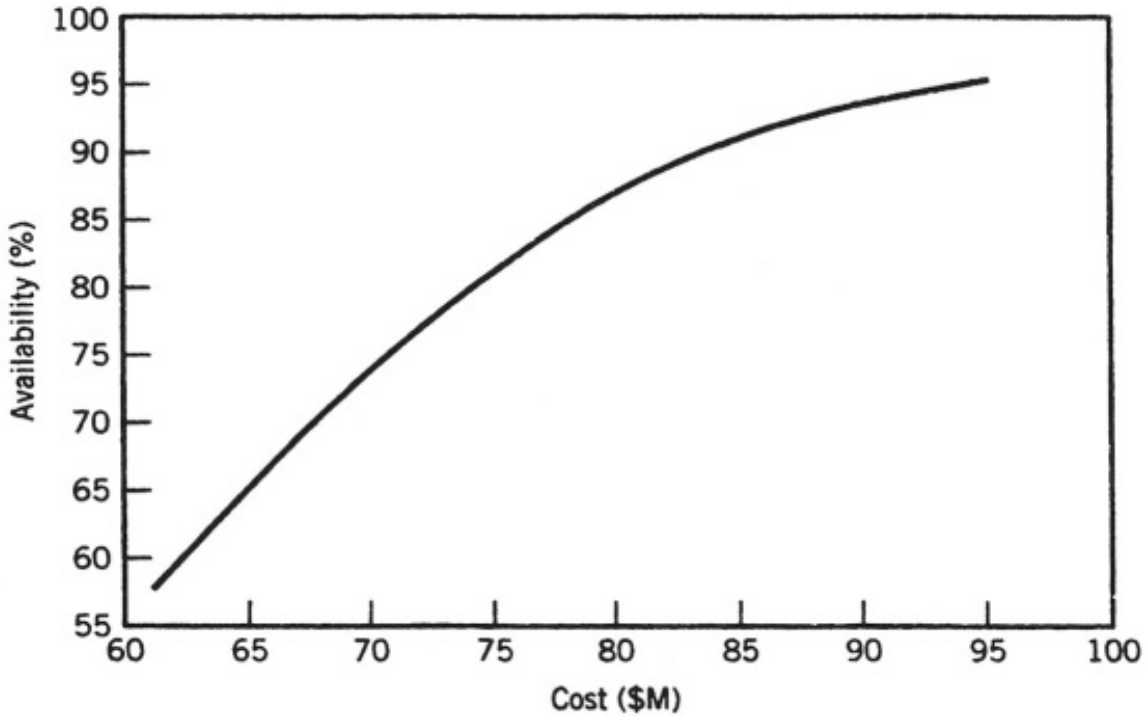


Figure 77. Optimal availability-cost curve example. Source: Sherbrooke (2004, 4).

NOTE: A strong parallel relation can be observed between the type of questions and answers sought by managers from the system approach, and the type of questions or metrics used for validation of the model. The model validation questions should trace back to the manager's needs.

3. The Item Approach

Traditional inventory theory uses the item approach, where calculations for spares are determined by simple formulas that balance the costs of holding inventory, ordering, and stockout (Sherbrooke 2004). This approach has been popular due to its simplicity. Calculations on the number of spares units of stock to buy on an item are made without considering other items. However, the disadvantage of the item approach is that the availability and total investment in the system of items are uncontrolled outputs of the item decisions (Sherbrooke 2004). Furthermore, if stockage decisions on each item were done independently, we would not know if the item decisions would produce acceptable system availability or satisfy budget requirements (Sherbrooke 2004).

4. The conceptual "Physics" behind the Multi-Echelon Inventory Problem

Sherbrooke (2004, 6–7) offers a description of the problem's "physics" before developing the theory that defines the problem mathematically.

> The simplest version of the problem is as follows: When a malfunction is diagnosed on an aircraft, the malfunctioning item is removed from the aircraft and brought into base supply. If a spare is available, it is issued and installed on the aircraft; otherwise a backorder is established for that user. We call this a first indenture item, because it is installed directly on the aircraft. Note that a base backorder on a first indenture item implies that there is a "hole" in an aircraft that causes it to be grounded. Later we discuss backorders at the depot and on lower-indenture items at the base. These supply system shortages are important and must be considered in our models, but they do not impact the aircraft directly, e.g., a base backorder for a second indenture item does not necessarily result in a "hole" in an aircraft.
>
> The malfunctioning first indenture item is taken to a base maintenance shop and a determination of repairability is made. If the item can be repaired, it is scheduled into base repair and at some later time, when fixed, it is sent to base supply, where it is used to satisfy an outstanding backorder, if any, or is added to serviceable supply on the shelf. If not base-repairable, it is sent to the depot and a resupply request is levied on the depot. After some resupply delay, the length of which depends on the situation at depot supply, a serviceable unit is received by base supply. Note that usually a different unit of the item is received from the depot than the one sent to the depot. One of the complications of the repairable item theory is that these repair and resupply delays are not fixed; there is a probability distribution for the time to repair an item at the base, depending on the complexity of the repair and the availability of personnel, shop equipment, and spare parts. The order-and-ship time is defined to be the time from placing a request on depot until the time when the item is received at the base if there was stock on the shelf at the depot. There is a probability distribution for this time as well. There is also a probability distribution for the waiting time at the depot until an item is available to ship to a base. All of these probability distributions must be taken into account by our theory. (Sherbrooke 2004)

Figure 78 offers a graphical interpretation of Sherbrooke's textual description. One observation is that the "physics" description does not mention what happens with condemnations (scraps) or new item procurements. These concepts are covered by Sherbrooke (2004) in later chapters where he notes that condemnation rates for first

indenture items is quite low, and the procurement of new units happen as to replace condemnations as they occur.

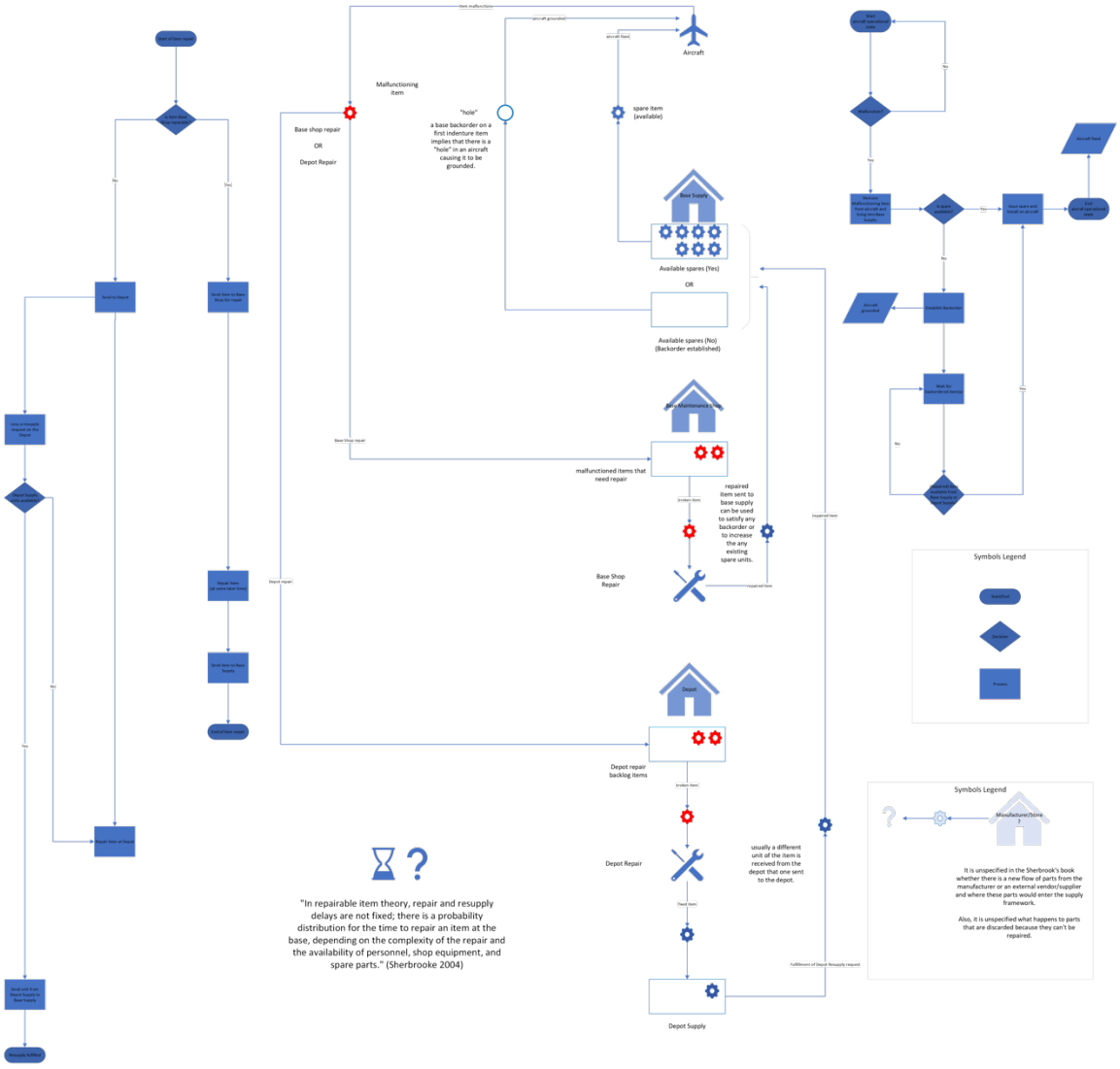


Figure 78. "Physics" of the problem. Adapted from Sherbrooke (2004, 6-7).

For comparison, other graphical depictions of the "physics" behind the multi-echelon model theory, are shown in Figures 79, 80, 81, and 82.

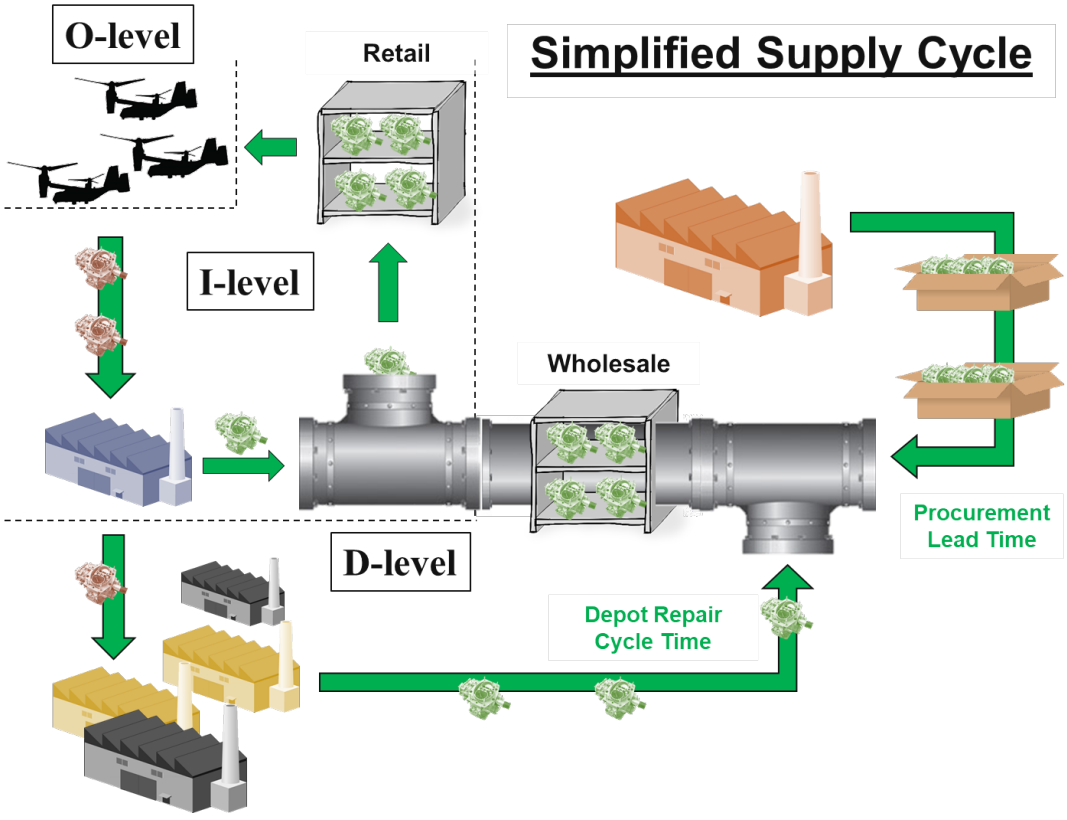


Figure 79. Multi-Echelon pipeline model to enable "end-to-end simulation." Source: Institute for Defense Analyses (2023b).

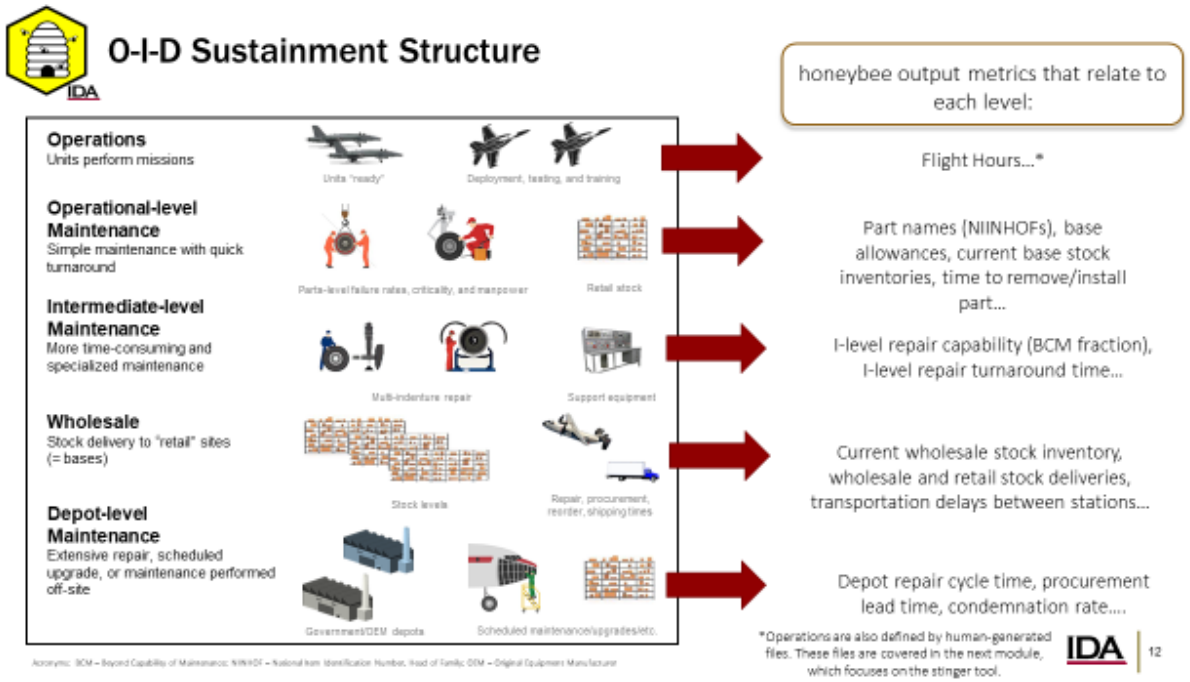


Figure 80. O-I-D sustainment structure. Source: Institute for Defense Analyses (2023a, 12).

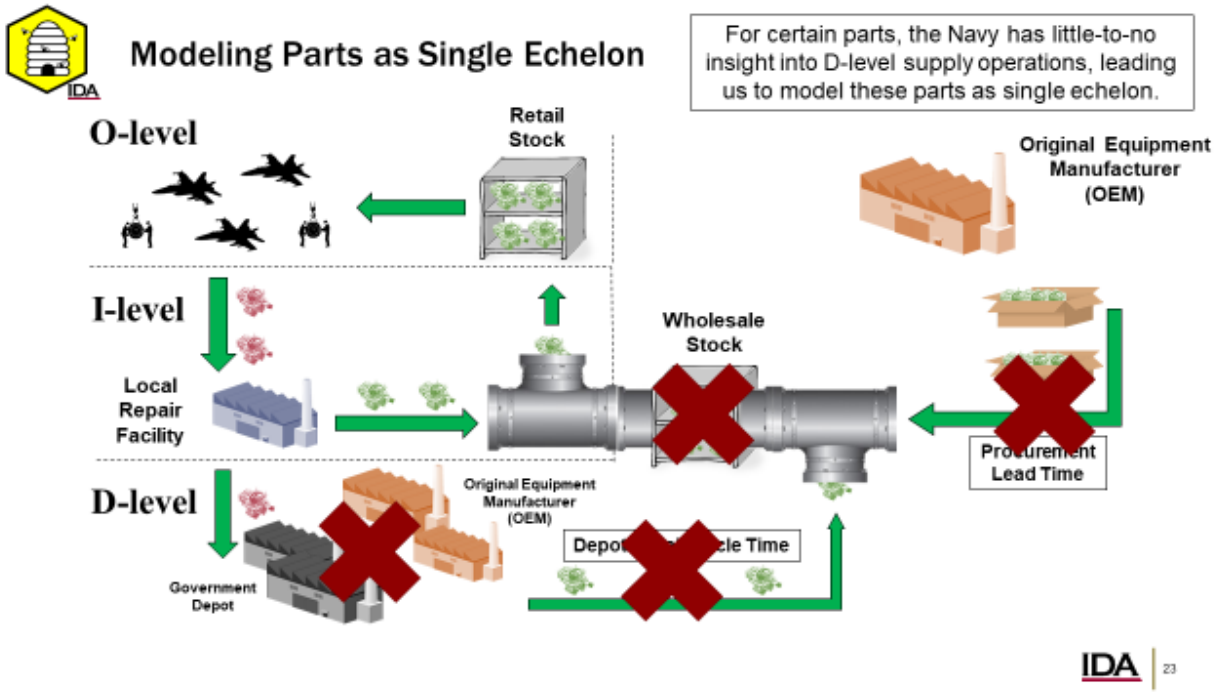


Figure 81. Modeling parts as a single echelon. Source: Institute for Defense Analyses (2023a, 23).

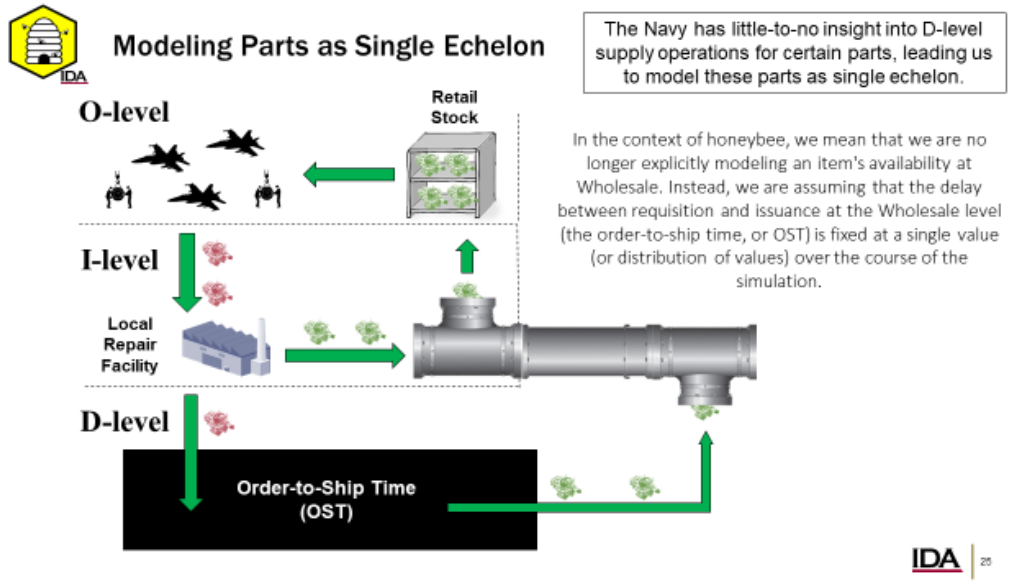


Figure 82. Modeling parts as a single echelon (continued). Source: Institute for Defense Analyses (2023a, 25).

a. METRIC: A Multi-Echelon Technique for Recoverable Item Control

METRIC is a 1960s mathematical model of a two-echelon, base-depot supply system. Item demand is represented by a compound Poisson with a mean value estimated by a Bayesian procedure (Sherbrooke 1966;1968). "METRIC was the first practical application of multi-echelon inventory techniques, and it has formed the theoretical foundation for a number of models used by the military services of the United States and its allies as well as a number of private companies" (Sherbrooke 2004). The following is a simplified explanation given by Sherbrooke (1968):

> When a unit fails at base level there is a probability *r* that it can be repaired at the base according to an arbitrary probability distribution of repair time, and a probability 1-*r* that it must be returned to the depot for repair according to some other arbitrary distribution. In the latter case the base levies a resupply request on depot. No lateral resupply between bases is considered in the model. For high-cost, low- demand items the appropriate policy is (*s*-1, *s*), which means that items are not batched for repair or resupply requests. This problem has a simple analytic solution that is a function of the mean repair times rather than the repair time distributions (Sherbrooke 1968, 122)

Sherbrooke implemented the mathematical model as a computer program that can evaluate any distribution of stock, compute the optimal redistribution of stock, and minimize expected backorders for any system investment (Sherbrooke 1968). Computing power and time was an important consideration at the time, so a balance between simplicity and precision was made. For an in-depth description of the model and equations, see the reference on Sherbrooke's (2004) work.

b. VARI-METRIC: Improved Approximations for Multi-Indenture, Multi-Echelon Availability Models

METRIC theory serves as the basis of multi-echelon models. Expanding upon this model by including the dependencies between indentures leads to the more accurate VARI-METRIC theory that is able to handle multi-echelon, multi-indenture problems (Sherbrooke 1986; 2004).

5. The Conceptual "Physics" behind the Multi-Echelon, Multi-Indenture Inventory Problem

Sherbrooke (2004) provides a description of the processes that occur in a multi-echelon, multi-indenture inventory problem:

> The combined multi-indenture, multi-echelon process begins when an LRU fails and is brought into base supply. If base supply has a spare LRU, it is issued; otherwise a base LRU backorder is incurred. The failed LRU has a probability of being repaired at the base; otherwise, if the repair is too complex, the LRU is sent to the depot for repair and a resupply request for the LRU is placed on depot….
>
> If the LRU is repaired at the base, we assume that one and only one SRU will be found to have failed. If a spare SRU is available, it is put on the LRU and the LRU repair is completed. The SRU has a probability of being repaired at the base; otherwise the SRU is sent to the depot and a resupply request for the SRU is made on the depot….
>
> When an LRU repair/resupply is completed, a backorder is satisfied if there are any outstanding; otherwise, the LRU stock on hand is increased by one. If the LRU is not repaired at the base, a similar process for SRU repair occurs at the depot. We assume that all SRUs can be repaired at the depot. (Sherbrooke 2004, 106)

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY AND PRESENTATION OF DATA AND RESULTS

This chapter presents the methods used by the author to develop the M&S VV&A model. This model is the primary artifact representing the M&S VV&A framework. The VV&A framework consists of the V&V elements and their relationships to the M&S system architecture entities. In essence, the M&S VV&A model is no different than a regular system model, but for this application, the system *is* an M&S product. Modeling the M&S system architecture provides a major part of the value. The remaining part of the value is provided by modeling V&V elements and their relationships to the M&S system architecture. Therefore, developing an M&S VV&A framework is equivalent to developing an M&S system model that incorporates the necessary V&V elements and relationships. Once developed, the M&S model serves as the guiding map or the framework that shows all of the V&V elements, such as methods or actions, and their trace or allocation to M&S entities. The author calls this an M&S VV&A framework because the model identifies: M&S entities, V&V entities, and their relationships, all structured and organized according to some ontological framework—in this case the MagicGrid v.1 Framework. V&V elements are the primary focus of the full VV&A domain; V&V is of key importance because accreditation is usually a result of successful M&S V&V. Also, V&V serves as the foundation for accreditation.

For each modeling task, the author provides a section where the author describes: the general modeling objectives and guidance—to help others who may wish to use this methodology; a table of the information or data collected by the author to support modeling tasks; the modeling steps and results; and proposed future work.

1. What the M&S VV&A Model Is, and What It Is Not!

The M&S VV&A model is a descriptive (MBSE) model of the Cost Readiness Impact Model (CRIM) implemented in Cameo System Modeler (v.2021.x) and expressed in SysML using the MagicGrid v.1 framework. Therefore, the M&S VV&A model is equivalent to the CRIM MBSE model, which is a more precise name. CRIM is more than

just a model. CRIM is an open system, meaning that CRIM interacts with external accompanying systems, such as naval databases, computer data entry applications and users or actors. The descriptive (MBSE) modeled developed in this research captures CRIM's technical baselines and system architecture.

The CRIM MBSE model is **not** a model of the real-life Fleet Operations and Maintenance (O&M) system. This confusion should be avoided. Figure 82 illustrates this point by showing that the CRIM conceptual model is withing the subject of interest domain for the developed MBSE model—not the real-life system, which is the fleet. Figure 83 is a bit misleading. The MagicGrid framework domain of interest is broader than just CRIM's conceptual model. The MagicGrid framework views CRIM from a system's viewpoint. Therefore, additional information of details of CRIM, as a system, are captured.

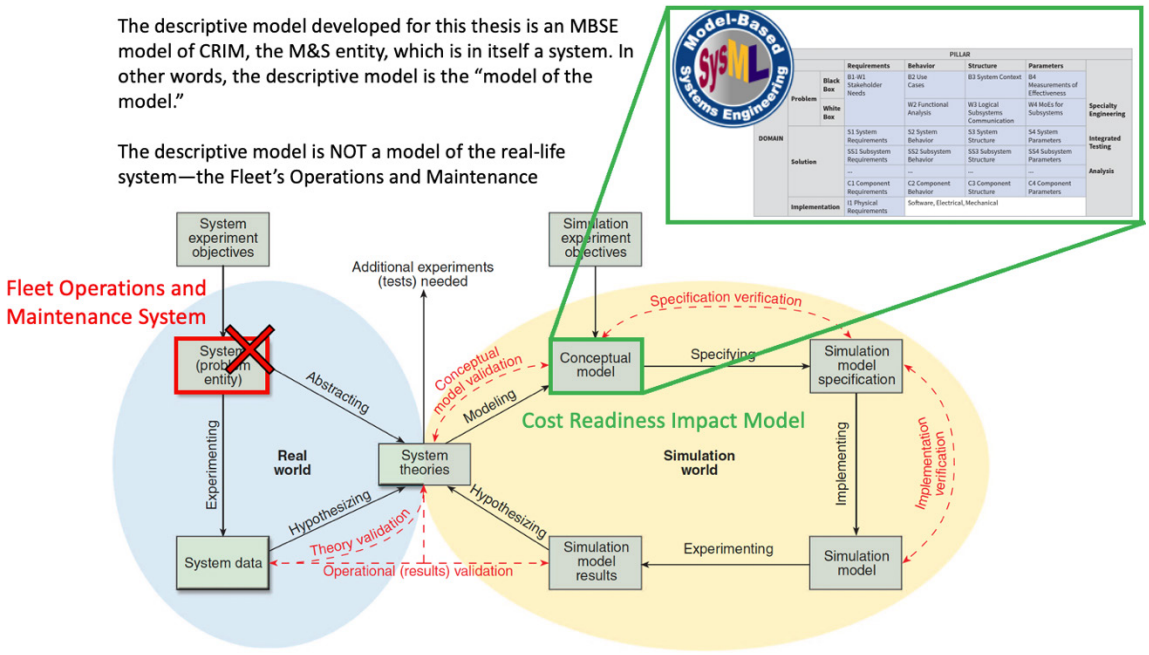


Figure 83. CRIM MBSE model domain of interest. Adapted from Pace (2004).

CRIM **is** directly based on the fleet's O&M system. CRIM is composed of a set of two quantitative models, the Driver Tree model and the SIMLOX based discrete event Monte Carlo simulation model. These models are directly based on the fleet's O&M

entities and relationships, but their structure, inputs and outputs are numerical in nature as opposed to descriptive.

It would be ideal to have a descriptive model of the fleet O&M system! There would be several benefits to this. A descriptive model of the fleet O&M system could directly capture all the entities and physical processes of interest. Enhancements or refinements to the CRIM quantitative models could be made with the information available in the fleet O&M system's descriptive model. Additionally, there would be value in comparing the entities and physical processes modeled in the fleet O&M system against the entities and physical processes captured in CRIM's descriptive model.

2. Goals of this Methodology

a. Generic goals:

- Capture the domain of interest, as fully as possible.
- Provide a useful descriptive mapping of CRIM's entities and processes so that VV&A plans and actions can be traced to the respective parts of CRIM.
- Refine the domain of interest: remove ambiguity using the SysML language and the use of systems theory concepts to facilitate the abstraction and encoding of CRIM information into the descriptive MBSE model.
- Achieve a "discrete" level of decomposition required to understand the functions of the system and the components that perform the functions. "Discrete" in this case means that a level of decomposition is achieved that supports necessary V&V activates—this level of decomposition may be subjective to what is deemed as useful or appropriate by the stakeholders.
- Facilitate the modeling and traceability—this is a property granted mainly by the modeling tool.

- Manage CRIM's technical baselines, such as the requirements baseline, the functional baseline, and the allocated baseline.

b. Specific Implementation goals:

- Model CRIM's system boundaries in such a way that is helpful with partitioning and isolating CRIM's entities and their functions so that V&V planning and activities can be managed easier, with better visibility and organization.
- Model CRIM at the system-level.
- Model CRIM at the subsystem-level.
- Model CRIM at the component-level.

3. When Is the MBSE Model Good Enough, Or What Is The Required Level Of Detail?

With respect to using graphical or descriptive models to support M&S validation (primarily M&S conceptual validation), Sargent (1986) stated that the level of modeling decomposition can be taken down to the algorithm and pseudo code level. The level of detail that the MBSE model captures must be decided by the V&V needs of the stakeholders. The author believes that there are likely diminishing returns or benefits to modeling down to the algorithm or pseudo code level. Instead, the author proposes that the stakeholders identify the appropriate level of M&S system boundaries or partitions that have inputs and outputs that are of interest and importance to the simulation model's operating principles, concepts, assumptions, and results. This may happen at various levels or nested layers of the simulation model. When those M&S boundaries are determined, the entities inside the boundary may be treated like a black-box and assigned a described function or process. This may be as high as the system-level or down to its more discrete partitions. The primary value is that the mapping will allow stakeholders to identify which M&S entities are important, the functions of the entities and the inputs and outputs value types—all of these captured as MBSE model elements. The stakeholders can then, one-by-

one, identify and prescribe appropriate V&V techniques for the respecting M&S entity. If the MBSE model is captured appropriately, it can serve as the main central source for communicating and planning V&V efforts that must be applied to the simulation model.

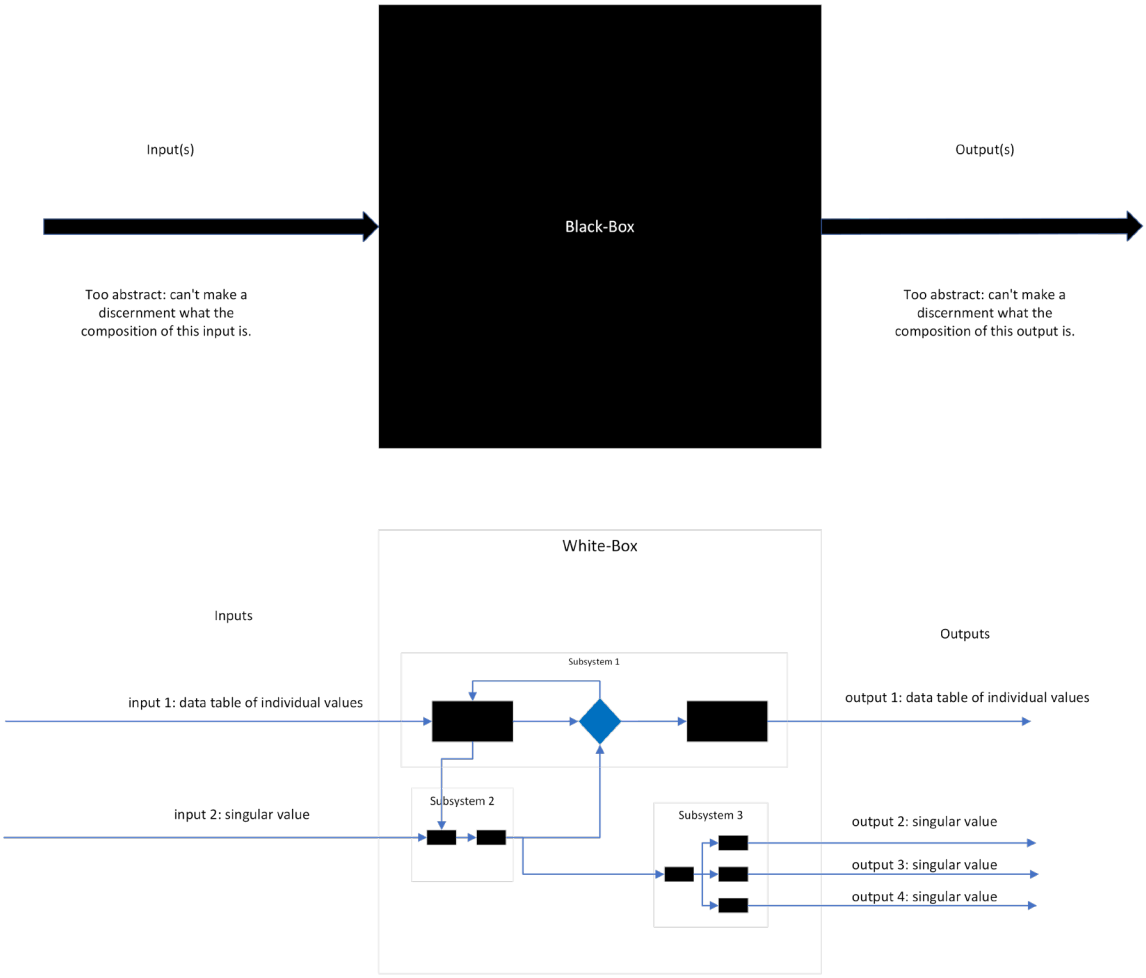


Figure 84. M&S system boundary partitions.

Figure 84 illustrates how the system can be decomposed so that system boundaries may be identified. Decomposed system boundaries or partitions are helpful because they provide a precise mapping of boundary relationships, i.e., the inputs, the function or process that transforms the inputs into outputs, and the outputs. This mapping allows stakeholders to organize, *isolate*, and pinpoint M&S entities at various levels and assign V&V plans or required actions. A black-box assessment can be made at the system black-

box boundary level or at any lower black-box boundary level down to the most concrete level—if the boundaries, inputs/outputs, and functions can be described without ambiguity down to the most discrete or concrete level. However, in order to fully understand the overall system, subsystems, or components behavior, the discrete set of functions, inputs and outputs and their discrete values must be known—this happens at the lowest (appropriate) level of decomposition.

Data input and outputs can be abstracted in the MBSE model, but to be useful, these abstract entities should identify or reference a more *concrete entity* such as a data table where the discrete values can be obtained for V&V or other purposes.

4. "Discrete" Descriptive Modeling

The term "discrete" used in this context is more in reference to the qualitative properties of a numerical entity rather than the numerical properties. The Merriam-Webster dictionary provides two useful definitions: (1) "constituting a separate entity: individually distinct," and (2) "consisting of distinct or unconnected elements."

In a descriptive context where the focus is to describe entities and their relationships, a "discrete" value is a singular numeric value entity that cannot be decomposed without losing its identity and that represents a characteristic or metric of an entity in the model or system—this context does not relate to discrete (integer) or continuous (non-integer) numbers. For example:

> In a multi-echelon, multi-indenture inventory model, the monthly number of aircraft part demands is a distinct "discrete" metric defined by its identifying characteristics or conditions, i.e., the aggregate, singular number of observed or calculated periodic demands for a specific type of part, for a time span of one month—if one of these identity conditions is removed, such as the time period is changed to yearly, or daily, this metric loses its identity. This metric may be: (1) observed from the system and captured as system data, or (2) statistically calculated by the model based on system data and used for estimating future month demands where data is not yet observed but a demand value is needed to simulate and forecast future impacts. A demand means that an aircraft part failed, and that the aircraft has a demand on the supply system for that part. The monthly

demand in this case is a derived value; an aggregation of demands for a specific type of part within the month time period.

A "discrete" function has a unique identity and is a single operation or combination of operations that ingests "discrete" values and calculates "discrete" values. For example:

> The freefall velocity equation ($V_f = V_i + gt$) is a unique identifiable formula in physics. It includes input variables for "discrete" input values such as the initial velocity (V_i) in meters per second, the gravitational acceleration "constant" (g) in meters per second squared, and the fall time (t) in seconds. This formula follows an order of operations: (1) the fall time (t) is multiplied by the gravitational acceleration (g), and the product is then summed to the initial velocity (V_i); the completion of all operations is equal to the "discrete" output, the final velocity (V_f) value. Similarly, the final velocity (V_f)., even though it is derived, its identity is also characterized by a single value.

Again, "discrete" is more related to the single "atomic" (non-divisible or decomposed) identity of a value and not whether the value is an integer or non-integer. For example, in the freefall equation, the variables entities such as initial velocity (V_i) are represented by a single value. Figure 85 shows a simple graphical model representation of the freefall equation.

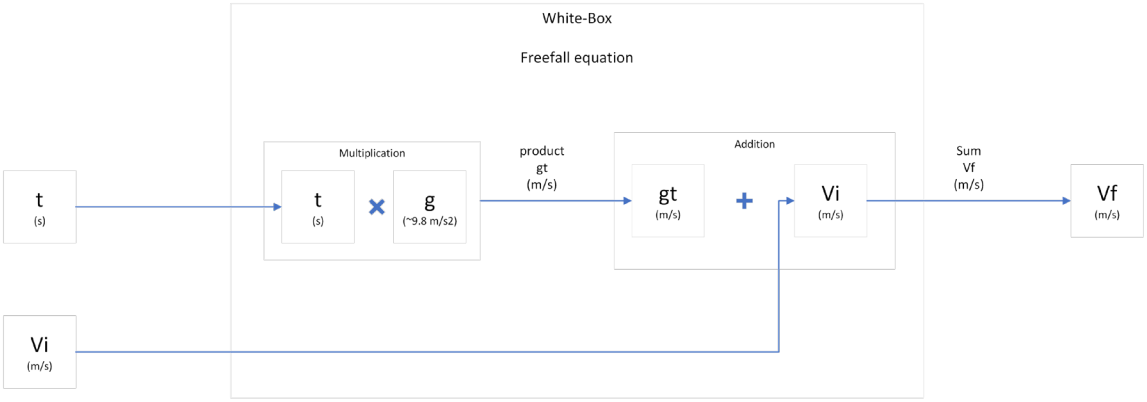


Figure 85. Freefall equation described as a graphical model showing inputs, functions or operations, and a single output.

For convenience, it is useful to have a mental reference of the identifiable entities in an algebraic formula and their nomenclature. Figure 86 illustrates the parts of an algebraic formula; coefficients, variables, constants, and their equation structure are built into a quantitative model. However, for computing purposes usually only the variables are changed. Variables are usually associated with direct (system boundary) or indirect (internal function) inputs—in the case of a chain of formulas.

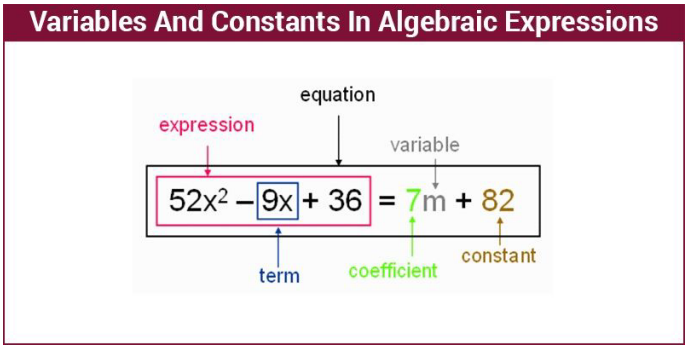


Figure 86. Parts of an algebraic expression. Source: BYJU'S (n.d.).

B. METHODOLOGY

1. Methodology Summary

The author collected a set of information pertaining to CRIM's architecture. The set of information includes graphical information, text-based information, and verbal information. The author collected information from various sources such as CRIM's PM, Jason Thomas, and the developers and integrators of CRIM's models and data pipeline: the Institute for Defense Analyses (IDA), Booz Allen Hamilton (BAH), and IBM, formerly known as International Business Machines. The author collected, organized, and interpreted the information and its underlying semantics so that the information could be translated into SysML model elements. These relationships among the various CRIM design stakeholders are illustrated in Figure 87.

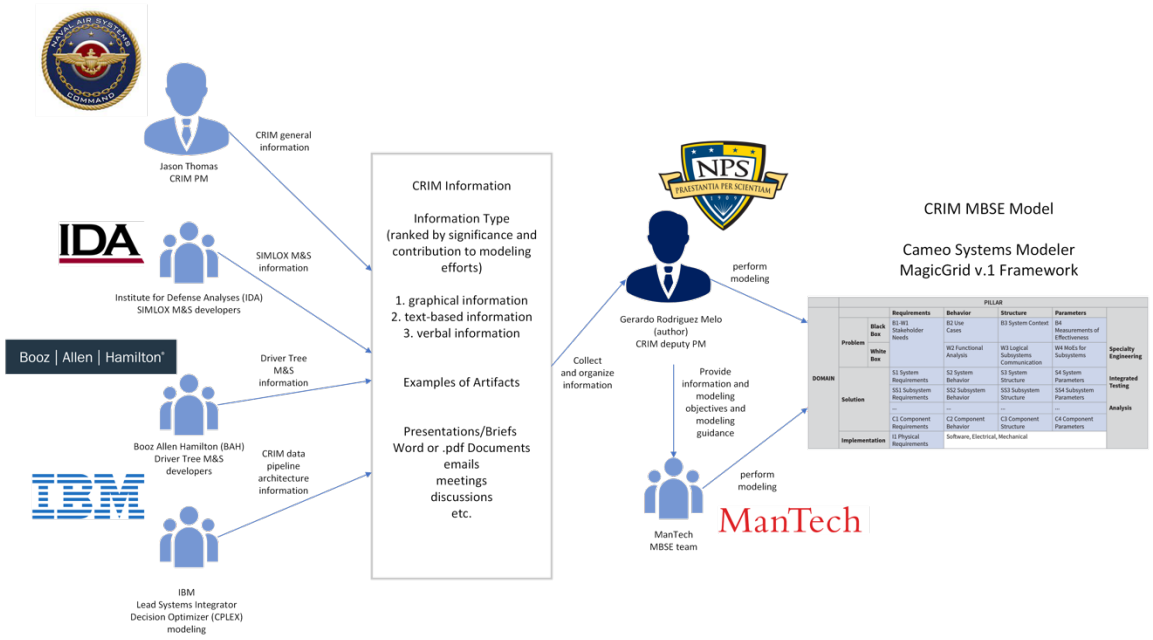


Figure 87. Modeling process summary

After CRIM's information was captured into model elements, to the fullest extent possible with the available information, the author evaluated the information captured in CRIM's MBSE diagrams from an M&S V&V perspective. This evaluation is presented in Chapter IV.

The MagicGrid v.1 framework was used to organize and structure the model elements. The author generated the majority of the model's elements, but modelers from ManTech aided the author in cleaning up the model and probably modeled close to a third of the model with the guidance of the author.

The MagicGrid framework is offered in Cameo Systems Modeler (CSM) as a template, shown in Figure 88.

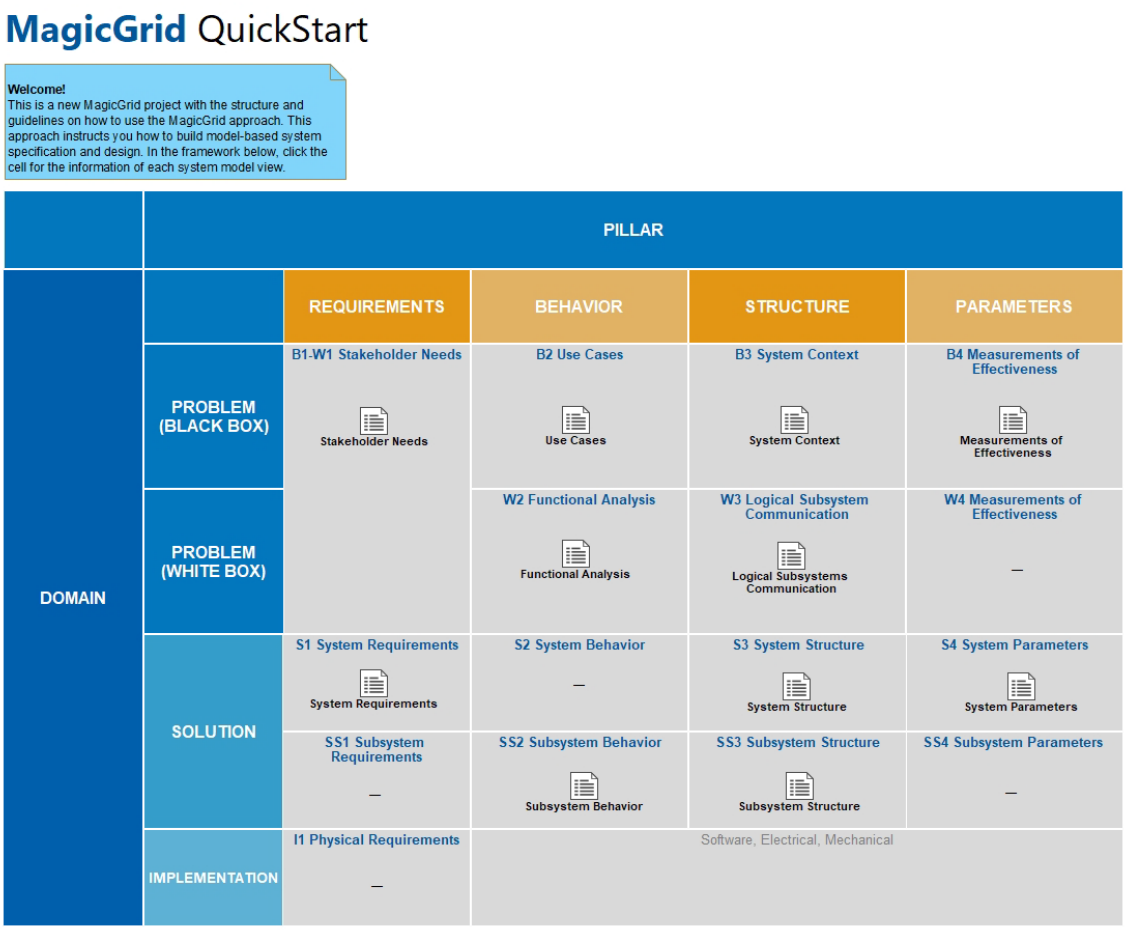


Figure 88. MagicGrid QuickStart template view

The MagicGrid Framework does not specify a certain order for modeling or specifications on how to model, but the structure of the framework provides general guidelines for what information needs to be captured.

The overall modeling process was somewhat recursive, but the steps will be presented in the general logical order in which modeling was performed. Figure 89 shows the overall results of this methodology; the MagicGrid has been populated with the diagrams that were developed as part of the modeling process. Diagrams were created from the underlying model and placed in their corresponding grid. The purpose of these diagrams is to convey visual information to the stakeholders, facilitating communications about CRIM design details and M&S V&V elements.

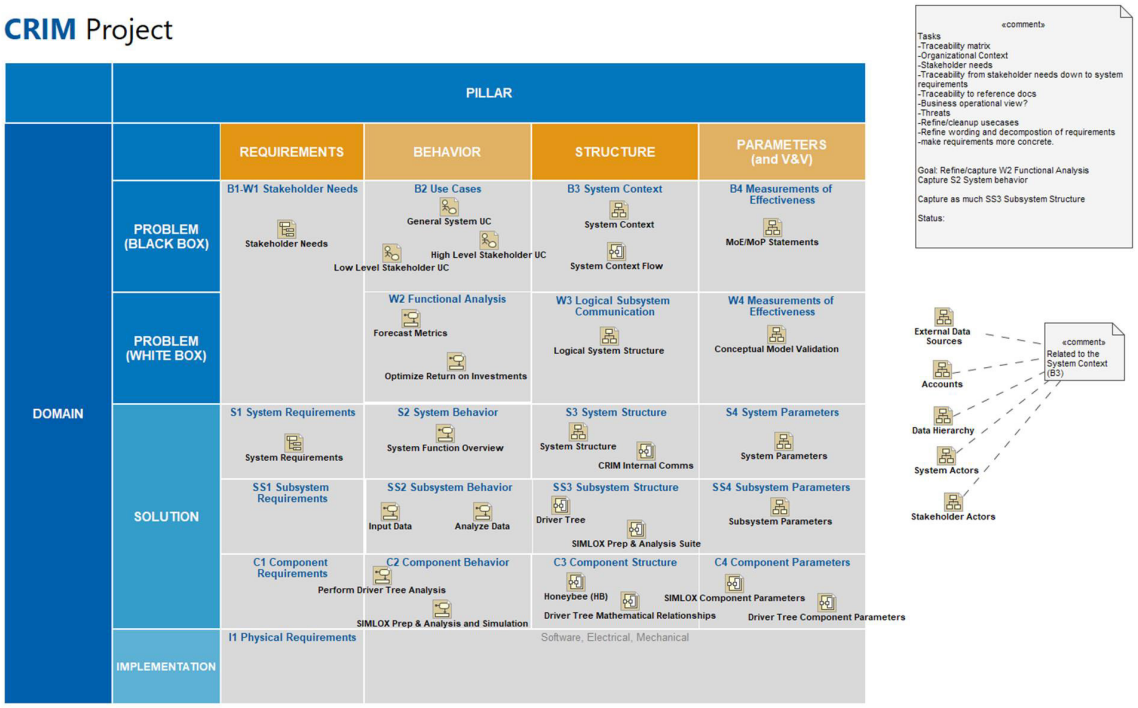


Figure 89. Overall modeling results.

2. Reverse Engineering the Cost Readiness Impact Model (CRIM)

Most of the information gathered by the author represents CRIM's implemented (solution domain) state. At the time of this research, CRIM had already been implemented and was/is operational, meaning that the SIMLOX model and the Driver Tree model could be employed to simulate "modellable" projects. Each SIMLOX model is custom build for each Type/Model/Series (T/M/S) aircraft; some T/M/Ss have already been modeled by the Institute for Defense Analyses (IDA). The problem with CRIM's current technical documentation is that it's missing fundamental design information such as:

- The stakeholder's needs and requirements
- What specifically CRIM must accomplish (outcomes) or be able to model, i.e., what is the domain of things CRIM needs to be able to model or are applicable to CRIM's conceptual models.

- *What* CRIM (as the system) must be able to do from a solution or technology neutral perspective—requires a formal functional analysis.
- Measures of Effectiveness (MoEs) of what CRIM needs to do to satisfy its operational concepts, and Measures of Performance (MoPs)—how well CRIM must perform its effectiveness measures.
- The logical or conceptual model description
- Validation and Verification methods.

This type of information primarily belongs to the "problem domain." Capturing and defining CRIM's problem domain is an objective of this research. The problem domain exists regardless of CRIM or the solution that addresses the stakeholder needs or problems, therefore it is essential to model this domain because it provides the source for CRIM's purpose and requirements. For CRIM to be a valid M&S product, it must address the *right* problem, and conversely, we can know that CRIM is the *right* system only if we understand the problem domain.

The application of this methodology demonstrates the feasibility of applying MBSE to develop CRIM's M&S VV&A framework which identifies and structures VV&A processes in relation to CRIM's entities. This addresses this thesis's *primary research question*:

> Can Model-Based Systems Engineering (MBSE) be leveraged to develop a re-usable framework and process for Validation Verification and Accreditation (VV&A) of Modeling and Simulation (M&S) models?

To be more precise, "MBSE," as mentioned in the thesis question, encompasses the language: SysML; the tool: Cameo Systems Modeler (versions 19.x and 2021.x were used); and the "method": the MagicGrid v.1 framework. Although MagicGrid is primarily a model framework for a system's architecture, it can guide the modeling objectives and activities. Systems theory concepts were used to perform the abstraction needed to model, that is, to take the source information and encode its semantics as best as possible into the model using SysML. Previous education, knowledge and experience in systems

engineering and systems architecting helped guide modeling decisions. In fact, the systems engineering technical baselines, i.e., the requirements, functional, allocated, and product baselines were kept in mind thorough the modeling process. Also, system architecture concepts were heavily used during modeling; concepts such as the solution-neutral functional architecture, the functional architecture, and the physical architecture. These concepts helped in determining the most appropriate way to capture the system's (CRIM's) decomposition of both behavior and structure.

3. Forward Engineering Future M&S Products and Their VV&A Elements

Using the same methodology, one can model a future M&S product or system during the design phase, before it is implemented. The MagicGrid framework allows stakeholders to incrementally capture the M&S design along with the related VV&A elements. This method, approached from a forward engineering mode, would also demonstrate the feasibility of using MBSE to develop the future M&S system's VV&A framework which identifies the M&S VV&A processes in relation to the M&S entities being designed.

This methodology is *re-usable* because it can be applied to essentially any system. According to systems thinking, an M&S product can be viewed and treated like a system. This approach addresses this thesis's *secondary research question*:

- If a new M&S model is being conceptualized or developed, can the framework developed by this thesis help guide and facilitate the design and development of M&S architecture and system requirements?

The author recommends that along with this method, the M&S stakeholders adopt Systems Engineering Technical Reviews (SETRs) to control each of the technical baselines captured in this framework. Figure 90 shows an example of how SETR events can be coordinated to match with M&S design activities carried out using the MagicGrid v.1 framework in an MBSE approach. Stakeholders should appropriately tailor their SETR events to meet their design needs and goals.

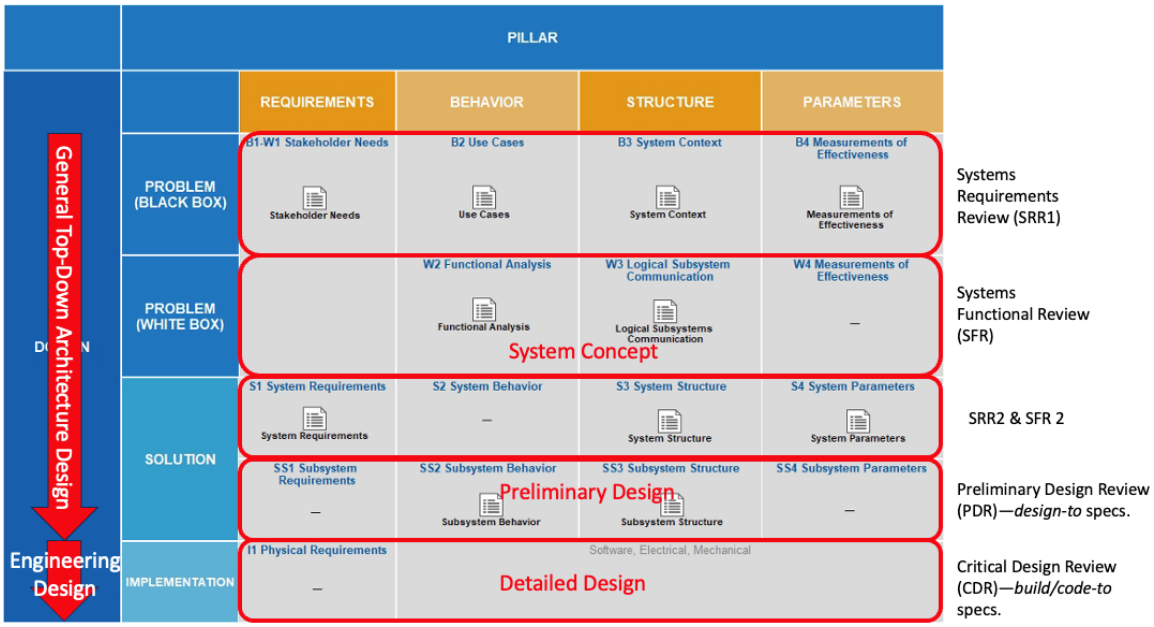
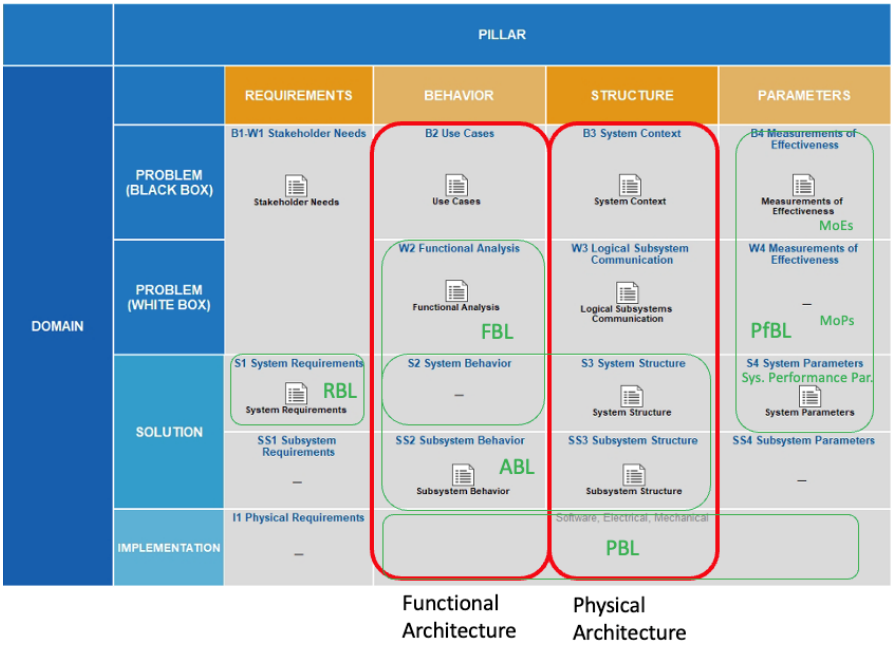


Figure 90. Proposed SETR events that roughly or loosely coordinate with the MagicGrid v.1 Framework. Adapted from Aleksandraviciene and Morkevicius (2018)

Figure 91 roughly illustrates how the system's baselines and the system's architecture overlap with the MagicGrid v.1 framework. This is only notional and not absolute. Stakeholders should define their baselines according to what aspects or details of the design they wish to control.



RBL: Requirements Baseline; FBL: Functional Baseline; PfBL: Performance Baseline; ABL: Allocated Baseline; PBL: Product Baseline. Adapted from Aleksandraviciene and Morkevicius (2018)

Figure 91. Rough approximation of the system's baselines overlap with the MagicGrid v.1 framework.

4. Modeling of Stakeholder Needs

The context here is that stakeholders need to make business or managerial decisions. M&S can be an effective tool aiding data-driven decision making that allows stakeholders to make more informed and better decisions. The subject modeled here, The Cost Readiness Impact Model (CRIM), seeks to provide insights that will allow stakeholders to better quantitatively understand the fleet *availability* impacts given a specific investment, such as funding a project to improve a part's reliability so that the new fleet baseline incorporates this improvement. The impacts of multiple investments can be simulated, and their costs and overall budget constraint can be used to identify the set of investments that yield the optimal selection of investments, i.e., the investments that give the best return on investment (ROI) value. This is tied to the fleet's *affordability* due to the limited budget.

a. Modeling Objectives and Guidance

For this task, the objective is to capture stakeholder needs. The stakeholders and the organization(s) they represent have a specific set of business or managerial objectives they must accomplish through a set of essential tasks. These high-level needs or tasks should capture the essential technology or solution-neutral needs of the stakeholders—these are actions or outcomes that the stakeholders must accomplish regardless of *how* they are accomplished. From this perspective, needs may be stated as the outcomes or impacts the stakeholders must make and are related to their standard business processes that align and support a focused business domain set of objectives.

In systems engineering, the most common format to encode this information is a tabular or matrix format. The MagicGrid v.1 QuickStart template offers a couple of template guide options: (1) is a stakeholder's need table, and (2) is a traceability matrix that helps the modeler capture the traceability of functions and Measures of Effectiveness to their Stakeholder needs. The MagicGrid v.1 QuickStart template for Stakeholder Needs is shown in Figure 92.

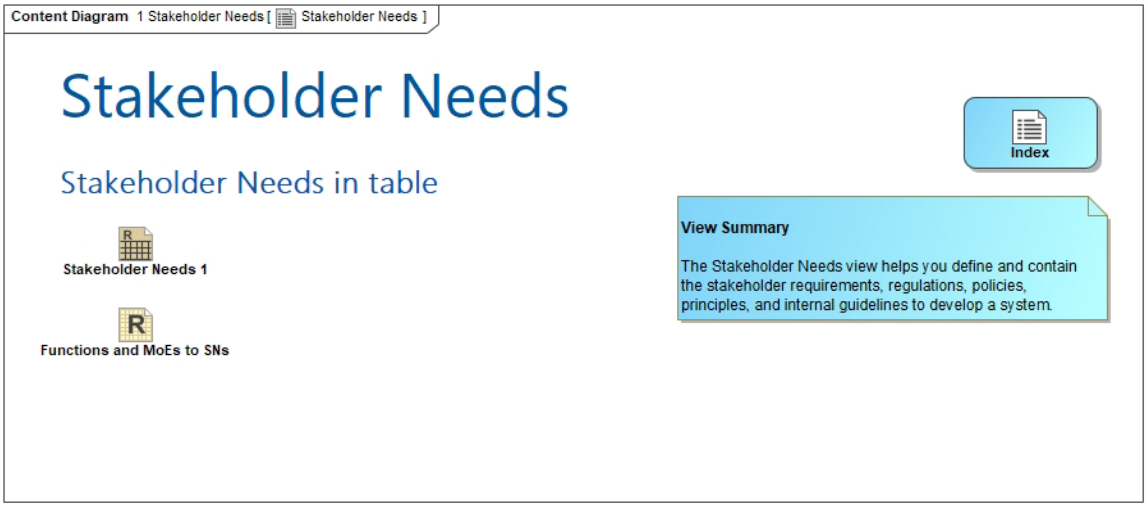


Figure 92. MagicGrid v.1 QuickStart template guide for Stakeholder Needs. Image from CSM v.2021.x.

b. Information/Data Collected

Table 7. Stakeholder Needs Information and Data Collected

Information Gathered	Artifacts/ Information Mode	Source
Stakeholder Needs information	Verbal: CRIM discussions	Jason J. Thomas
	Textual: Meeting notes	CRIM project meetings, various stakeholders
	Mental model inferred by deductive reasoning	The author

c. Modeling Steps and Results

The MagicGrid v.1 QuickStart Template recommendations were not followed. Instead, the author opted for using a *requirements diagram*. Some potential benefits of using a requirements diagram and modeling stakeholder needs as requirements may be that requirements can also be shown on other diagrams for better reference. A requirements diagram may also offer added flexibility in the future by providing the use of the following requirements relationships: *containment*, *trace*, *derive requirement*, *refine*, *satisfy*, and *verify*.

The results of the modeling actions are shown in Figure 93. The *derived requirement* relationship is used to show that some captured need (captured as a requirement) at the tail (client) end is derived from another need (requirement) at the arrowhead (supplier) end. For example, "Stakeholder Need 4" is derived from "Stakeholder Need 1." Additionally, comments are captured to help provide modeling guidelines, information, or reminders.

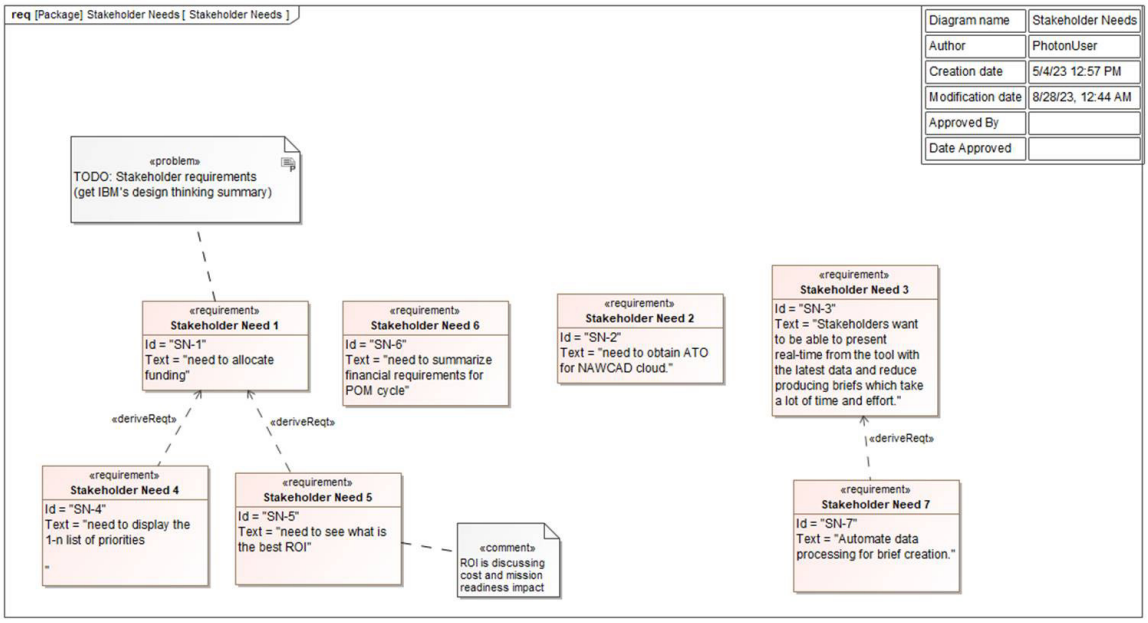


Figure 93. Stakeholder Needs Modeling Results

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Precisely identify the traceability of the Stakeholders Needs to their respective stakeholders. The author has limited knowledge as to who these needs belong. Likely stakeholders would be personnel at the Program Management Office-Air (PMA) level, Program Executive Office (PEO) Level; NAVAIR executive leadership level, and NAVAIR commander (AIR-00) level.
2. Perform Stakeholder Needs definition workshop with stakeholders (highly recommended).
3. Review captured Stakeholder Needs with stakeholders (highly recommended).

5. Modeling CRIM Use Cases

a. Modeling Objectives and Guidance

The objective of this modeling activity is to capture CRIM's use cases. An actor should interact with CRIM directly and invoke one or more use cases. Use cases can be refined from the stakeholder needs. The MagicGrid v.1 QuickStart template provides general guidance for developing system use cases, shown in Figure 94. A use case should be related to an essential use of the system. The completion of the use case should result in some system-level effect or outcome that is of utility or value to the user and supports some user need.

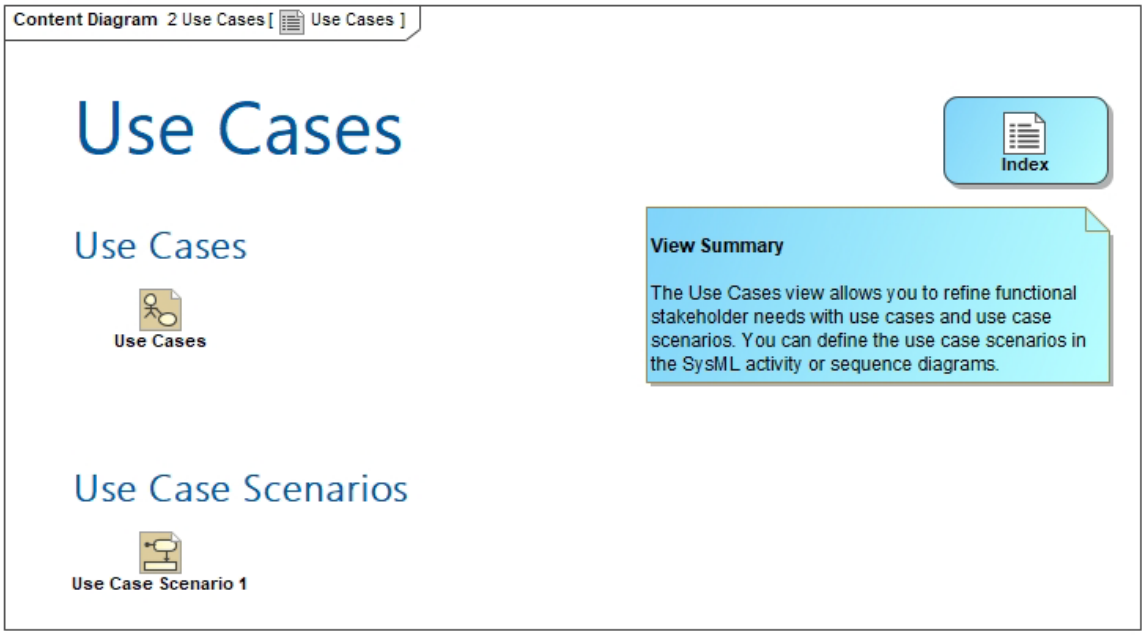


Figure 94. MagicGrid v.1 QuickStart template guide for Use Cases. Image from CSM v.2021.x.

b. Information/Data Collected

Table 8. Use Case Information or Data Collected

Information Gathered	Artifacts	Source
Use Case Information	Verbal: CRIM discussions	Jason Thomas
	Innoslate Use Case Diagram	Jason Thomas
	Textual: Meeting notes	CRIM project meetings, various stakeholders
	Mental model inferred through deductive reasoning.	The author

c. Modeling Steps and Results

Three use case diagrams were produced: shown in Figures 95, 96, 97 and 98:

- General Use Case diagram: A general use case was developed to show the essential use cases of CRIM as the system. This use case abstracts the CRIM user to a block, conveying that some CRIM user invokes some use case captured. Once the use case is invoked and starts executing its scenario, there may be other users that must interact with CRIM to support the completion of the use case. However, this is abstracted. These use cases are modeled from a reverse engineering perspective. Like a work breakdown structure (WBS), these general use cases seek to consolidate or lump together all of the major actions or uses of CRIM; these should try to capture most or all of the underling CRIM internal functional architecture. Sub-behavior elements should ideally be able to trace back up to a use case.

- High-Level Stakeholder Use Case diagram: the high-level stakeholder use case diagram was generated to address the question of "*where do I see myself in the use case context of CRIM?"* from the perspective of a high-level stakeholder, such as a program manager. A high-level stakeholder will likely not interact with CRIM, but they may indirectly invoke a use

case or have a narrower set of use cases. The high-level stakeholder use case gives visibility to these use cases.

- Low-Level Stakeholder Use Case diagram: The low-level stakeholder use case diagram was generated to address the question of "*where do I see myself interacting with CRIM and for what uses?*" from the perspective of a low-level stakeholder, such as a subject matter expert. A low-level stakeholder will likely interact with CRIM directly. This interaction may be part of a larger use case specification belonging to one of the general use cases.

The high-level and low-level use case diagram idea and modeling work is accredited to Levi Herrin, from ManTech. For the low-level stakeholder use cases, reference material was pasted into the canvas to facilitate modeling and to provide insights as to what type of details should be captured.

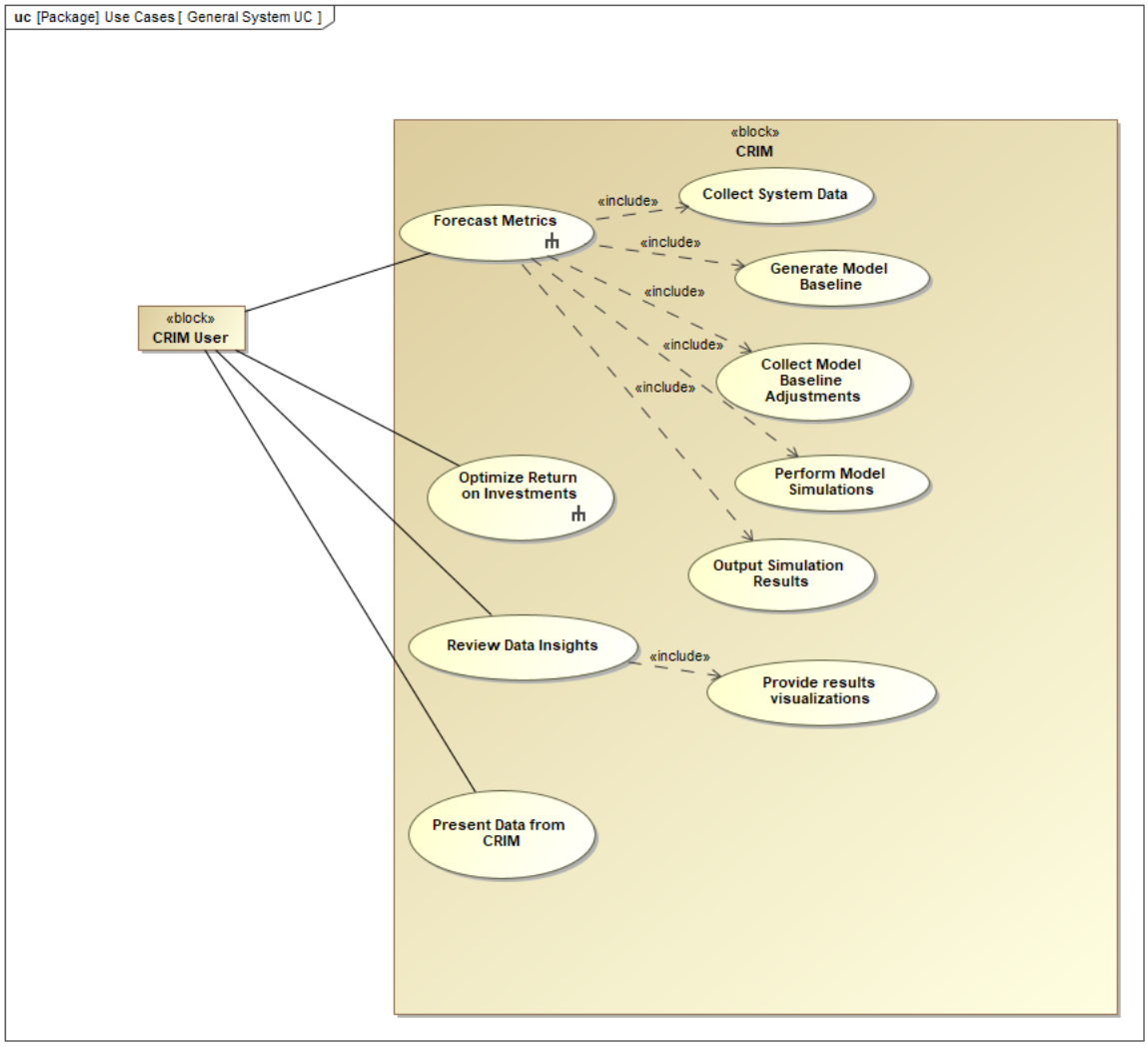


Figure 95. General Use Cases modeling results.

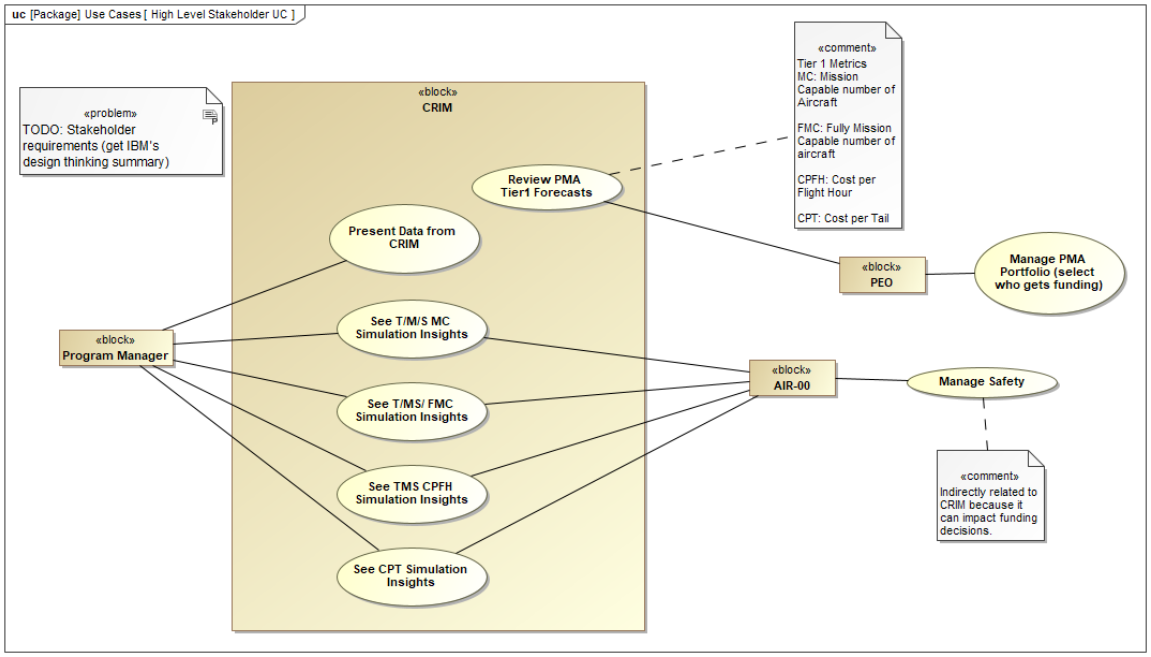


Figure 96. High-Level Stakeholder Use Cases modeling results

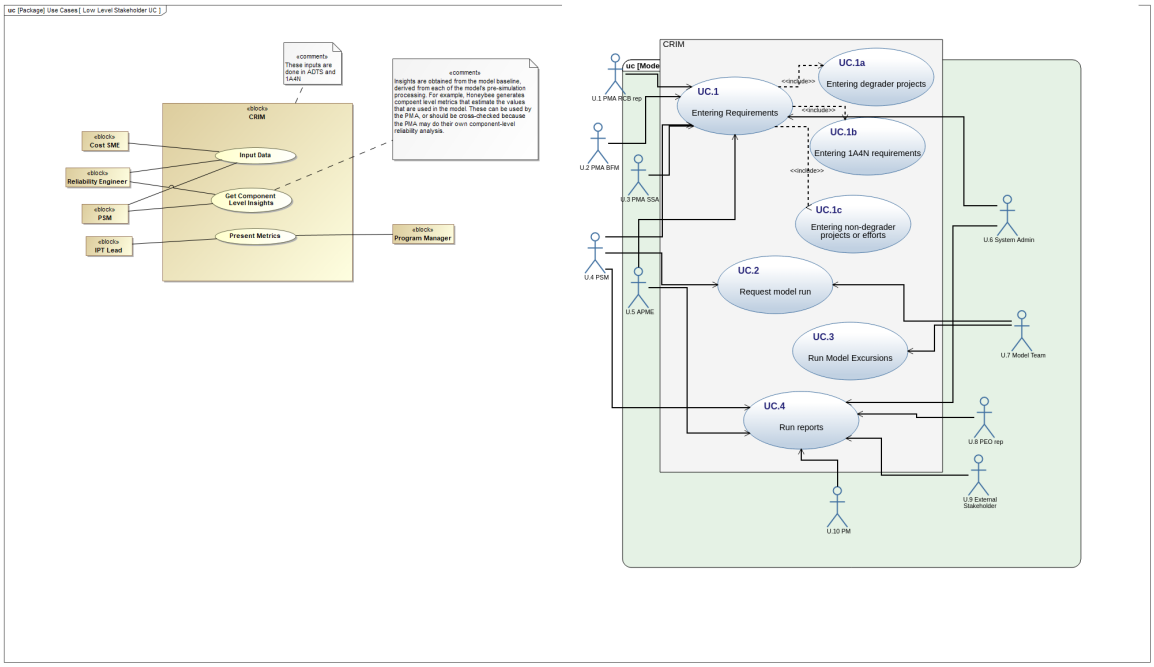


Figure 97. "Low-Level Stakeholder" use cases modeling results with pasted reference artifacts (Innoslate CRIM Use Case Diagram). Zoom in to see small details.

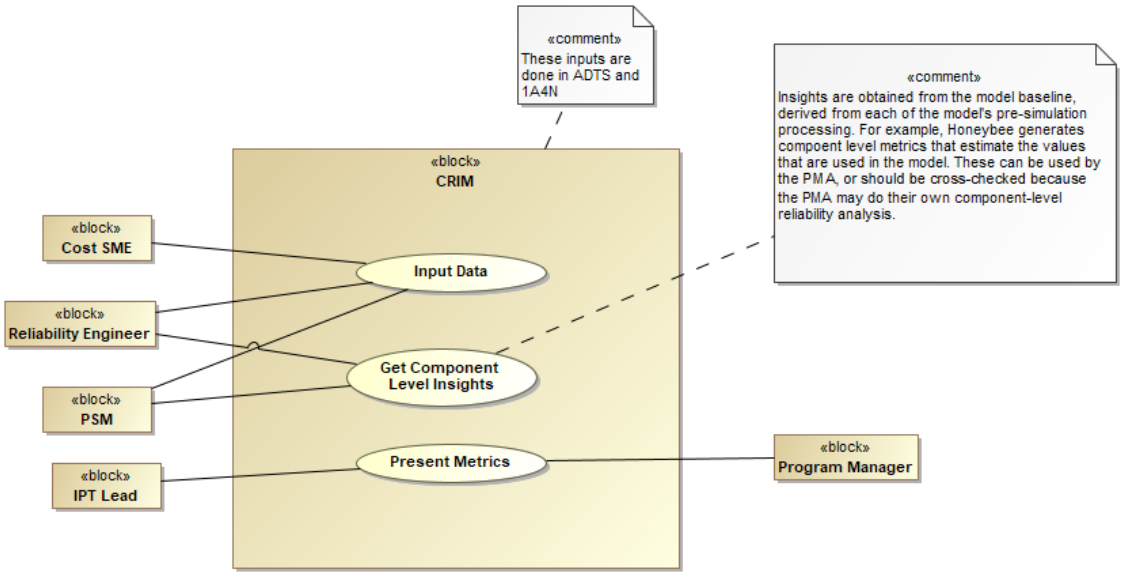


Figure 98. Low-Level Stakeholder Use Cases modeling results. SysML model elements only.

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Review and validate modeled use cases with stakeholders (Recommended).
2. Perform a use case workshop with stakeholders to identify any missing use cases or refine existing ones. (Recommended).
3. Continue gathering information and refining the model.

6. Modeling CRIM's System Context

a. Modeling Objectives and Guidance

The goal of the system context is to model important structural and interaction details between the system, its environment, and accompanying systems. The MagicGrid v.1 QuickStart template provides a guide for modeling the system context, shown in Figure 99.

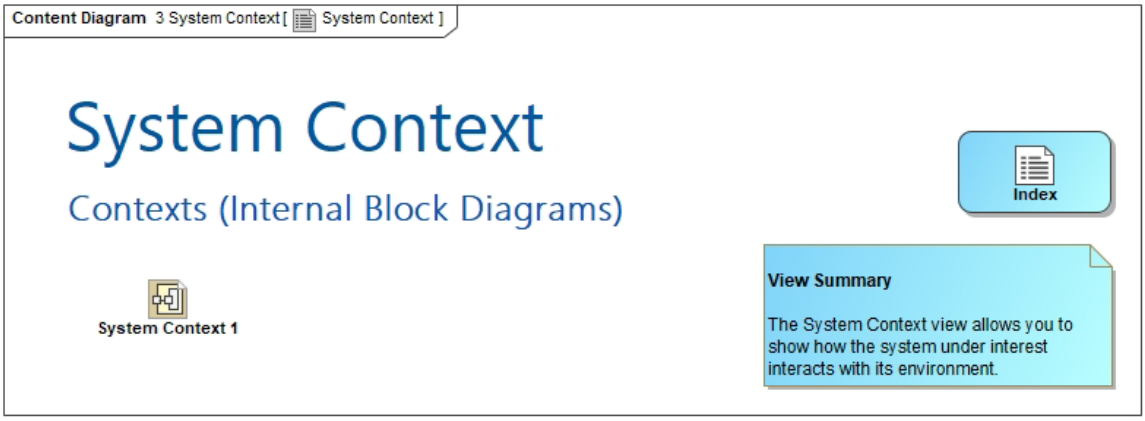


Figure 99. MagicGrid v.1 QuickStart template guide for the System Context. Image from CSM v.2021.x.

b. Information/Data Collected

Table 9. System Context Information or Data Collected

Information Gathered	Artifacts	Source
System Context Information	Mental model inferred from general CRIM architecture documents	IBM and Jason Thomas
	Verbal: General CRIM architecture discussions	Various CRIM project meetings and discussions

c. Modeling Steps and Results

Two diagrams were developed, shown in Figures 100 and 101. A block definition diagram (bdd) was generated showing the structural decomposition of the system context. The system context is composed of "CRIM," the system of interest; the users, abstracted as the "CRIM User"; and the accompanying systems, abstracted as the "Data Source."

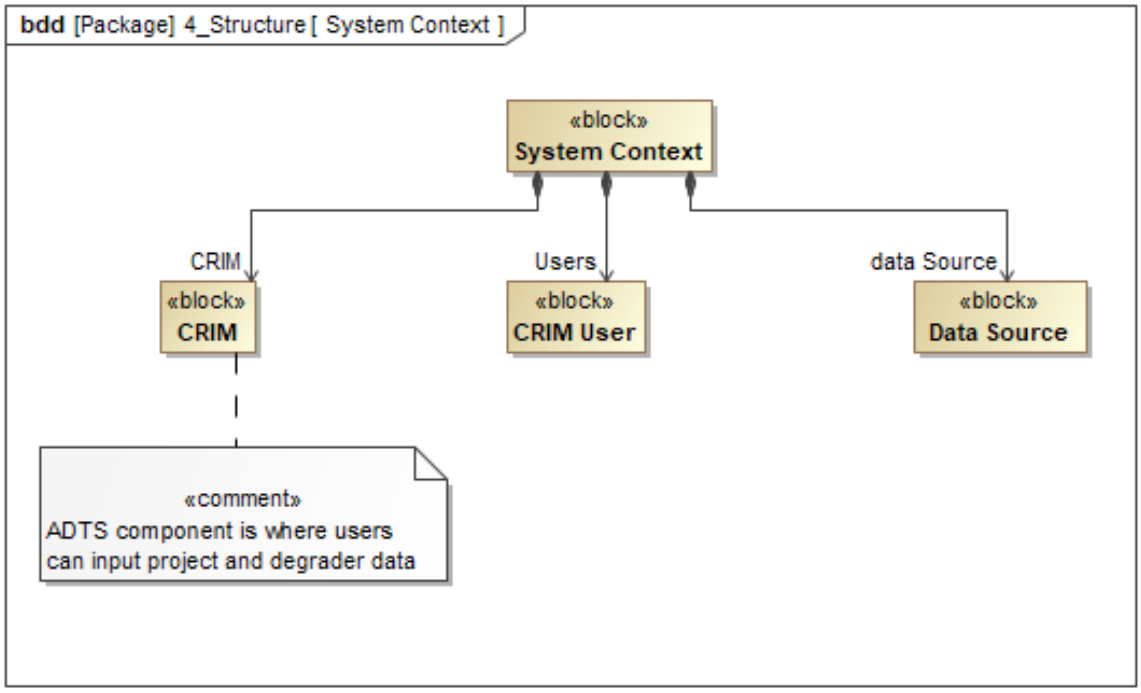


Figure 100. "System Context" block definition diagram showing the entities that compose the system context in its entirety.

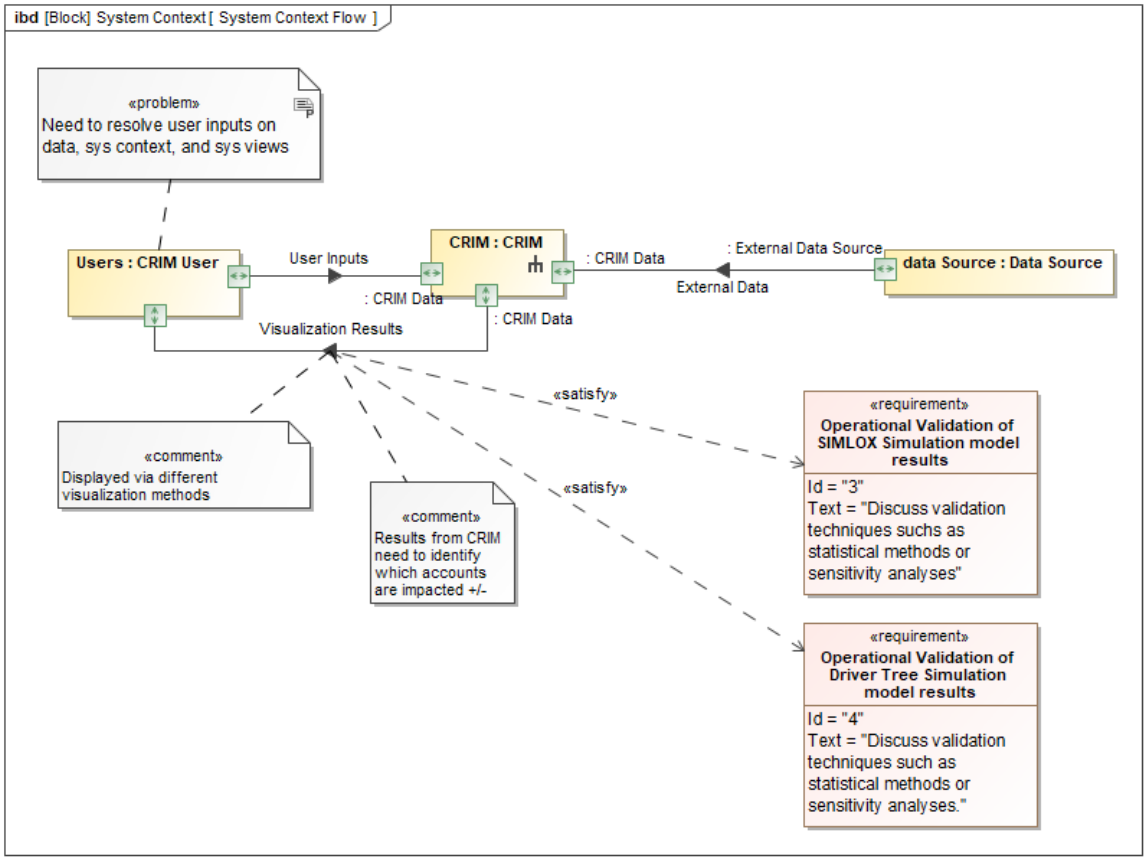


Figure 101. System Context internal block diagram showing the constituent parts of the system context and their connective structure.

d. Future Work

There is no future work required or desired at the moment. The model elements generated for this grid provide a useful foundation of elements that are subsequently used to: (1) decompose all of CRIM's systems elements of interest, (2) decompose all of Data Source or accompanying systems elements of interest, and (3) "decompose" all types of CRIM users or stakeholders.

The abstraction level modeled here is useful because it provides a broad representation for all sub-elements belonging to the main system context elements defied here, i.e., the "User(s)," "CRIM," and "Data Source(s)." this abstraction also broadly covers all of the data flows between these entities.

7. Modeling CRIM's Functional Analysis

a. Modeling Objectives and Guidance

The goal for this step is to capture the system's solution-neutral functional architecture. This states *what* the system must do, not *how* the system is to accomplish its solution-neutral functions. This perspective allows stakeholders to consider the essential system functions regardless of technology.

The MagicGrid v.1 QuickStart template provides a modeling guide, shown in Figure 102. The functional analysis uses the previously defined system use cases as the source and basis for further functional analysis and decomposition. During this task higher-level functions are decomposed into sub-functions.

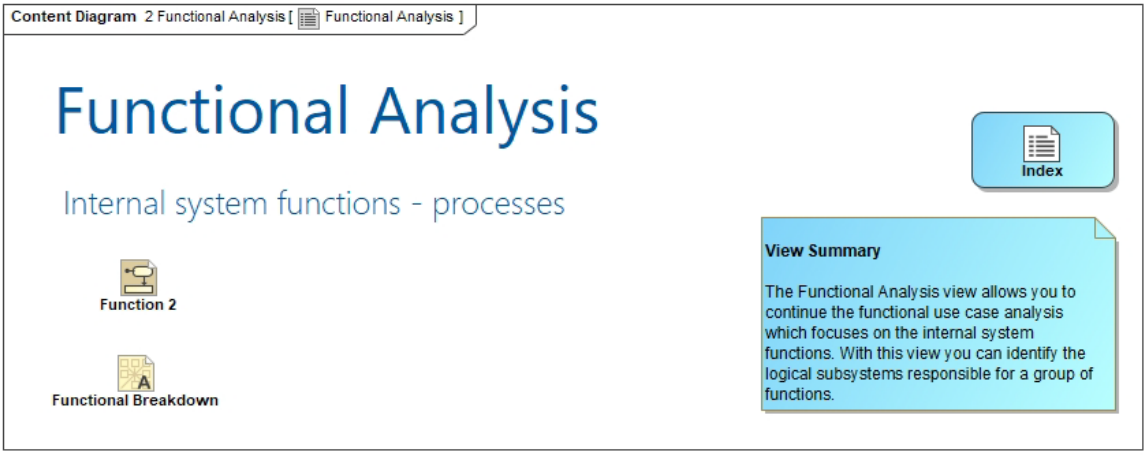


Figure 102. MagicGrid v.1 QuickStart template guide for modeling the Functional Analysis. Image from CSM v.2021.x.

b. Information/Data Collected

Table 10. Functional Analysis Information or Data Collected

Information Gathered	Artifacts	Source
System functional vision/ concept information	Innoslate diagram	Jason Thomas
	CRIM general modeling discussions	CRIM technical meetings
	Mental model inferred from deductive reasoning.	The author

c. Modeling Steps

Two use cases from the "General System UC" diagram are decomposed into activities that further define the respecting use case specifications. The two use cases are the "Forecast Metrics" use case and the "Optimize Return on Investments" use case. The former use case runs the entirety of the end-to-end simulation modeling thread, from "collecting System Data," to "Output Simulation Results." This is the primary use case of interest with regards to validating and verifying the simulation model results. The latter use case is associated with the linear optimization model that is used to rank the return on investment according to a set of projects' investment amounts and simulated availability and affordability impacts.

This functional analysis captures the essential functional patterns for the "Forecast Metrics" and "Optimize Return on Investments" use cases. The functions described are mostly implantation or technology neutral except for the action, "Configure CPLEX Optimization Model." This is already identifying which concept or solution is being used. Ideally this function would be described in a more solution neutral manner, but the modeling decision was made to provide more concrete context. At this point in time or in the future, alternatives to the CPLEX solution may be some other linear optimization program, and thus would not greatly impact the functional steps that compose this use case specification.

Relevant information was copied and pasted into the activity diagram canvas. The results of the "Forecast Metrics" functional analysis is presented in Figure 103. Figure 104

shows zoomed in details of the “Forecast Metrics” SysML model elements. Figure 105 shows the “Optimize Return on Investments” activity.

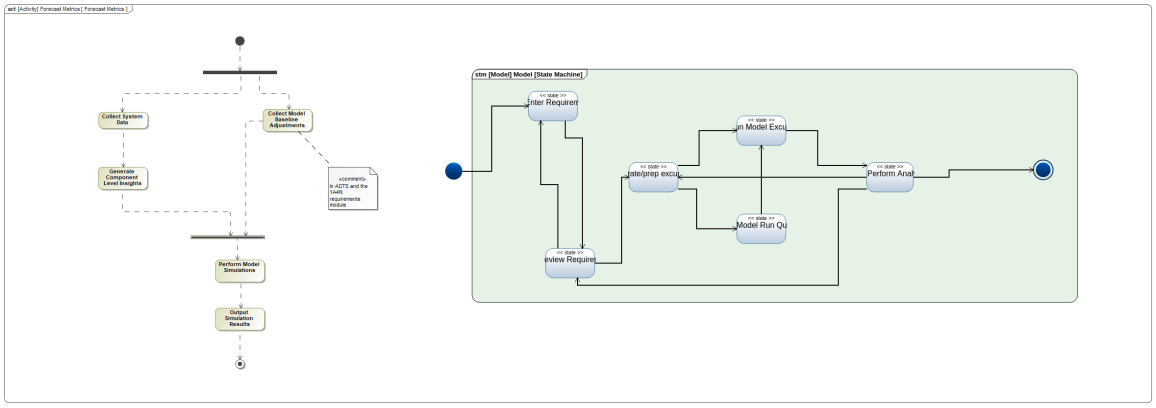


Figure 103. Forecast Metrics functional analysis, modeled in an activity diagram. Zoom in to see finer details.

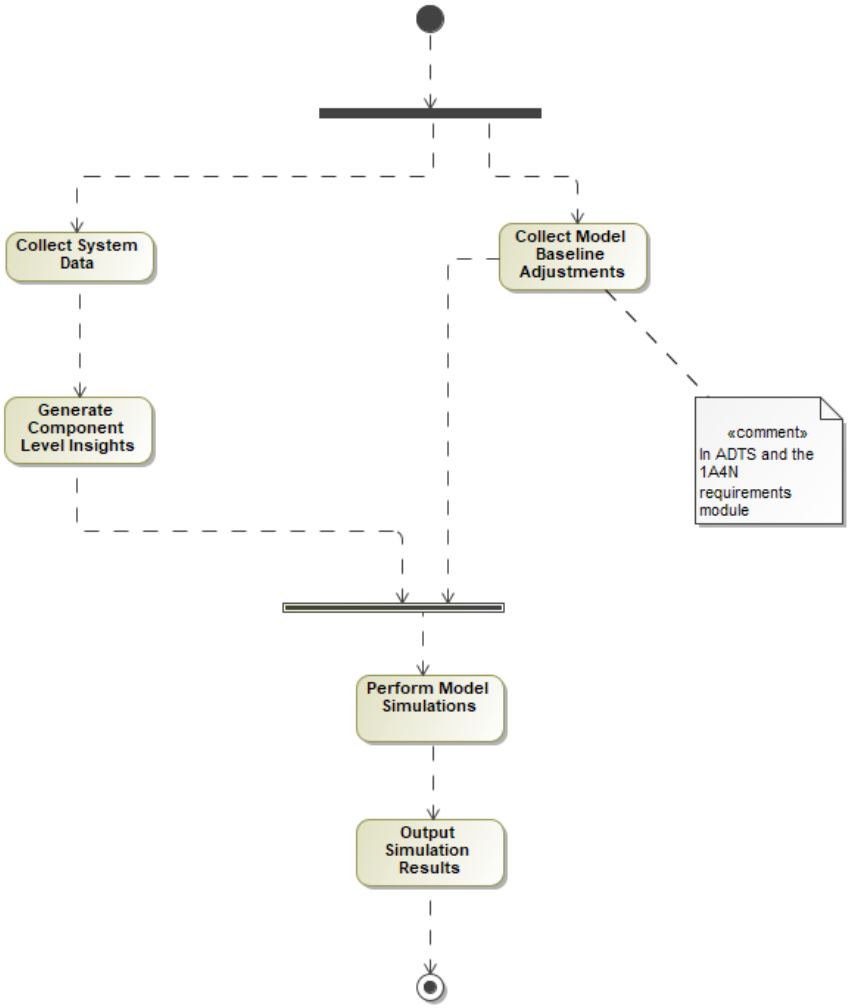


Figure 104. "Forecast Metrics" functional analysis, modeled in an activity diagram. This graphic shows only the SysML model elements.

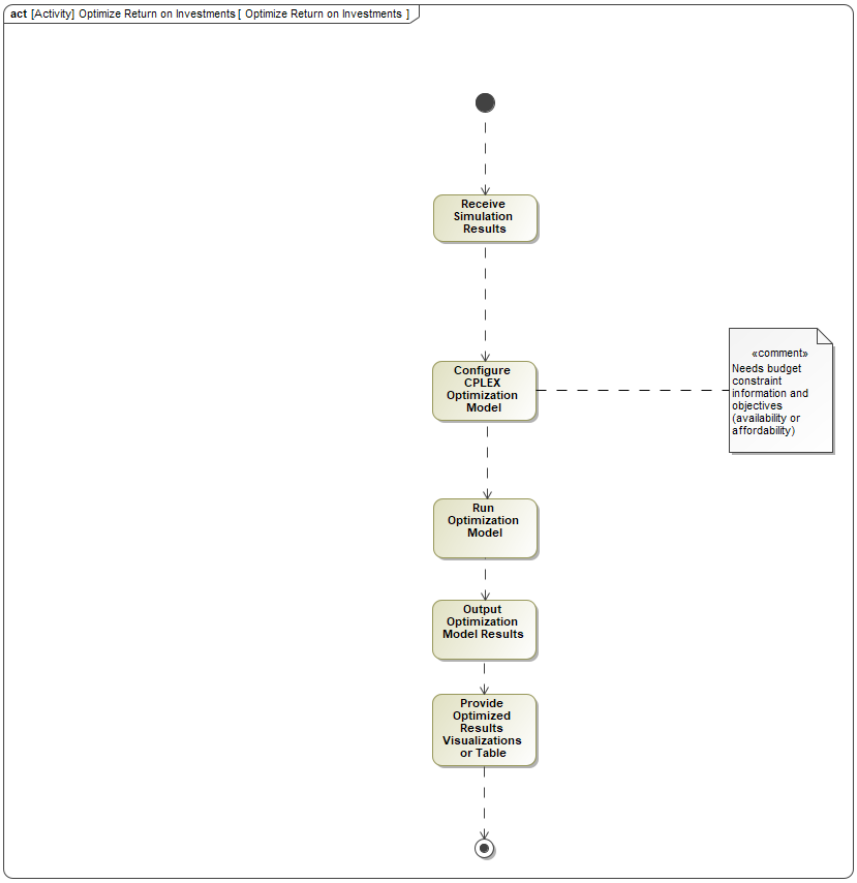


Figure 105. Optimize Return on Investment

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Allocate the solution-neutral functions to their respective logical subsystem elements identified in the (W3) "Logical Subsystem Communication" grid.
2. Review the functional analysis with M&S SMEs and other stakeholders.

8. Modeling CRIM's Measurements of Effectiveness

a. Modeling Objectives and Guidance

Measures of Effectiveness (MoEs) are defined in the model for this step. Stakeholder needs and system use cases can be directly refined and restated as MoEs, or they can inform and guide the derivation of MoEs. MoEs should capture *what* the system must do or *what* outcomes or effects the system must have on its environment, accompanying systems, or users in order for it to effectively satisfy its operational concepts. MoEs are associated with the system's emergent behavior or system quality attributes.

The MagicGrid v.1 QuickStart template provides a guide for how to model MoEs, shown in Figure 106. The guide states that "non-functional stakeholder needs are refined with measures of effectiveness," and that "you [can] express system characteristics in numerical format." Instead of following the QuickStart MoE guide, the author opted to capture MoEs in terms of functional things that CRIM must do or in terms of things that CRIM must provide for it to be effective. Along with MoEs, the author modeled Measures of Performance (MoPs). MoPs provide measures or attributes that are quantitative in nature. MoPs should address how well CRIM must perform its effectiveness measures or MoPs should define the quality attributes or characteristics of the things that CRIM has or provides.

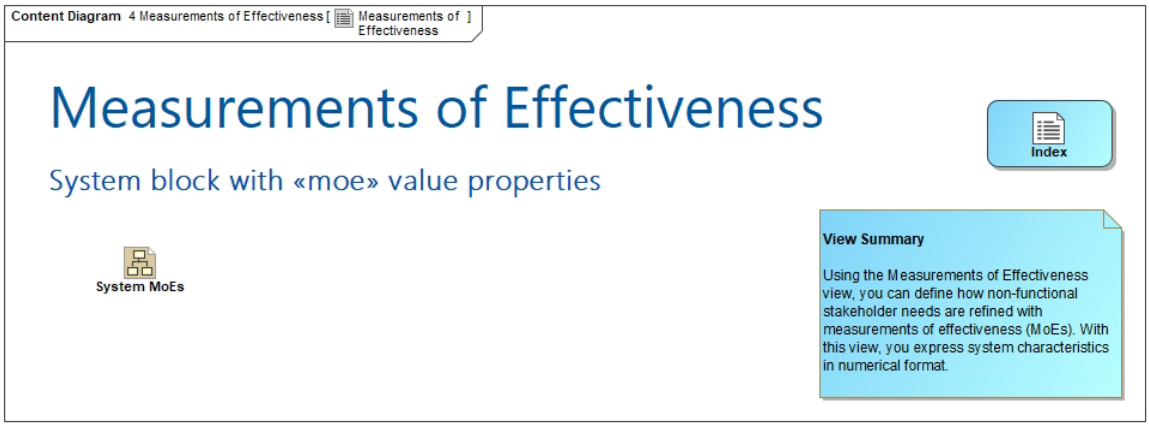


Figure 106. MagicGrid v.1 QuickStart template guide for Measures of Effectiveness.

b. Information/Data Collected

Table 11. Measurements of Effectiveness Information or Data Collected

Information Gathered	Artifacts/Information mode	Source
System Measures of Effectiveness information	Verbal- CRIM discussions and whiteboarding sessions	Jason Thomas
	Verbal- CRIM general discussions	CRIM stakeholders

c. Modeling Steps and Results

The author generated a block definition diagram to capture information related to CRIM's MoEs and MoPs. The diagram serves as the namespace that can contain modeling elements and other information related to MoEs and MoPs. Two blocks are used to represent the existence of MoEs and MoPs. Comments are used to capture MoE and MoP information. Alternatively, the MoE and MoP block can be decomposed into subparts if desired. This way information can be partitioned and allocated better or in a more organized manner. The modeling results are depicted in Figure 107.

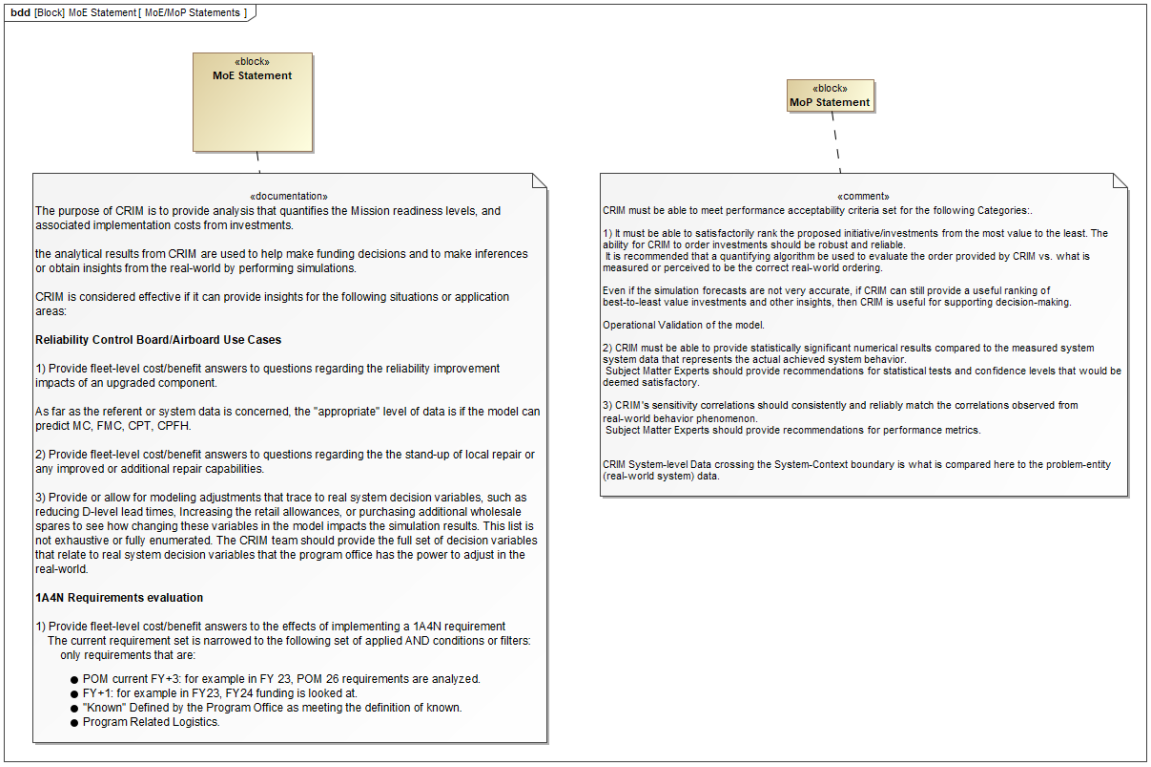


Figure 107. CRIM system level MoE and MoP statements

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Review MoEs and MoPs with stakeholders.
2. Refine MoEs and provide more concrete MoP quantitative measures.
3. Optionally, format MoEs and MoPs in a bdd hierarchy structure to better show the taxonomy of MoEs and MoPs.

9. Modeling CRIM's Logical Subsystems Communication

a. Modeling Objectives and Guidance

The goal of this modeling task is to capture CRIM's logical subsystem. The logical subsystem begins to define the form that the implemented solution concept will take without concretely defining the concept. The system's structure is defined by the types of

components that compose CRIM and their logical relationships in terms of composition and structure. It is during this step the M&S conceptual model can be captured.

The MagicGrid v.1 QuickStart template provides modeling guidance on what details should be modeled here; shown in Figure 108. The QuickStart template provides a good insight, stating that the logical elements and structure modeled here should be "based on the control and resource flows defined in the Functional Analysis view," which should be defined in a solution neutral manner.

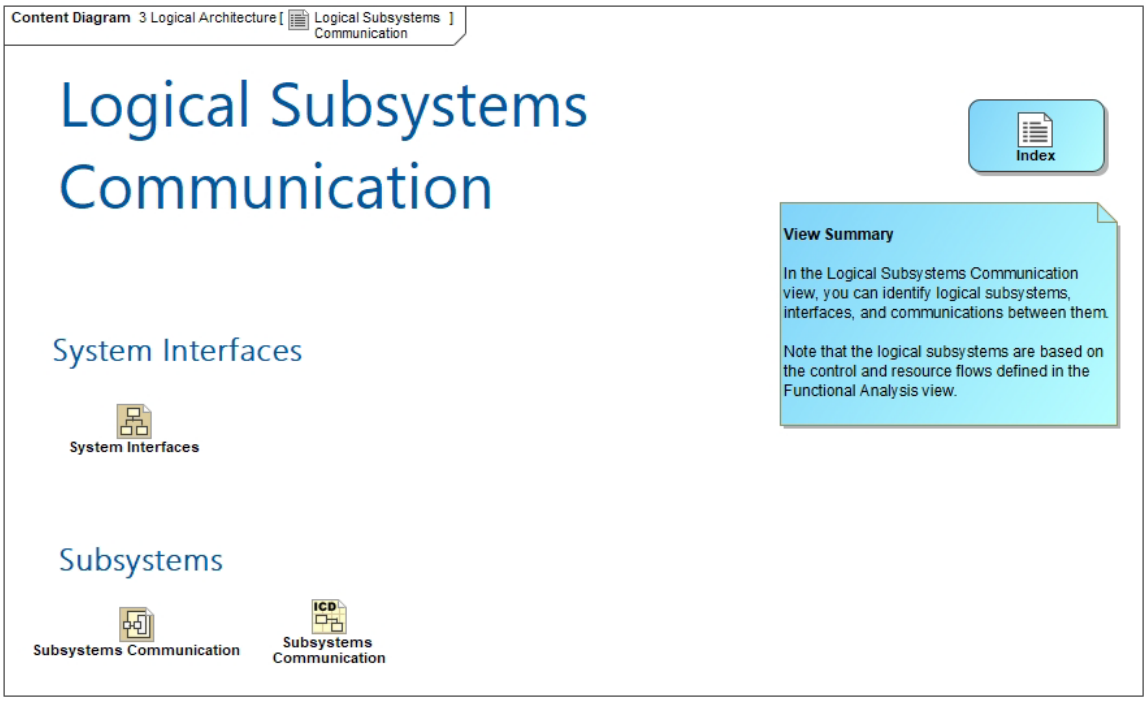


Figure 108. MagicGrid v.1 QuickStart template guide for modeling the Logical Subsystems Communication

b. Information/Data Collected

Table 12. Logical Subsystems Communication Information or Data Collected

Information Gathered	Artifacts/Information Mode	Source

Logical System Structure Information	Mental model: inferred using inductive reasoning; taking concrete CRIM system information from various documents and abstracting the details to depict CRIM's logical components.	The author

c. Modeling Steps and Results

The author has knowledge and awareness of CRIMs implemented state. Therefore, the author had to abstract and classify CRIM's concrete elements into their logical elements according to their essential functional identities.

The logical subsystems communication architecture is a good foundation for capturing the M&S conceptual model. The author has limited knowledge of all the conceptual model parts and assumptions. At the very least, the type of conceptual models are captured here: the "Reliability Tree" simulation model, and the "Multi-echelon, multi-indenture" discrete event Monte Carlo simulation model. With more complete information and clarifications these models can be further decomposed and refined. Modeling results for this task are shown in Figure 109.

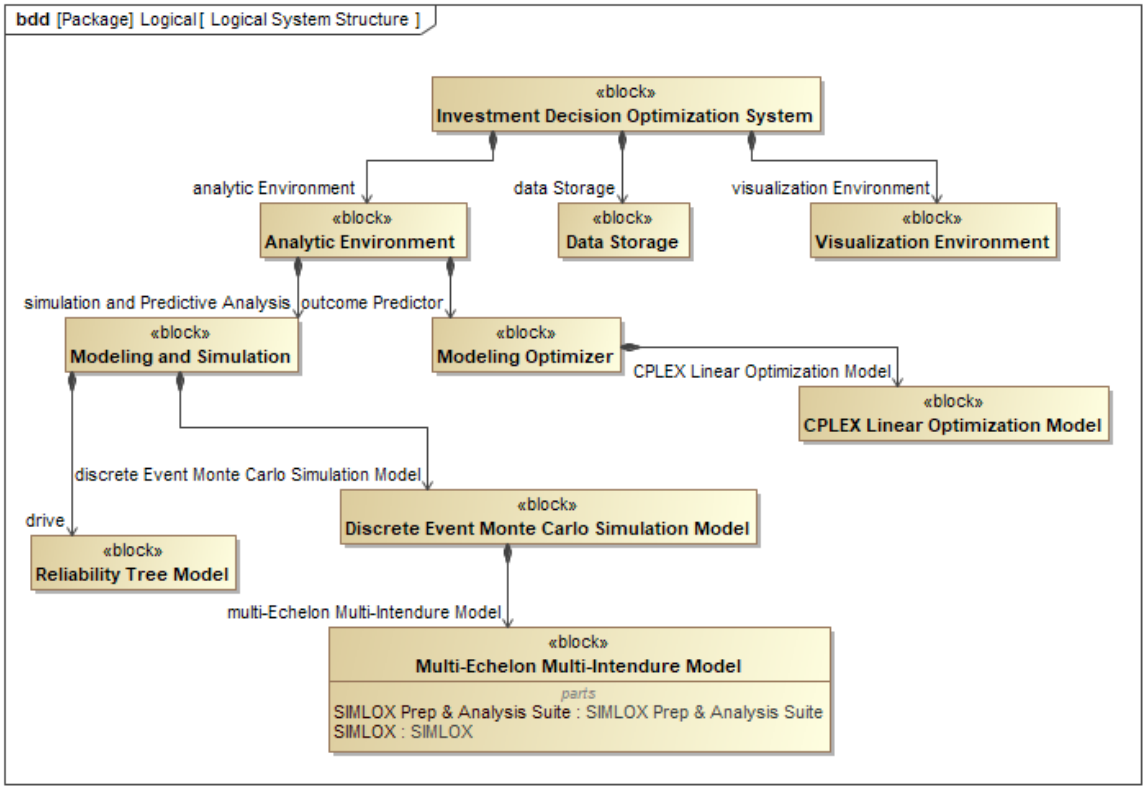


Figure 109. CRIM logical system structure

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Decompose and refine the M&S conceptual models. The conceptual model should identify the working principles and assumptions of the simulation model, but at a more abstract logical level.
2. Review logical subsystem communication architecture with stakeholders.
3. Allocate functions from the functional analysis to the logical structural components.

10. Modeling (W4) Measures of effectiveness

a. Modeling Objectives and Guidance

In the previous step, the logical subsystem structure was defined. As a result, the M&S conceptual models have also been identified. Now that CRIM's conceptual models have been distinguished from each other, validation methods can be allocated or referenced as placeholders to facilitate future V&V discussions.

The modeling objective is to identify the conceptual model validation methods that may be suitable for the defined M&S conceptual models.

b. Information/Data Collected

No artifact was used to support this modeling task. The author used concepts from general systems theory and M&S VV&A concepts presented in Chapter II of this thesis document.

c. Modeling Steps and Results

In this simple step, the two types of CRIM conceptual models are modeled as blocks to serve as placeholders so that additional validation methods can be associated to them. Additionally, these blocks serve as the basis for further decomposition. A common validation method for M&S conceptual models is facial validation. In this case, the author proposes that this MBSE descriptive model be used by M&S subject matter experts (SMEs) to communicate and do a walkthrough of the conceptual model assumptions and working principles to other stakeholders, particularly SMEs that are familiar with the real system that the M&S model is based on. Results of this modeling step are shown in Figure 110.

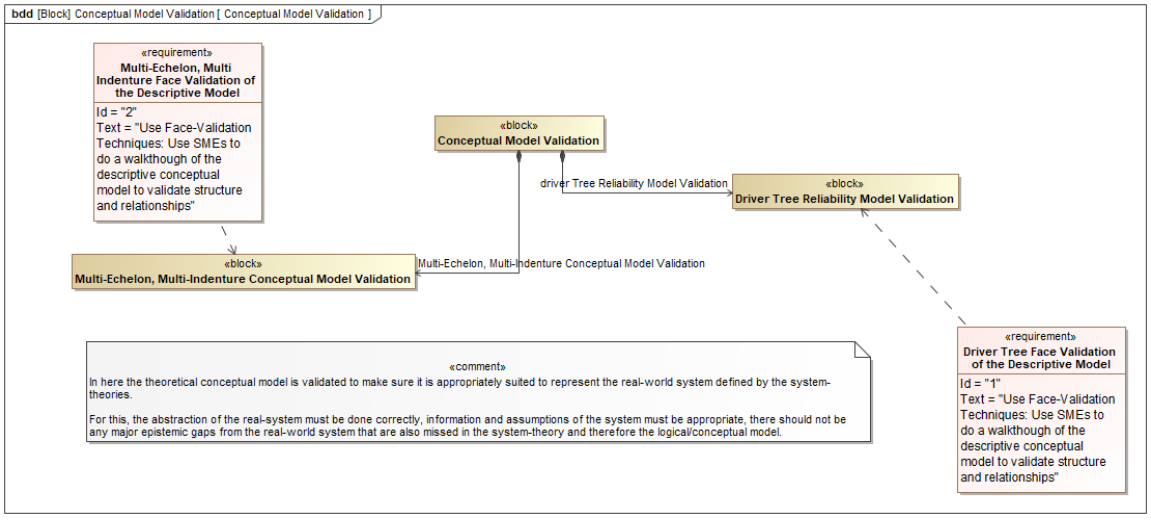


Figure 110. Conceptual model Validation (W4 Measures of Effectiveness).

d. Future Work

Below is the modeling improvement that could be adopted in the future:

1. The conceptual model validation blocks can be further decomposed and refined. This would allow isolating specific conceptual model validation parts.

11. Modeling System Structure

a. Modeling Objectives and Guidance

The objective for this modeling task is to decompose the system's structural architecture into its subsystems or primary parts. The MagicGrid v.1 QuickStart template provides guidance on what diagrams can be used to model the system structure, shown in Figure 111.

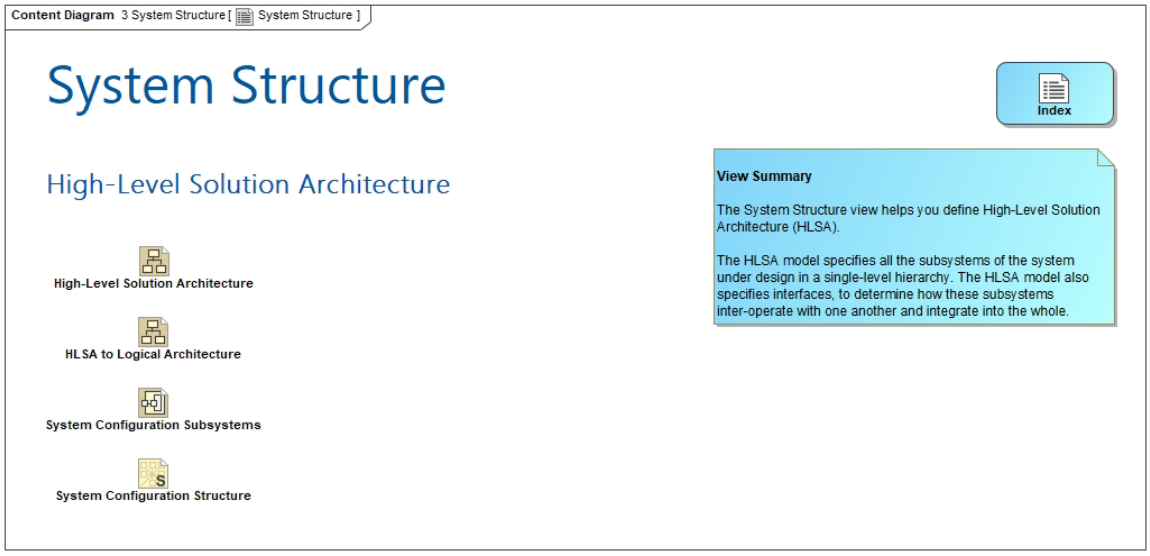


Figure 111. MagicGrid v.1 QuickStart template guidance for the System Structure

b. Information/Data Collected

Table 13. System Structure Information or Data Collected

Information Gathered	Artifacts/Information mode	Source
System Structure Information	Innoslate bdd diagram	Jason Thomas
	Innoslate asset diagram	Jason Thomas
	Cartoon Architecture graphics of CRIM data pipelines	IBM

c. Modeling Steps and Results

The author developed a block definition diagram to capture CRIM's structural composition hierarchy, and then the author with ManTech's help created an internal block diagram that provides structural relationships between CRIM's internal components. The author pasted artifacts that served as useful reference material. The results for this modeling task are shown in Figures 112 and 113.

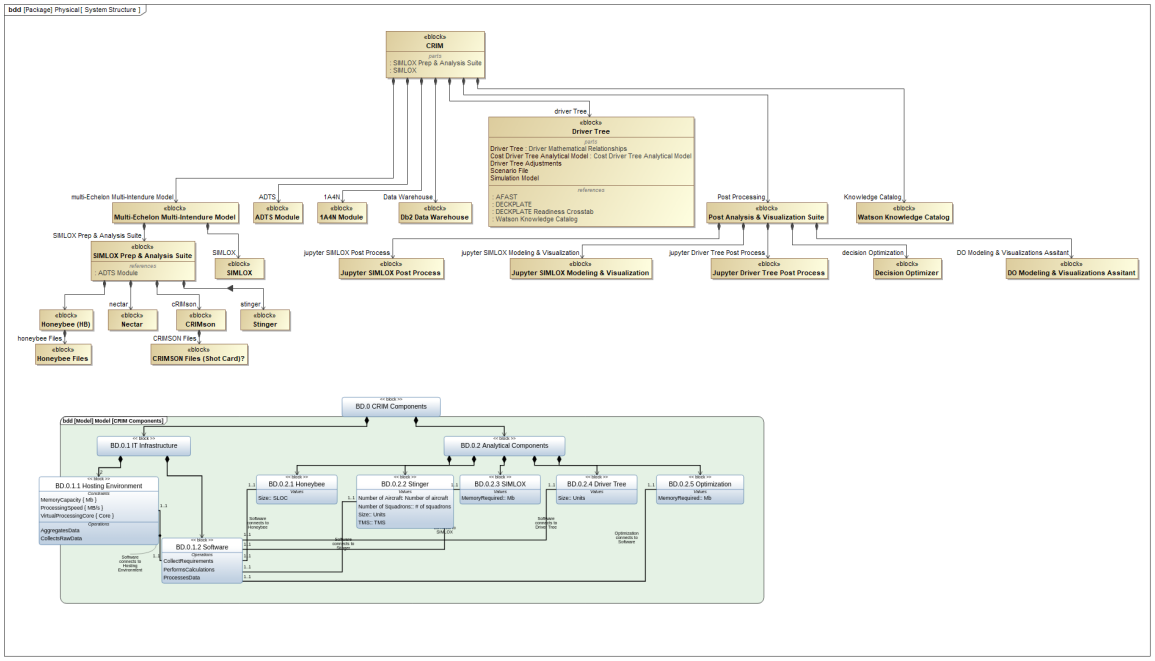


Figure 112. CRIM System Structure, block definition diagram with a pasted artifact. Zoom in to see small details.

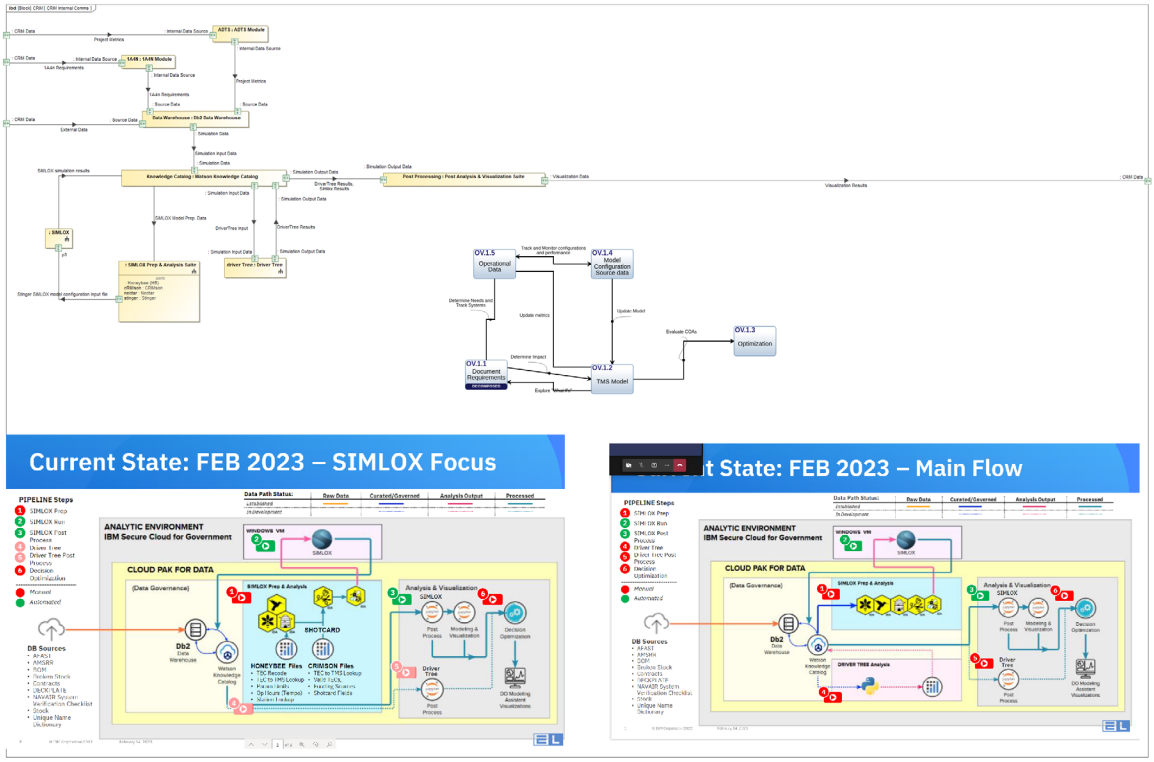


Figure 113. CRIM's internal block diagram with pasted artifacts. Zoom in to see small details. There is an enlarged version of this image in the appendix.

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Decompose all subsystems of interest. So far, only the "Driver Tree" model subsystem, the SIMLOX Prep & Analysis Suite" and the "SIMLOX" subsystems have been decomposed into internal parts. These are the primary modeling entities with the greatest impact to the modeling working principles and operations. If data validity is acceptable, then these elements are responsible for providing valid or useful results.

2. Decompose all data flows of interest into their discrete data elements. Knowing the discrete data elements and their values is crucial for supporting V&V analysis. Analysis on system partitions cannot be

performed fully unless the inputs and outputs are known, thus supporting quantifiable testing methods.

12. Modeling CRIM's System Behavior

a. Modeling Objectives and Guidance

The solution-neutral functional architecture provides the basis from which to derive the more concrete system functional architecture that defines *how* the system is going to achieve or perform the solution neutral functions. The objective of this step is to capture the system level functional architecture. This level should broadly capture all primary functions so that these may be decomposed in later modeling. The system level functional architecture is an implementation concept. Along with the solution-neutral functional architecture, the stakeholder needs, and a general system functional vision can additionally inform what an effective implementation concept can be—creativity and design freedom could provide suitable conceptual variations of the system functional architecture. The MagicGrid v.1 QuickStart does not provide a template for this modeling task.

b. Information/Data Collected

Table 14. System Behavior Information or Data Collected

Information Gathered	Artifacts	Source
System level behavior information.	Innoslate high-level system architecture diagrams	Jason Thomas
	Innoslate high-level sequence diagram.	Jason Thomas
	Mental model-Inductive reasoning from more concrete details to a generalization of what the system does.	The author

c. Modeling Steps and Results

For this step, the current mode of operations or system functionality of CRIM is **not** modeled. Instead, the future state/vision of how CRIM will function is what is modeled.

The current state of CRIM requires a lot of manual processing and operations by the CRIM modeling team made up of data scientists and analysts. The future vision of CRIM is to integrate and automate as many data flows and processes as possible. Data scientists and analysts are used for more "bespoke" or complex modeling projects. It is more valuable to model the future state because this will help define the system-level design specifications that are needed to design and integrate CRIM components so that the future state works as envisioned here.

The author used the listed references for specific insights as to what type of information needs to be translated into the SysML model elements. The author also used his general knowledge of CRIM's current processes to infer what an automated version of CRIM may look like. The core or essential operations should not change because technology is used rather than manual labor, but the order or structure of these may vary. There is likely more degree of freedom now with manual processing, but to support integration and automation, a standard functional structure may be required. The sequence and structure of functions is determined largely by their logical dependencies so that limits the variation of how the overall system functional architecture will be defined.

The following figures, Figure 114, and Figure 115, capture CRIM's system-level design concepts. This design was done independently by the author—not ideal. This level of design should incorporate the views of a broader set of stakeholders, especially the users and lead system integrators, in this case IBM.

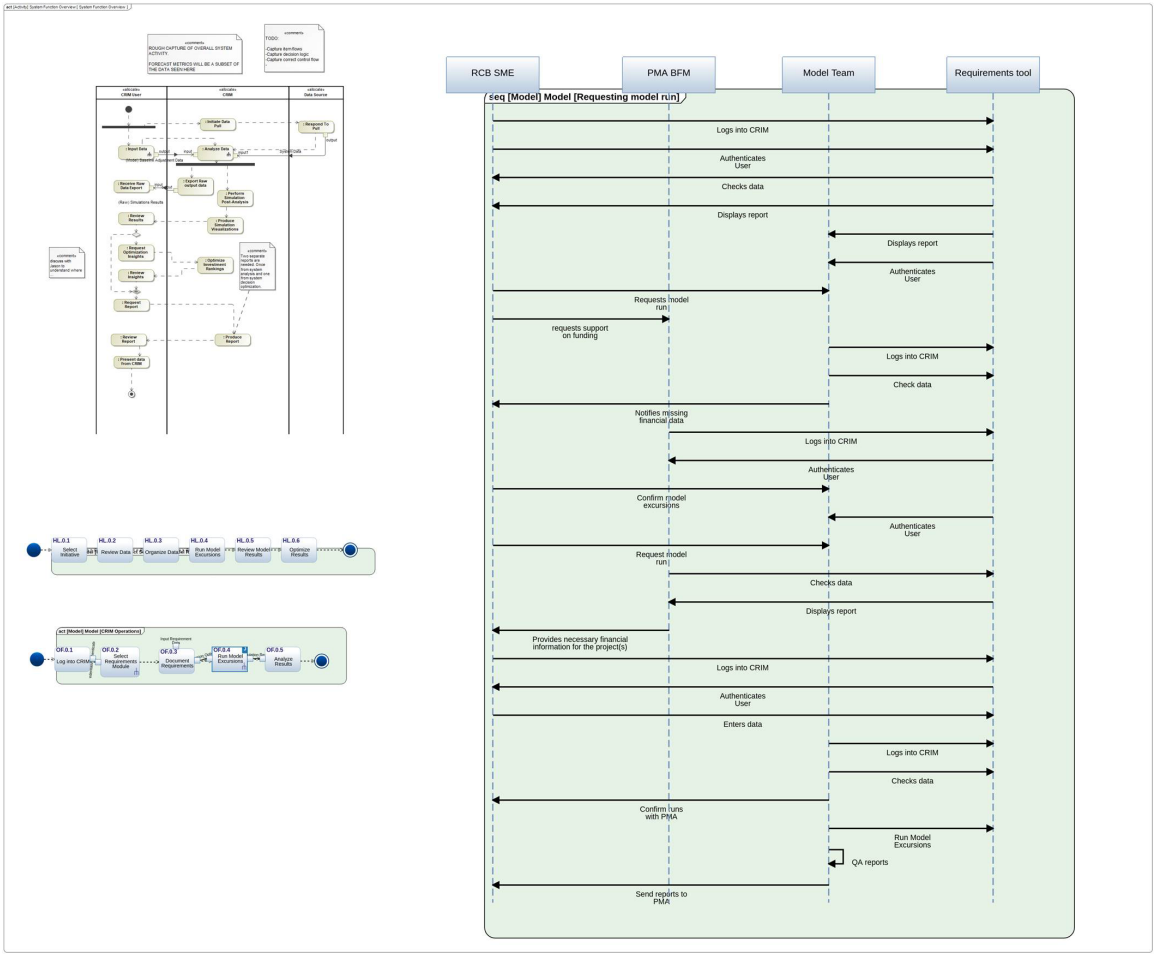


Figure 114. CRIM (proposed) System Behavior with reference artifacts. Zoom in for finer details.

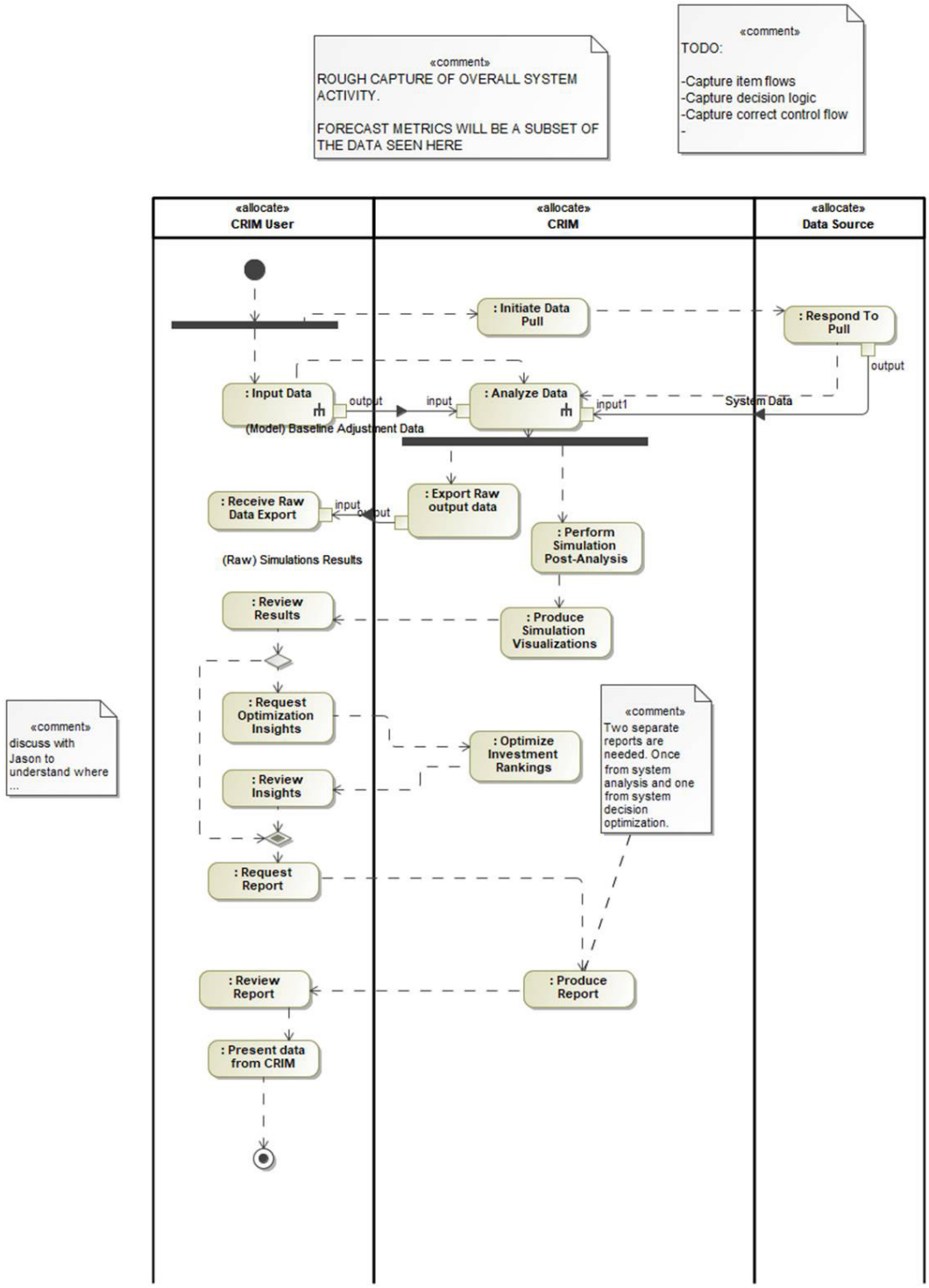


Figure 115. CRIM System behavior, SysML model elements only.

d. Future Work

Below is the modeling recommendation that should be performed in the future:

1. Review and refine CRIM's system level functional architecture with stakeholders and the CRIM design team.

13. Modeling CRIM's System Requirements

a. Modeling Objectives and Guidance

After defining the system level functional architecture and understanding its functions, inputs and outputs, it is easier to then define more technical system requirements. Other requirements may be non-functional or represent attributes or characteristics that the system must possess.

At the system level, the author considers primary technical requirements to be the things of utility or value that system delivers at the system boundary, or the type of external effects or impacts the system must have. A change of the system state may also be an emergent thing of value to the stakeholders. Requirements can be written to capture the necessary system inputs and the outputs of value that the system delivers at the system boundary. Therefore, special emphasis must be placed on the interactions that happen at the system boundary and the internal system "value-pathway" that processes system inputs into the desired system outputs. To ensure that the system performs the necessary transformation of inputs to outputs, the core or essential system functions may be defined as requirements—these should capture all of the value-pathway functions. There may be various value-pathways depending on how system inputs are transformed into system outputs, and value-pathways may converge or diverge depending on the system functional architecture structure and how it is allocated to structural components.

The MagicGrid v.1 QuickStart template offers a guide to model system requirements, which includes traceability tables; shown in Figure 116.

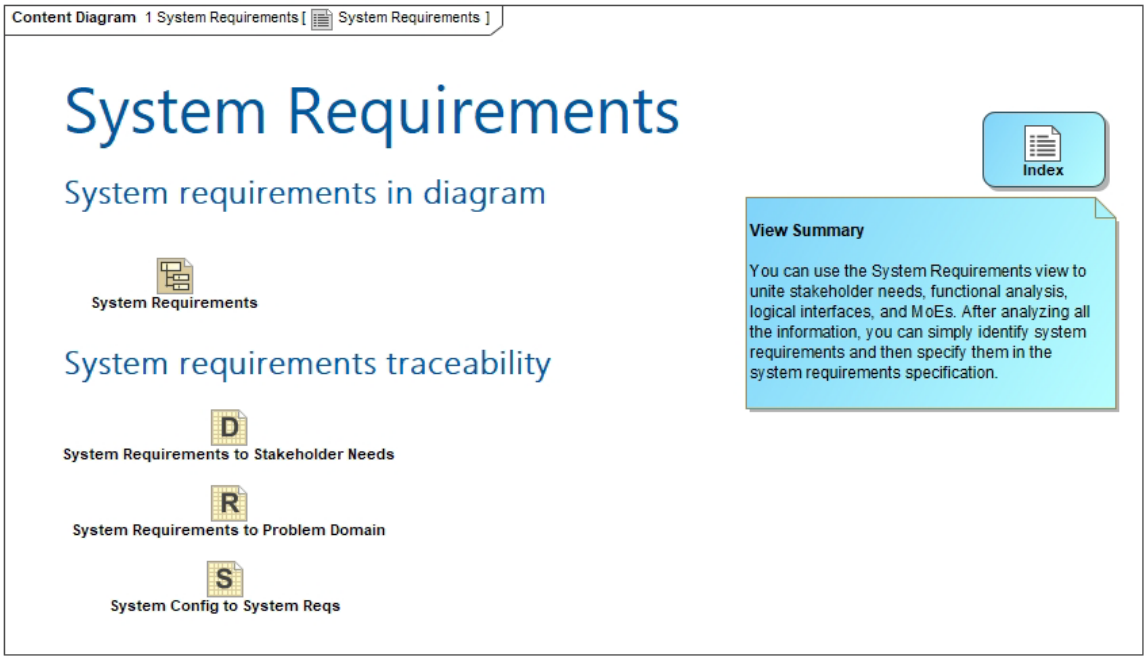


Figure 116. MagicGrid v.1 QuickStart template guide for System Requirements

b. Information/Data Collected

Table 15. System Requirements Information or Data Collected

Information Gathered	Artifacts/Information Mode	Source
System Requirements Information	Innoslate diagram	Jason Thomas
	Mental model of general CRIM knowledge	The author

c. Modeling Steps and Results

The author used his general knowledge of the CRIM system, the necessary inputs and outputs and the core essential functions to logically infer what the CRIM system requirements may be. Ideally, this type of modeling would be done with more stakeholders. The modeling results are presented in Figure 117.

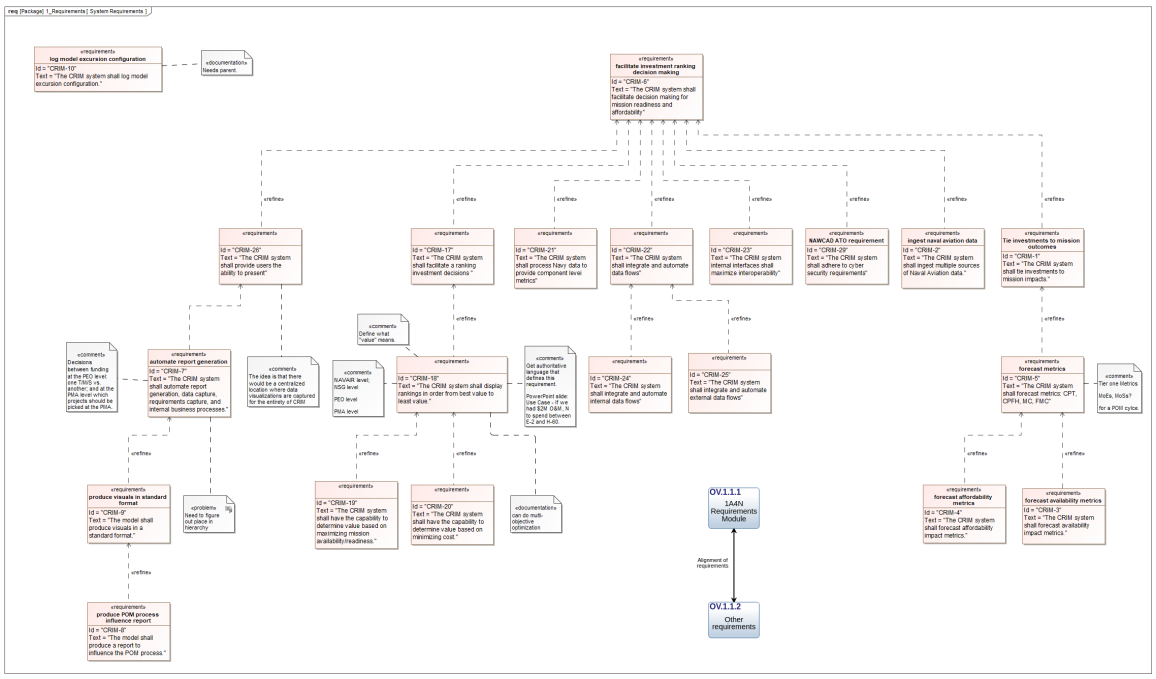


Figure 117. CRIM System Requirements with a reference Innoslate artifact. Zoom in to see finer details.

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Refactor requirement relationships. The “refine” relationship was abused here in some instances. The “refine” relationship has to do with taking an abstract definition of a requirement and refining it so that it is semantically more concrete or less ambiguous. The “containment” relationship is better suited to capture sub requirements that are part of a larger complex requirement made up of many requirements. The “derive requirement” is better suited to convey that a requirement is derived from another one through some logical inference or causality.

2. Re-structure. This diagram captures requirements that go below the system level. At the time of modeling, the goal was to maximize the amount of requirement information modeled. The requirements captured here are still

useful, but the MagicGrid framework provides grids/spaces for subsystem requirements and component requirements.

3. Review and refine requirements with CRIM stakeholders and the CRIM design team.

14. Modeling CRIM's System Parameters

a. Modeling Objectives and Guidance

The goal of this modeling task is to capture the system's performance parameters. These may be a refinement of the MoEs and MoPs defined. The refinement is done because at this point a more concrete system concept has been defined. The MoEs and MoPs are still important because these are defined in terms of stakeholder needs regardless or less dependent on how they are satisfied, i.e., more solution or technology neutral. The System parameters modeled here are associated with the implemented system's performance and quality attributes.

The MagicGrid v.1 QuickStart template provides a guide for modeling system parameters, shown in Figure 118. The author deviates somewhat from this guide which states that system parameters help quantify characteristics of "non-functional" system requirements. The author includes parameters or measures that are associated to system functions. Such as model accuracy or statistical significance. An example for an automobile's system parameters or performance measures may be speed, acceleration, weight, seating capacity, maneuverability (measures defined by some standard set of road tests), crash safety (measures defined by some standard set of tests), cabin road noise, range, fuel efficiency, etc. Some of these measures are non-functional and associated to quality attributes, but some of these measures are associated with how well the system performs functionally.

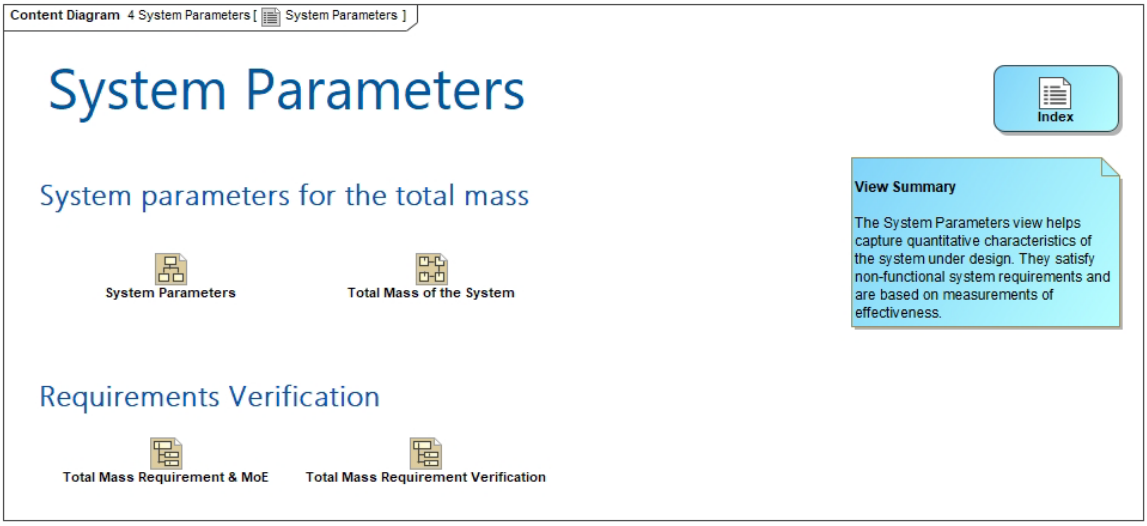


Figure 118. MagicGrid v.1 QuickStart template guide for modeling System Parameters

b. Information/Data Collected

Table 16. System Parameters Information or Data Collected

Information Gathered	Artifacts/Information Mode	Source
System parameter information	Deductive Reasoning: General understanding of CRIM's functionality translated into specific system parameter details.	The author

c. Modeling Steps and Results

The author inferred what the system parameters may be based on general M&S VV&A theory and from his general understanding of CRIM. In this step the author developed a taxonomy of system parameters that may be useful. The higher taxonomy levels provide the classification structure, but the lowest indenture (children) elements should be associated to some system element; functional or qualitative. For example, a concrete measure for system accuracy, in the case of CRIM, may be the measure of the statistical significance of the simulation results compared to system data, measured by a t-

test or some other type of statistical test. These parameters are classified at the system (emergent) level. System level parameters may be dependent upon a combination of sub parameter measures or performance. For example, a car's top speed may be dependent on the engine's power, drivetrain, car weight and aerodynamics. CRIM's system parameters modeled here should cover parameters for both the Driver Tree model and the SIMLOX model. Figure 119 shows the results of this modeling step.

Alternatively, a more refined conceptual model or ontology of system parameters and their relations to system elements may be formed. This taxonomy primarily only shows composition. Generalization relationships can be used to show inheritance or subsumption hierarchy properties. Inheritance or specialization happens when a more abstract entity or class is refined by a more concrete entity or class. The concrete "child" identity possesses the essential identity or qualities of the "parent," and has some specialized properties of its own, not conversely possessed by the more abstract parent. A subsumed entity inherits the essential qualities of the higher classes. A child class is subsumed by its parent class. A conceptual model or ontology can show relationships of entities across taxonomies or other concepts.

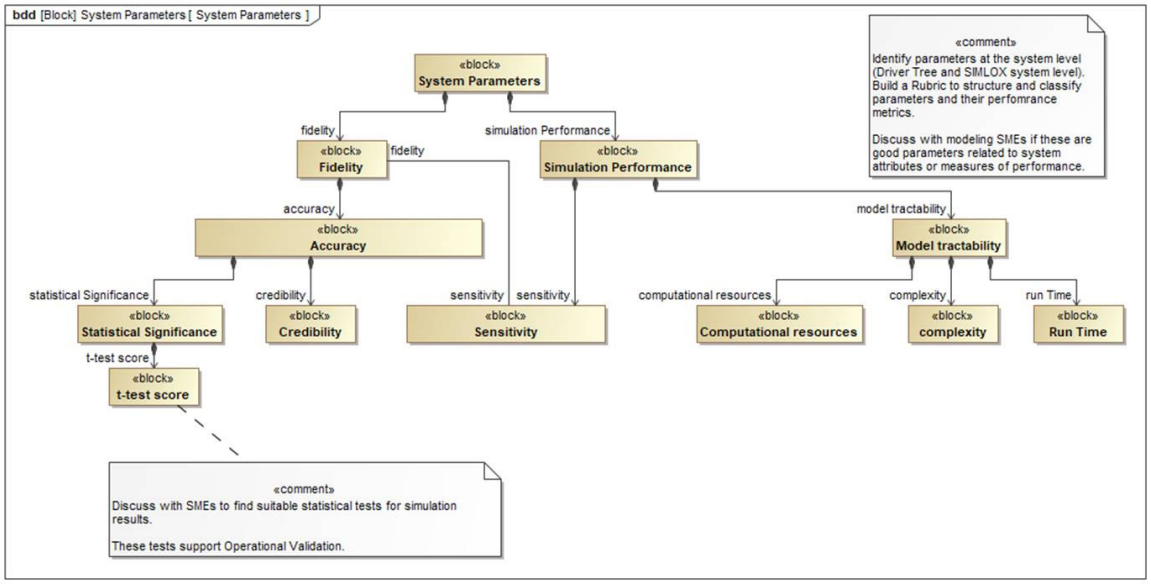


Figure 119. CRIM System Parameters

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Review and Refine parameter measures with the CRIM development team. Parameter measures should be associated to some V&V methodology.
2. Trace parameters to V&V methods. A fuller set of V&V methods needs to be modeled to support this traceability in the model. V&V methods should be assigned or recommended by SMEs and should be allocated to the corresponding system component or partition.

15. Modeling Subsystem Behavior

a. Modeling Objectives and Guidance

The objective of this state is to decompose the system level functions, down to the level of decomposition that can be appropriately allocated to the subsystems defined in the subsystem structure. Behavior at this level is not emergent behavior of the system, but rather at the subsystem level. Performing a functional analysis of the system level functions supports decomposition and definition of activities at the subsystem level.

The MagicGrid v.1 QuickStart template provides guidance on how the subsystem behavior can be modeled, shown in Figure 120.



Figure 120. MagicGrid v.1 QuickStart template guide for modeling Subsystem Behavior

b. Information/Data Collected

Table 17. Subsystem Behavior Information or Data Collected

Information Gathered	Artifacts/Information Mode	Source
Data entry information to support subsystem behavior modeling	Innoslate activity diagram for data input modules	Jason Thomas
Data analysis information to support subsystem behavior modeling.	Innoslate activity and action diagrams for model excursion operations	Jason Thomas
General subsystem behavior information	Mental map inferred from CRIM functional principles.	The author

c. Modeling Steps and Results

In this step the author decomposed the "Input Data" and "Analyze Data" system level functions captured in the "System Behavior" grid. These actions are in the end-to-end modeling value-pathway, meaning that the sequence structure of these processes supports the end-to-end, input to output value that CRIM delivers to the users. Along this value pathway, the simulation models perform the core transformation of inputs into outputs.

Two activity diagrams were generated for the two system level actions "Input Data" and "Analyze Data." The "Input Data" activity diagram shows the functions performed in this behavior and their allocation to system context elements, i.e., the "CRIM User" (external) and the "1A4N Module" and "ADTS Module." A supporting artifact was pasted into the canvas for reference. This artifact is an Innoslate model diagram that roughly captures the user's data entry processes. The "Input Data" activity diagram is shown in Figure 121.

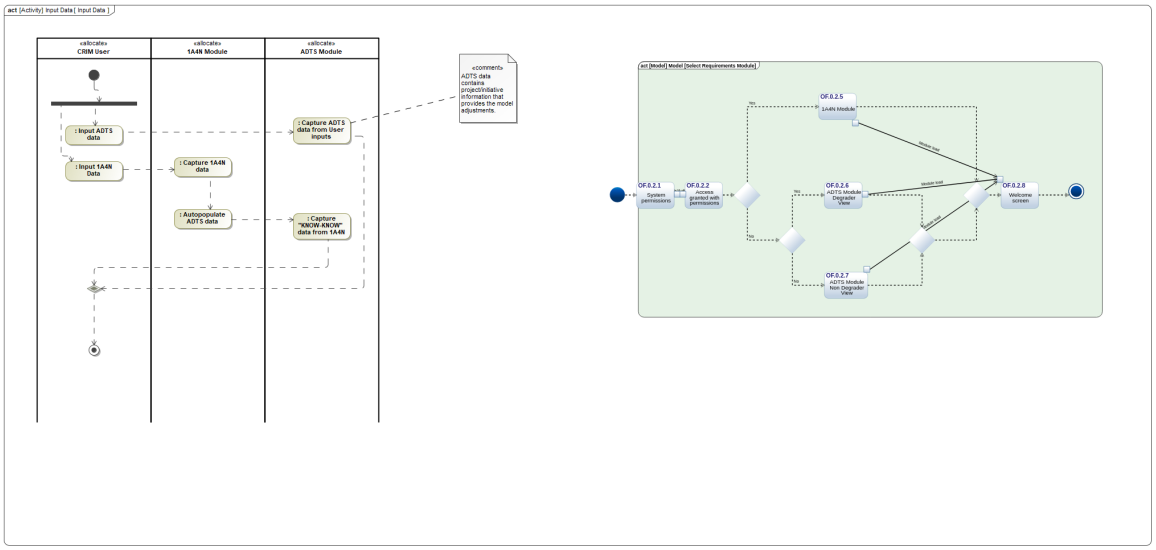


Figure 121. CRIM Subsystem Behavior, "Input Data" activity diagram. Zoom in to see small details.

The "Analyze Data" activity diagram shows the main functions that can be performed to analyze data. This diagram conveys that the user can either use the Driver Tree simulation model or the (SIMLOX) Multi-Echelon, Multi-Indenture simulation model to perform the analysis. Innoslate diagram artifacts were pasted into the diagram canvas for reference. Figures 122 and 123 show the results of this modeling step.

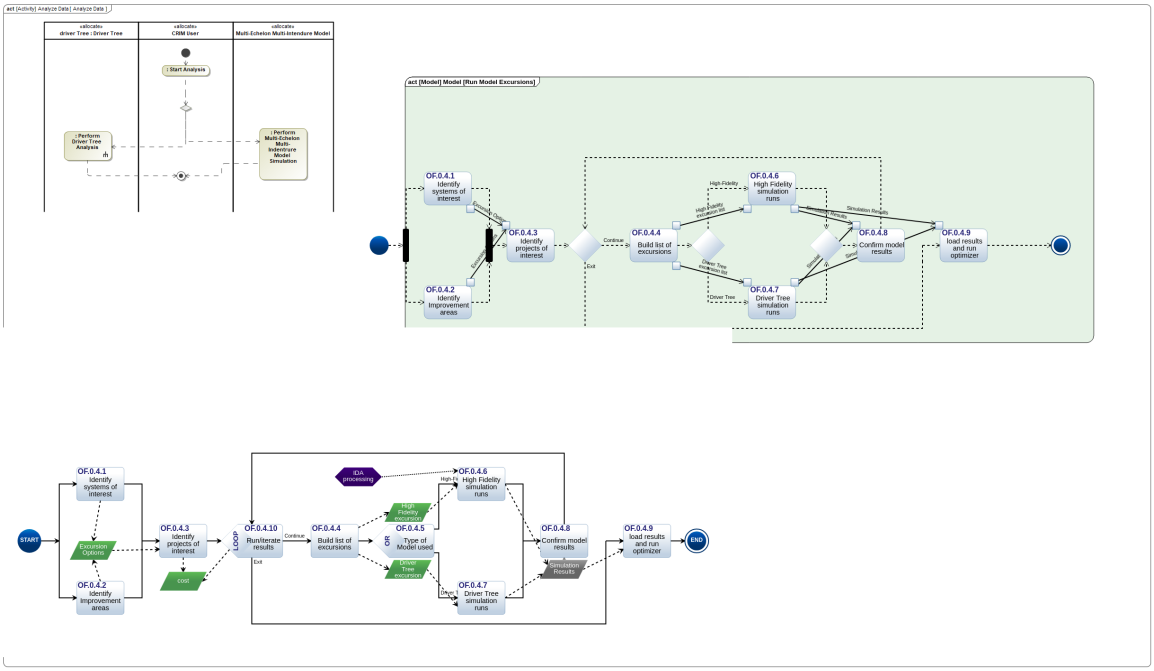


Figure 122. CRIM Subsystem Behavior, "Analyze Data" activity diagram.

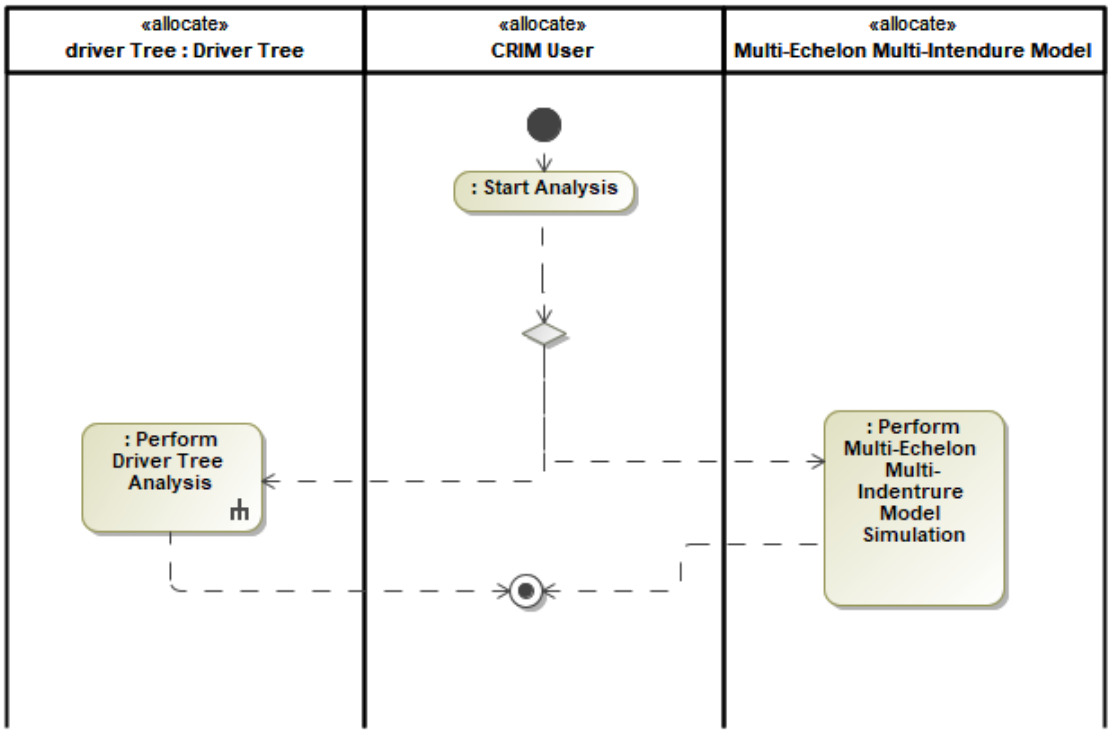


Figure 123. CRIM Subsystem Behavior, "Analyze Data" activity diagram showing only SysML model elements.

d. Future Work

Below is the modeling recommendation that should be performed in the future:

1. Review Subsystem Behavior diagrams with CRIM stakeholders. Overall, this level of detail is fine for partitioning the next level of decomposition.

16. Modeling Component Behavior

a. Modeling Objectives and Guidance

The objective of this step is to decompose the subsystem behavior to the level of functions that would be allocated to subsystem components. Decomposition is facilitated by performing a functional analysis of the subsystem level functions. The MagicGrid v.1 QuickStart template does not offer a modeling guide for this task.

b. Information/Data Collected

Table 18. Component Behavior Information or Data Collected

Information Gathered	Artifacts/Information Mode	Source
Component behavior information for the (SIMLOX) Multi-Echelon, Multi-Indenture model	Multi-Echelon, Multi-Indenture model graphics from presentations and other documents	Institute for Defense Analyses (IDA)
Component behavior for the Driver Tree model	Driver Tree graphical and textual based information from presentations and word documents.	Booz Allen Hamilton (BAH)

c. Modeling Steps and Results

In this step, the subsystem behavior for both the "Driver Tree" and "Multi-Echelon Multi-Indenture" subsystems was decomposed. Two activity diagrams were created, one for each of these subsystems. The quality of the modeling done for this task heavily depended on the information that was collected. This level of system detail is not commonly discussed during the CRIM project meetings. When these details were discussed, technical information details did not go much deeper than the information

gathered by the author. Therefore, the author was limited by the quality of the information available, what information was presented, which information was omitted or too ambiguous or abstract to properly support modeling (leading to descriptive epistemic uncertainty. The author collected CRIM project information, sorted the information useful for this task, and pasted graphical and textual artifacts into the diagram canvas to facilitate modeling actions and future communications with CRIM subject matter experts.

The results for the "Multi Echelon Multi Indenture" component behavior modeling are presented in Figure 124 and 125. The activity was named "SIMLOX Prep & Analysis and Simulation" to convey that the components perform preparatory steps—collecting and collating data and performing some analytical steps along the way—and then, the model simulation is performed at the end.

It is very desirable to further decompose the behavior of the "Honeybee" component. "Honeybee takes "cleaned" source data for various metrics, such as "demand history" for requisitioned aircraft parts or "NIINs" (National Item Identification Number) and creates "component-level metrics." In this case the "component-level metric" is the statistical distribution for "demand history," which can be used to support SIMLOX's stochastic discrete event simulation processes in the Multi-Echelon Multi-Indenture model.

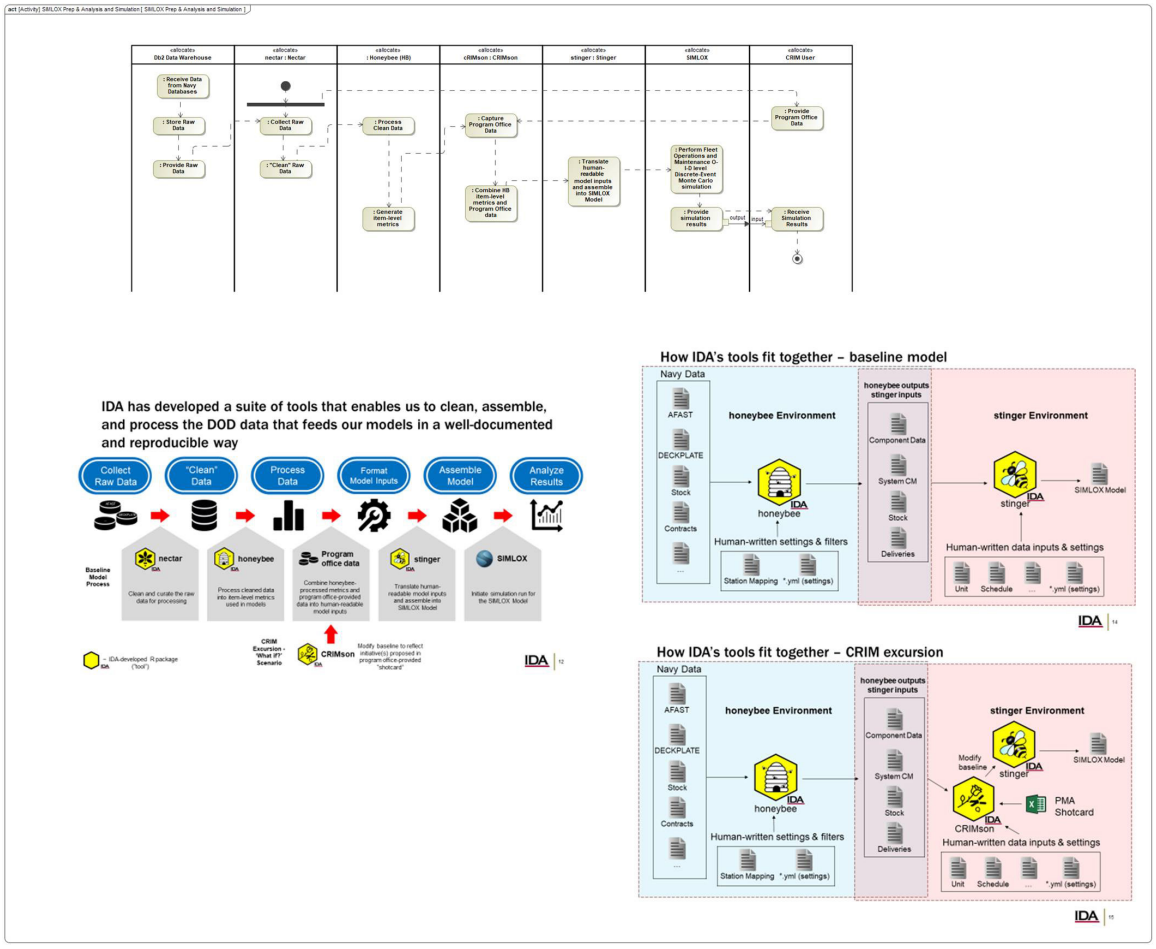


Figure 124. SIMLOX Prep & Analysis and Simulation activity diagram with pasted reference artifacts.

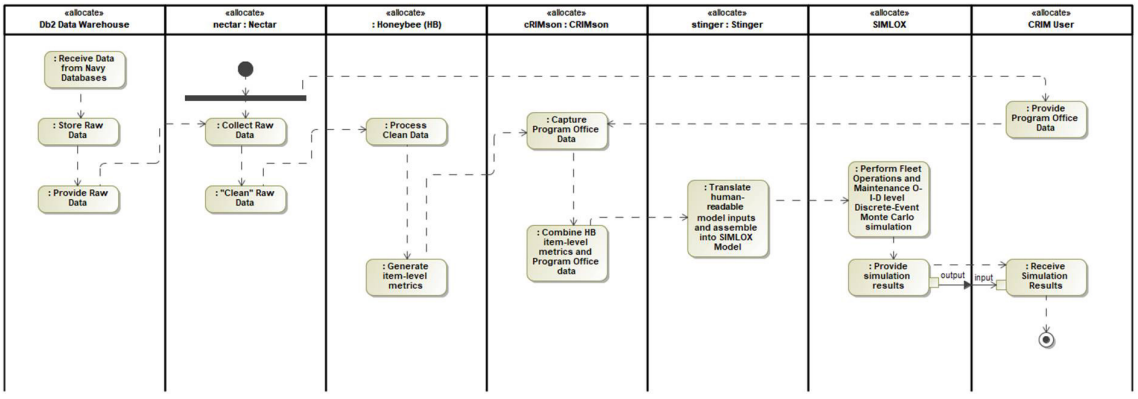


Figure 125. SIMLOX Prep & Analysis and Simulation activity diagram, SysML model elements only.

The Driver Tree component behavior was more difficult to model. Even though the author was able to collect a significant level of information, the information left the author with uncertainty about the Driver Tree details. Much of this uncertainty was due to ambiguity of the graphical and textual information. "Ambiguity exists linguistically due to our inability to express a meaningfully declarative sentence. Hence, we may have ambiguity with the truth condition of a proposition" (Modarres 2016, 2). In this case ambiguity was also caused by the graphical language semantics, i.e., Visio's "Basic Flowchart Shapes." It was difficult to know what was being conveyed graphically. Due to the ambiguity, the author pasted all the information useful for this diagram in the canvas to serve for future reference. The author also modeled a simple activity for Driver Tree's high-level functions, but the author was not able to allocate them. The modeling results are shown in Figures 126 and 127.

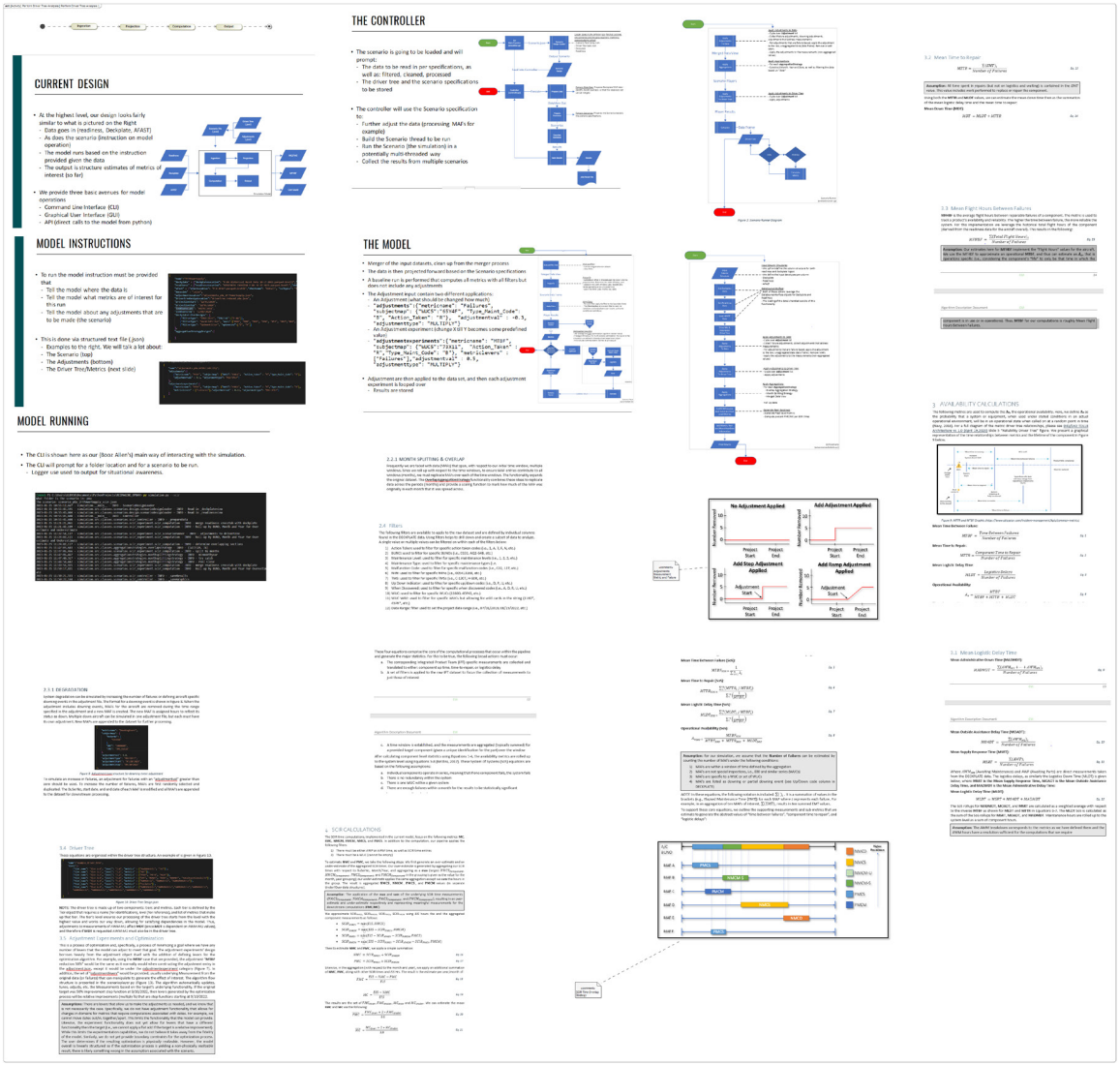


Figure 126. Perform Driver Tree Analysis activity diagram with pasted reference artifacts.

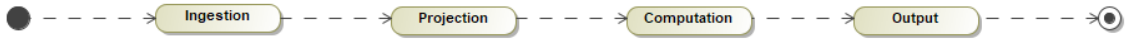


Figure 127. Perform Driver Tree Analysis activity diagram, SysML elements only.

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Review the current component behavior details with the CRIM design team and get additional details and clarifications.

2. Further decompose the behavior of components to the level of discrete details with discrete inputs and outputs.

NOTE: Alternatively, this level of detail could have been presented in the (SS2) "Subsystem Behavior," giving the viewer a white-box view of the Subsystem Behavior instead of a mostly black-box view, presented in the "Input Data" and "Analyze Data" activity diagrams in the (SS2) "Subsystem Behavior" grid. In other words, the behavior details for the (C2) "Component Behavior" grid are moved up, thus now leaving the (C2) "Component Behavior" grid space to model individual simulation mode components, like "Honeybee." In fact, this may have been the cleaner, more "technically correct" approach. The author may opt to make these modeling changes in the future.

17. Modeling Subsystem Structure

a. Modeling Objectives and Guidance

In this step the subsystem structure is modeled. Subsystems are decomposed into their main components. The MagicGrid v.1 QuickStart template provides guidance for modeling a subsystem's structure, shown in Figure 128.



Figure 128. MagicGrid v.1 QuickStart Template guide for modeling Subsystem Structure

b. Information/Data Collected

Table 19. Subsystem Structure Information or Data Collected

Information Gathered	Artifacts/Information Mode	Source
Driver Tree Subsystem Structure Information	Driver Tree UML class diagrams	Booz Allen Hamilton (BAH)
	Miscellaneous Driver Tree graphical information	Booz Allen Hamilton (BAH)
SIMLOX Prep and Analysis tool suite Subsystem Structure Information	SIMLOX Prep & Analysis tool suite graphics from presentations	Institute for Defense Analyses (IDA)

c. Modeling Steps and Results

The author collected structural information for the Driver Tree model and the Institute for Defense Analyses (IDA) suite of tools for preparing SIMLOX models labeled by the author as the "SIMLOX Prep. & Analysis Suite." This information was translated into two SysML internal block diagrams (ibds). One ibd captures the "Driver Tree" structure, and the other ibd captures the "SIMLOX Prep. & Analysis Suite" structure. Information that was useful for developing these diagrams was pasted in the corresponding diagram canvas.

Results for the "Driver Tree" structure is presented in Figures 129 and 130. The UML class diagrams provide the best structural information about the Driver Tree model's internal structure. The other graphical artifacts portray the current Driver Tree context structure and includes elements external to driver tree. It is unclear how much of this information will be applicable in the future fully integrated and automated CRIM state; they are pasted here for reference but are not very useful for providing internal details of the Driver Tree model.

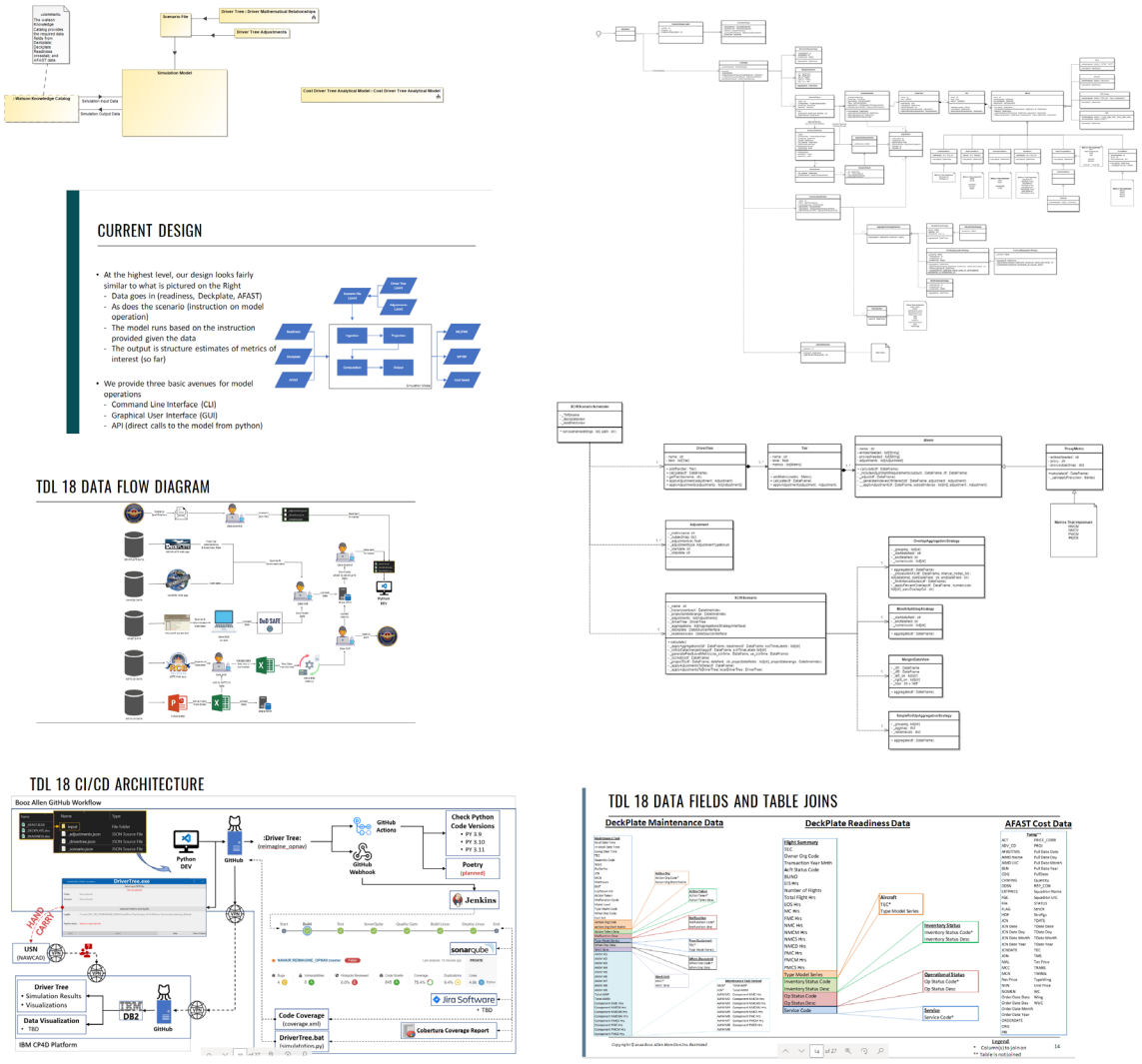


Figure 129. Driver Tree Subsystem Structure with pasted artifacts.

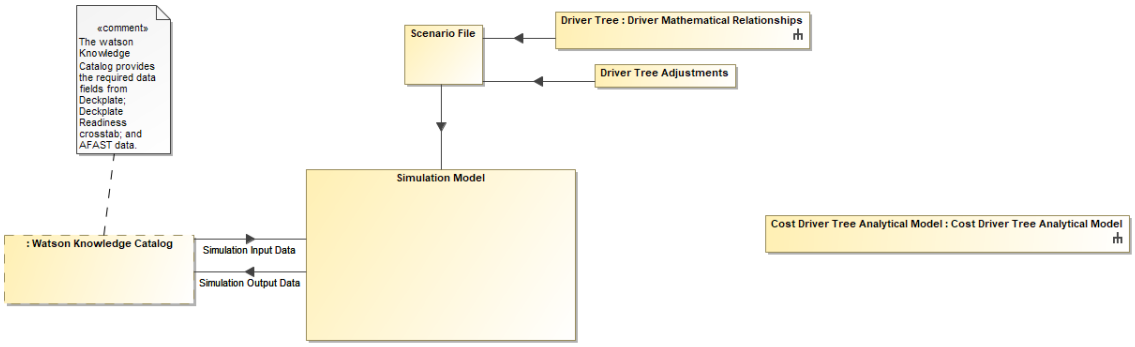


Figure 130. Driver Tree Subsystem Structure, SysML model elements only.

The results of the "SIMLOX Prep & Analysis Suite" are presented in Figures 131 and 132. The major components for this suite of tools are modeled as parts of this suite subsystem. The "SIMLOX" simulation model was not modeled because the author does not have any knowledge of this component's internal structure. In theory, the SIMLOX model **is** the Multi-Echelon, Multi-Indenture discrete event Monte Carlo simulation system. SIMLOX captures the Fleet Operations & Maintenance (O&M) multi-echelon levels, i.e., the Operational, Intermediate, and Depot Level (O-I-L) repair and maintenance echelons. Additionally, SIMLOX captures the aircraft system's "indenture structure," which is the modeled NIINs or parts that the Institute for Defense Analyses (IDA) was able to get "component-level" information and are incorporated into the simulation model.

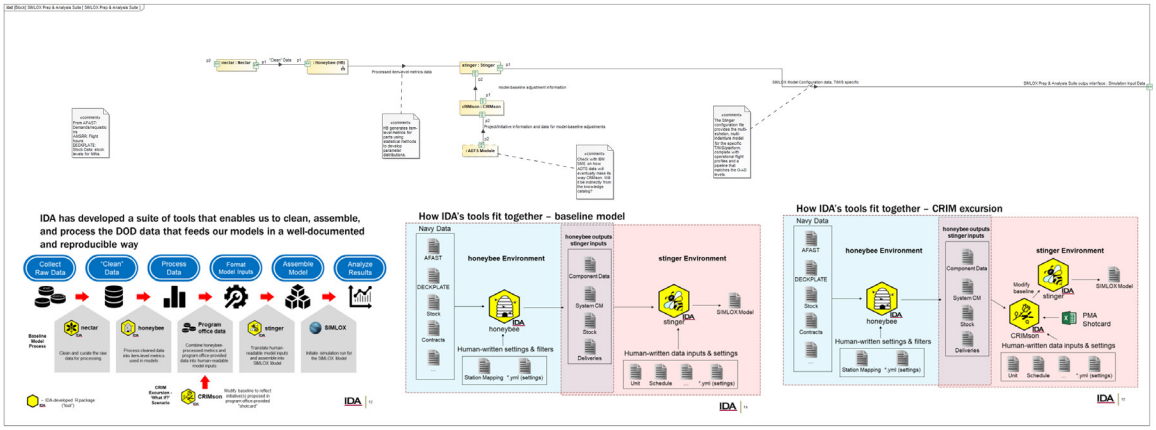


Figure 131. SIMLOX Prep. & Analysis Suite Subsystem Structure with pasted reference artifacts.

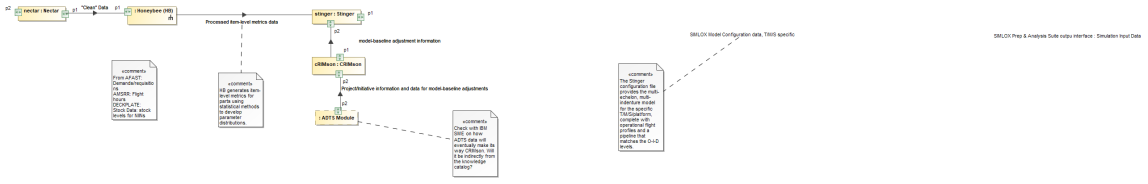


Figure 132. SIMLOX Prep. & Analysis Suite Subsystem Structure, SysML model elements only.

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

- Gather additional internal structure information for both Driver Tree and the SIMLOX Prep. & Analysis Suite of tools.
- Gather more information about how the SIMLOX multi-echelon, multi-indenture model is structured. For example, each multi-echelon, multi-indenture model of a specific aircraft Type/Model/Series (T/M/S) is built like a network with arcs connecting element nodes. There should be one custom multi-echelon, multi-indenture model for each specific aircraft T/M/S.
- Review the current model with the CRIM development team and get clarifications on the information.

18. Modeling Subsystem Parameters

a. Modeling Objectives and Guidance

For this step, the subsystem parameters are defined and modeled. An analogy for defining a subsystem parameter would be as follows:

- The system in question is a fighter/interceptor aircraft; the system parameter or performance measure is the "rate of climb" in feet/minute. Subsystem parameters that may impact the rate of climb are the power and propulsion subsystem and the airframe subsystem. The power and propulsion subsystem has a performance parameter for "thrust" which is a function of various engine factors such as engine speed in revolutions per minute. The airframe subsystem has non-functional parameters such as weight and aerodynamic drag measures. The subsystem parameters may cooperate or compete in how they contribute to the emergent system level performance parameter. In the case of thrust vs. weight and drag, these are competing parameters that affect the resulting system level rate of climb

performance, e.g., more thrust increases rate of climb, more weight and drag decreases rate of climb.

The Driver Tree and SIMLOX models operate independent of each other. In a context agnostic of CRIM, these may be treated at the system level, and not as subsystems. The (S4) System Parameters were developed for a simulation model system. Therefore, those parameters carry through to this level and can be applied here, to both the Driver Tree and the SIMLOX models.

b. Information/Data Collected

No information was gathered for supporting the development of subsystem parameters.

c. Modeling Steps and Results

The author created a block definition diagram that models the CRIM subsystem's composition, which serves like a taxonomy. Two blocks were used to model subsystem parameters. One block for the Driver Tree subsystem and one block for the SIMLOS subsystem. These blocks are essentially placeholders that will serve as the subsystem parameter namespace for internal component level parameters. The results of this modeling effort are presented in Figure 133.

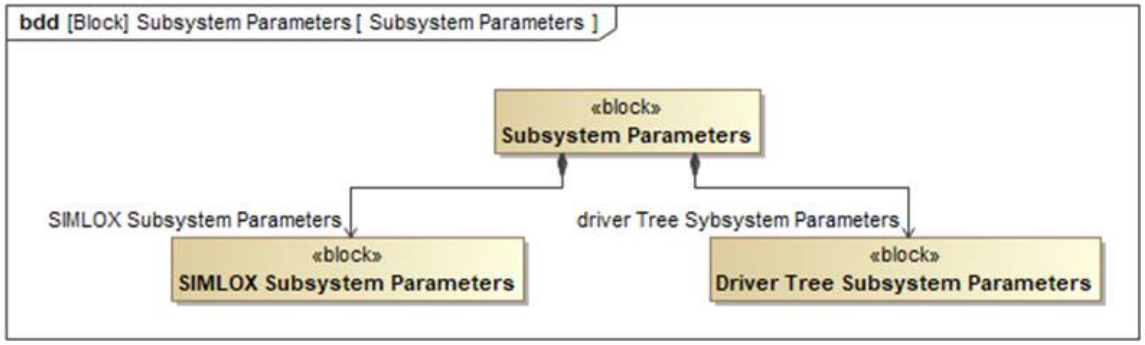


Figure 133. Subsystem Parameters bdd diagram.

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Identify the system parameters, or performance measures identified in the (S4) "System Parameters" grid that may be applied to the SIMLOX model and the Driver Tree model.
2. Review the allocation of subsystem parameters with the CRIM development team and the CRIM V&V team.

19. Modeling Component Structure

a. Modeling Objectives and Guidance

The goal is to model the internal structure of the system components. This is a relatively simple task if information about the internal details of the components is known.

b. Information/Data Collected

Table 20. Component Structure Information or Data Collected

Information Gathered	Artifacts	Source
Honeybee component structure information	Honeybee Mermaid charts (Mermaid is an open-source diagramming and charting tool)	Institute for Defense Analyses (IDA)
	Honeybee cartoon graphics from a tutorial presentation	Institute for Defense Analyses (IDA)
Driver Tree mathematical relationships	Graphic of the Driver Tree mathematical relationships	Booz Allen Hamilton (BAH)

c. Modeling Steps and Results

The author gathered as much information as possible that can be used for modeling the internal structure of components. The author was able to collect information on Honeybee and the Driver Tree mathematical relationships. Two components out of the many that exist are modeled here—Honeybee is a component of the SIMLOX Prep. & Analysis Suite, and the Driver Tree mathematical relationship model is a component of the

overall Driver Tree model subsystem. There is plenty of modeling work ahead to achieve a fuller system model.

An internal block diagram (ibd) was created from the Honeybee part. The author pasted all the artifacts that may be useful to infer Honeybees internal structure. The artifacts here are more behavioral in nature, and not structural. These diagrams may give insights into the inputs and outputs of Honeybee. This information can be used to model data flows into and out of Honeybee. However, the author noticed that "Nectar" which is the tool that directly ingests the raw source data, filters, cleans it, and passes it along to Honeybee is not shown in these diagrams; it is abstracted out. The author believes that the logical input relationships shown in the Honeybee Mermaid Charts are correct, but the author is uncertain of how "Nectar" plays a role. Therefore, the author needs clarification from the Institute for Defense Analyses. The author used this diagram as the namespace for collecting information on Honeybee. The results of this modeling effort are presented in Figure 134.

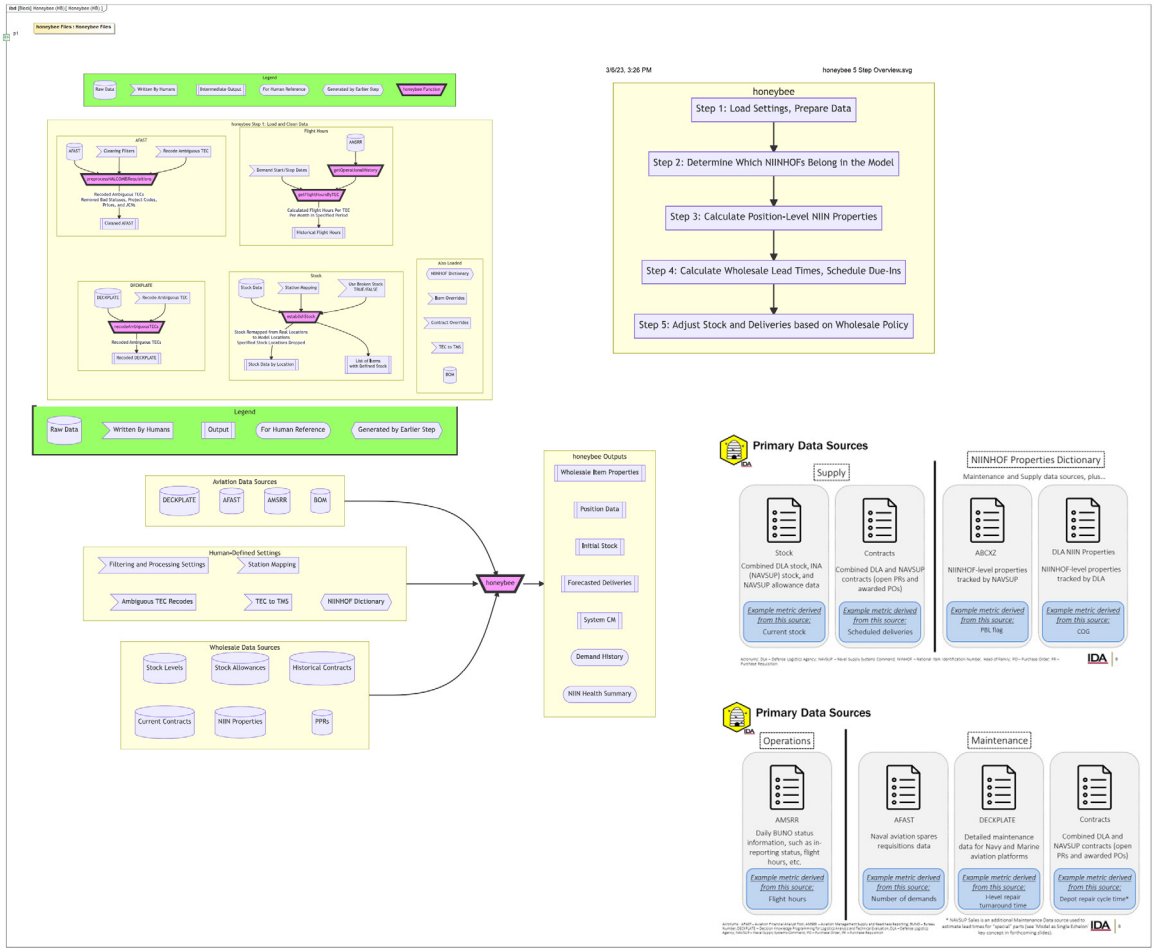


Figure 134. Honeybee Component Structure

The author had limited time and resources to develop the internal structural models of all the Driver Tree components. The Driver Tree mathematical relationship model is the deterministic framework that is used by the Driver Tree simulation to compute results. Therefore, the mathematical relationship model component is of high interest for V&V purposes.

The author is unsure of how input values are entered into the Driver Tree model or what the output results look like. The author created an internal block diagram (ibd) for the Driver Tree mathematical relationship model and pasted a graphical representation of the Driver Tree mathematical relationship model into the diagram. This diagram effectively serves as the namespace where the author can paste or model additional information about

this component. The results of this modeling effort are shown in Figure 135. No SysML elements, other than the diagram, were generated during this step.

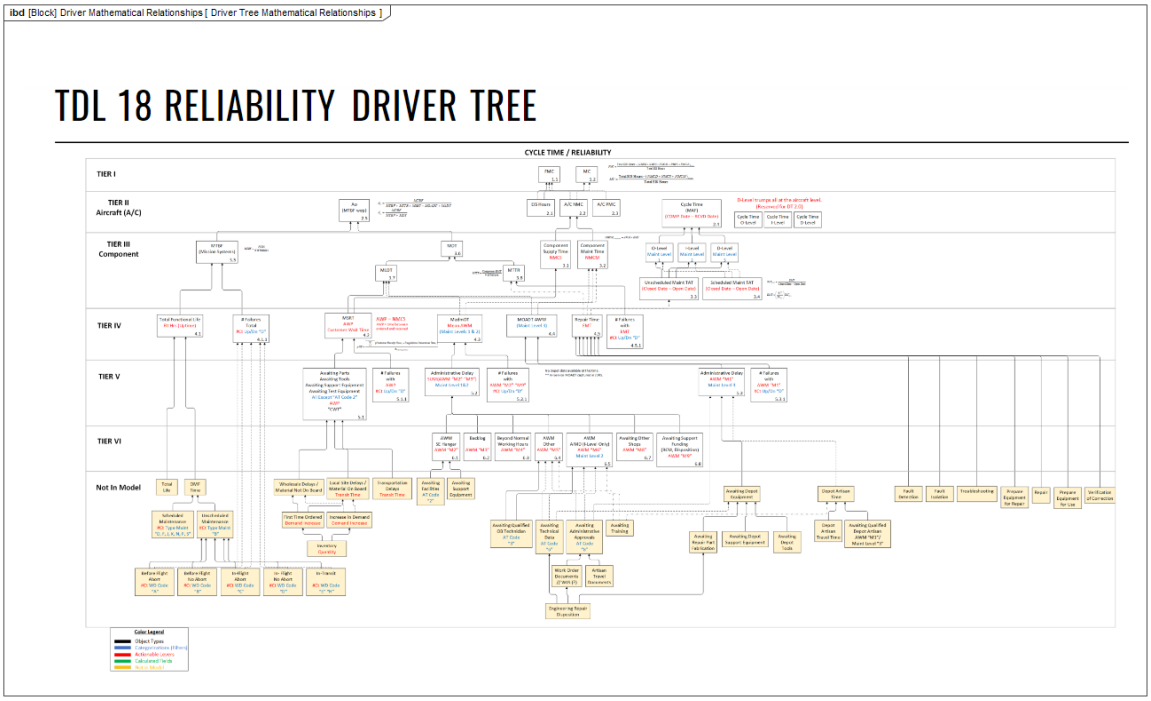


Figure 135. Driver Tree Mathematical Relationships model internal block diagram, only showing a pasted graphical artifact for this component.

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Create a Honeybee component behavior diagram that captures the functions of Honeybee as depicted in the Honeybee artifacts pasted in the Honeybee component structure diagram.
2. Model the internal structure of the remaining CRIM components.
3. Review the model with the CRIM development team and collect additional details and clarifications.

20. Modeling Component Parameters

a. Modeling Objectives and Guidance

The objective of this task is to model component level parameters. To achieve this, individual components must be identified, and their corresponding parameters must be modeled here.

b. Information/Data Collected

Table 21. Component Parameters Information or Data Collected

Information Gathered	Artifacts	Source
Honeybee component structure information	Honeybee Mermaid charts (Mermaid is an open-source diagramming and charting tool)	Institute for Defense Analyses (IDA)
	Honeybee cartoon graphics from a tutorial presentation	Institute for Defense Analyses (IDA)
Driver Tree mathematical relationships	Graphic of the Driver Tree mathematical relationships	Booz Allen Hamilton (BAH)

c. Modeling Steps and Results

The author identified various key component level functions performed by the SIMLOX model components and the Driver Tree model components. The author created two internal block diagrams to serve as the namespaces for component parameter information. The author used these diagrams to collect component parameter related information by pasting information useful or related to component level parameters. The author then made comments in the diagram to address the artifacts pasted in the model.

The ibd that collects SIMLOX related component parameter information is discussed first; shown in Figure 136. The author knows that some component generates operational flight profile information which is used in the simulation to mimic the flight profile of the fleet's aircraft (for a particular type/model/series). This component is likely Honeybee, but the author is not completely sure. One performance parameter measure for this function could be the degree of realism, fidelity or accuracy of the machine (model)

generated flight profile. Model generated flight hours may be statistically compared to the fleet flight hour data. A Turin test may be a good way to validate model generated flight profiles—a subject matter expert will look at the flight profile and try to distinguish if it was generated by a machine or if it represents real aircraft flight data. Statistical tests may also be a good way to validate that the model component generated flight profile is within an acceptable parametric range of performance, in this case some statistical significance. The modeling results are shown in Figure 137.

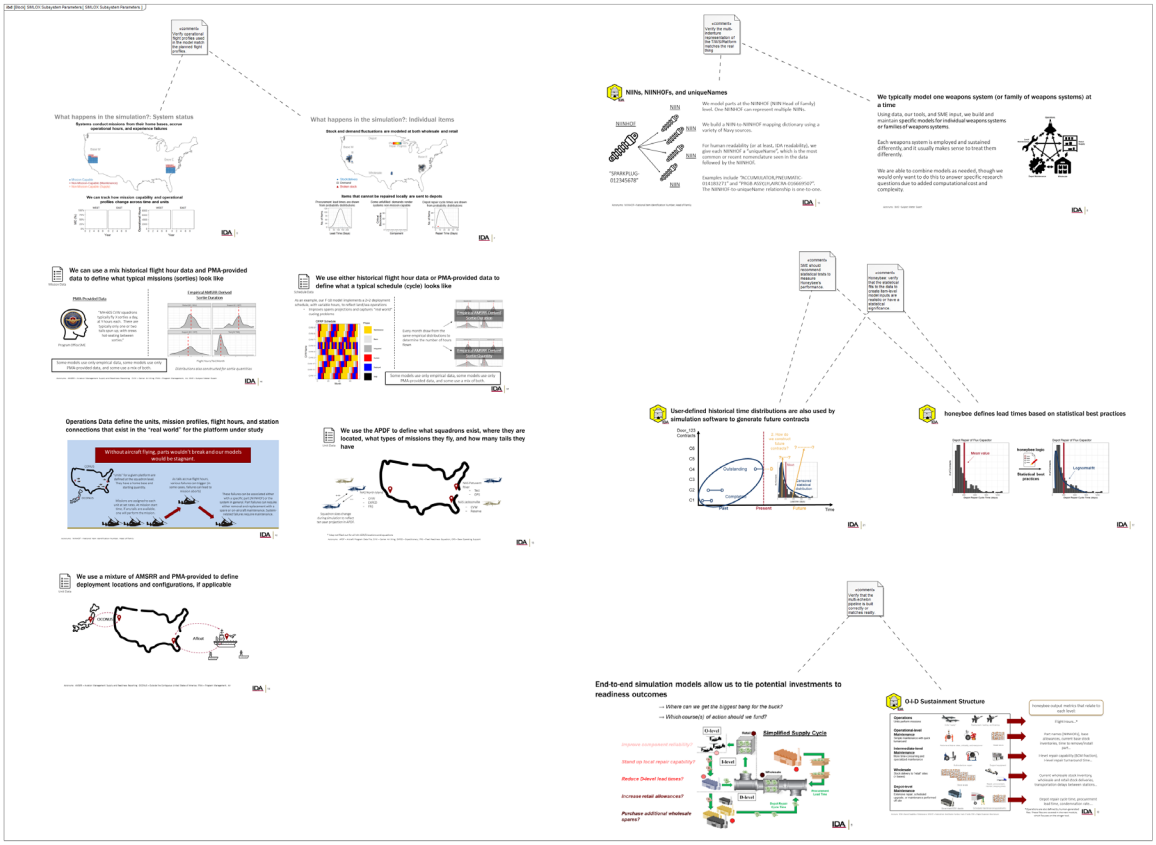


Figure 136. SIMLOX Multi-Echelon, Multi-Indenture model component parameters

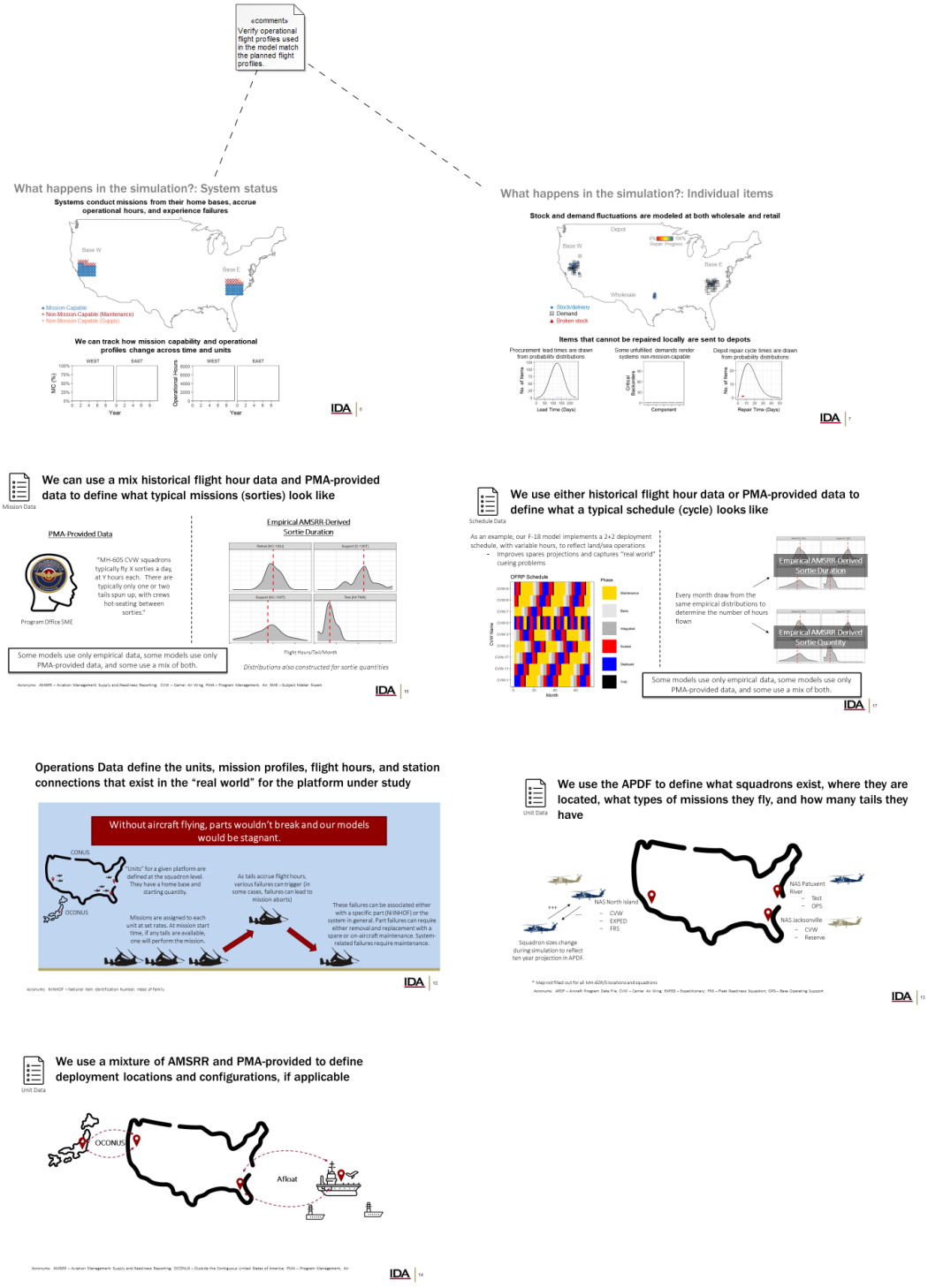


Figure 137. Model generated Operational Flight Profile performance parameter.

Another component parameter could be Honeybees' ability to generate the multi-indenture model of the aircraft system. This may be evaluated more qualitatively. Aircraft system subject matter experts can evaluate the parts list generated by Honeybee and determine its validity with respect to the level of realism or significance of the parts. The author has heard anecdotes that sometimes the parts that keep the aircraft from flying are simple nuts and bolts and not other major aircraft parts. However, nuts and bolts have low visibility in the supply and inventory system. The modeling results for this parameter are shown in Figure 138

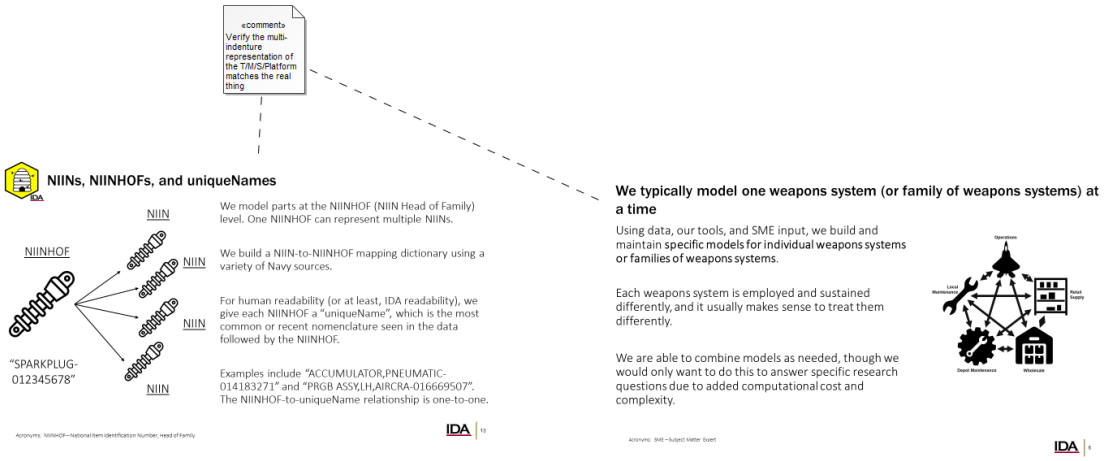


Figure 138. Model generated Multi-Indenture validity performance parameter.

The author knows that Honeybee generates statistical distributions from Navy data. The author does not know the fully enumerated list of distributions that Honeybee generates, but the author captures this type of general function that Honeybee performs in this ibd. The performance parameter for this distribution generating function can be the statistical accuracy of the distribution. Subject matter experts can independently verify the statistical distribution fits that Honeybee is generating. The modeling results for this parameter are shown in Figure 139.

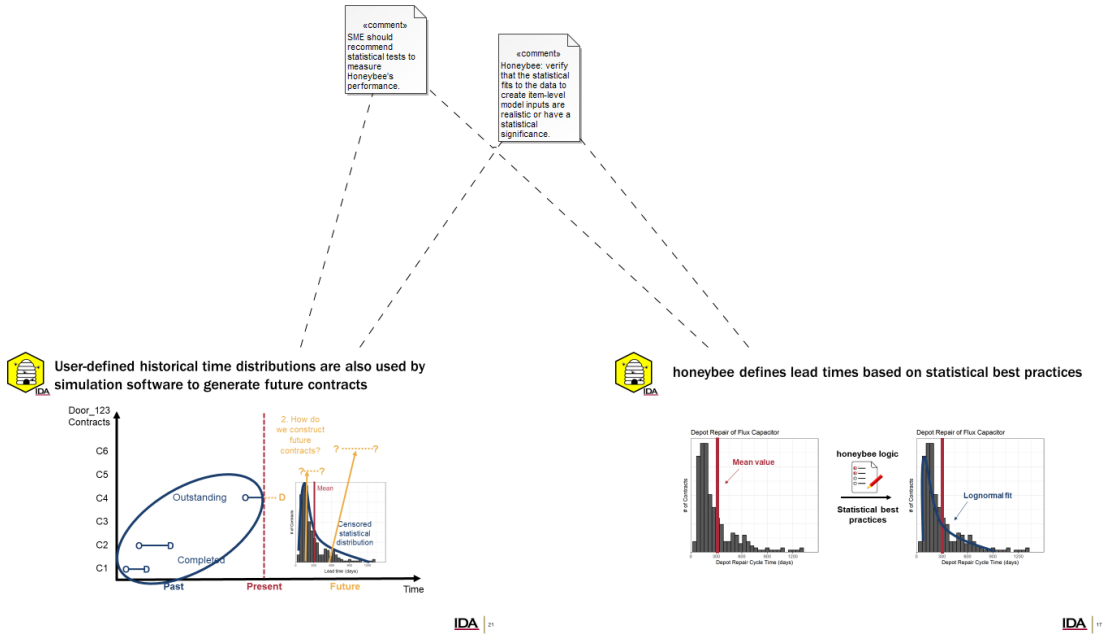


Figure 139. Model generated metric distributions performance parameter

The author understands that the SIMLOX model configuration simulates discrete events that follow logical processes based on the multi-echelon, multi-indenture model structure. The author does not know the level of complexity that is built into the multi-echelon, multi-indenture network or the laws that govern its events and operations. The author uses this diagram as a namespace to collect information on the multi-echelon, multi-indenture model. A parameter for the multi-echelon, multi-echelon model can be the degree of realism possessed by this model. An independent subject matter expert may analyze and evaluate the model by comparing it with the real-life physical entities and processes that occur in the fleet operations and maintenance system. Modeling results are shown in Figure 140.

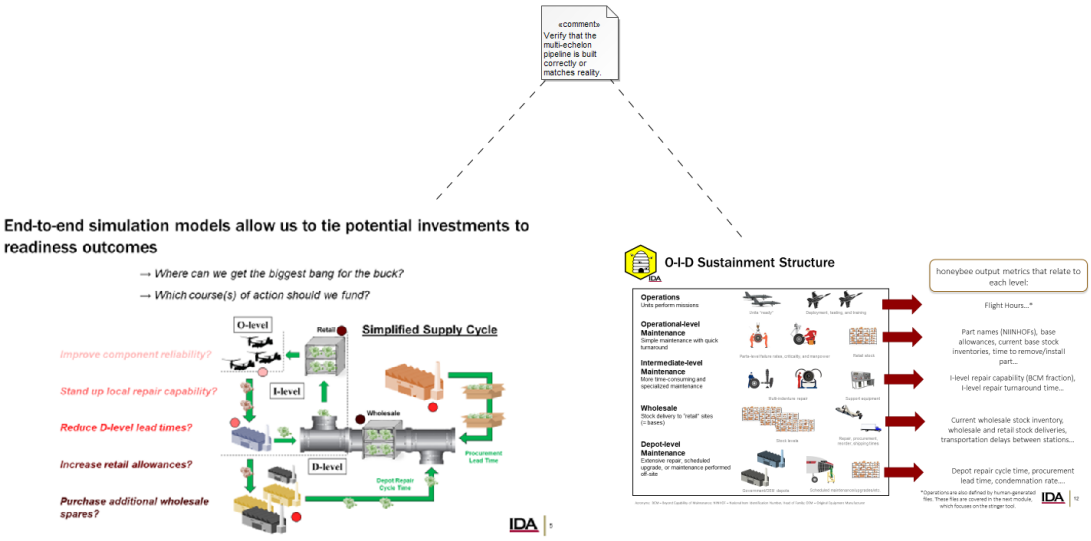


Figure 140. Multi-echelon, multi-indenture model performance parameters.

Performance parameter information related to Driver Tree model components is now presented. The author knows that the Driver Tree model performs a few algorithms during its solution computation. The author is uncertain of which components perform the functions referenced here, so the author generated comments that address the functions. The comments in the diagram are V&V centric. The parameter measures may be used to support the component level V&V. Performance parameters should be established for each of the functions mentioned here. These include:

- SCIR Time Overlap Functionality
- Data aggregation strategies
- Data filtering
- Simulation model baseline adjustment methods and profiles
- Availability calculations modeling
- Degradation modeling.

The modeling results for the Driver Tree component parameters are presented in Figure 141.

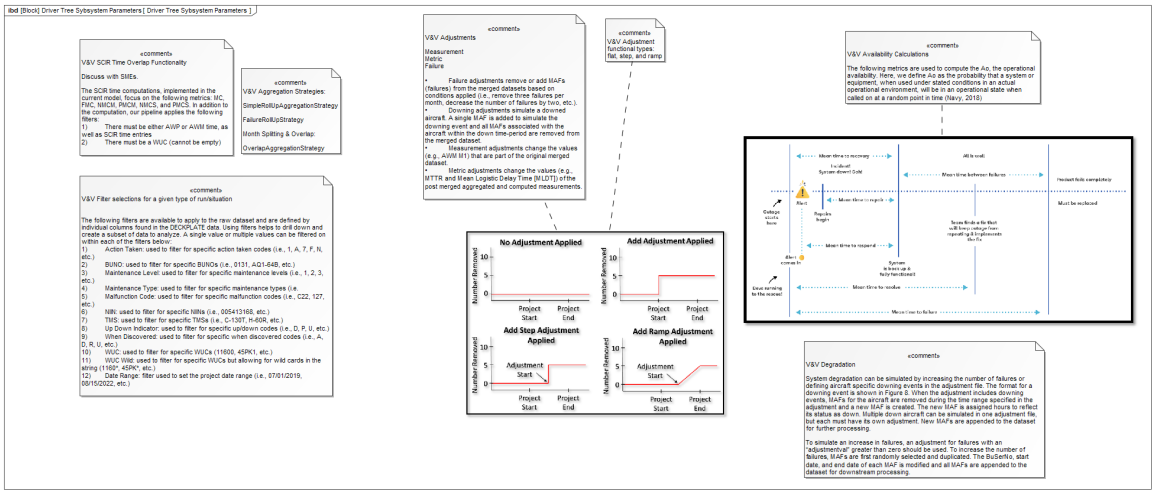


Figure 141. Driver Tree component parameters

d. Future Work

Listed below is an enumerated set of modeling recommendations or model details that could be improved or are deficient:

1. Precisely identify all of the components that should be referenced in the component parameter grid.
2. Discuss with the CRIM development team the type of parameters that may be allocated to model components.
3. Discuss with the CRIM development team and the CRIM V&V team how the parameters can be used to support V&V.

21. Other "Fit for Purpose Views"

This idea is borrowed from the Department of Defense Architecture Framework. "Fit-for-Purpose Views are user-defined views of a subset of architectural data created for some specific purpose" (Chief Information Officer n.d.a). The MagicGrid v.1 Framework ontology is limited to the type of intended information that goes inside each of its grids.

Some information that was useful to the author, for the purposes of VV&A, could not be neatly classified into the MagicGrid v.1 framework. Therefore, the author created standalone views.

The author with the help of ManTech developed the view that shows all the external data sources. Modeling these sources in the underlying model will come in handy to support future modeling efforts. The resulting view is captured in Figure 142.

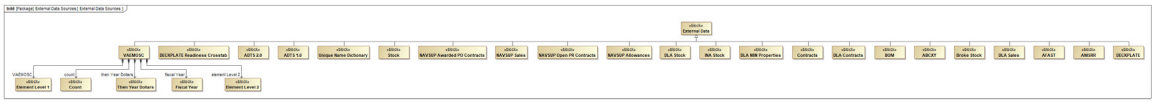


Figure 142. Navy External Data Sources view.

The author developed a view that captures the taxonomy of the Navy's funding accounts and captures additional information about each account using comments. This information is helpful because different accounts fund different types of projects. Therefore, the budget limits used in the results optimization depend on the budget limits for the respective accounts. Figure 143 shows the NAVAIR funding account view.

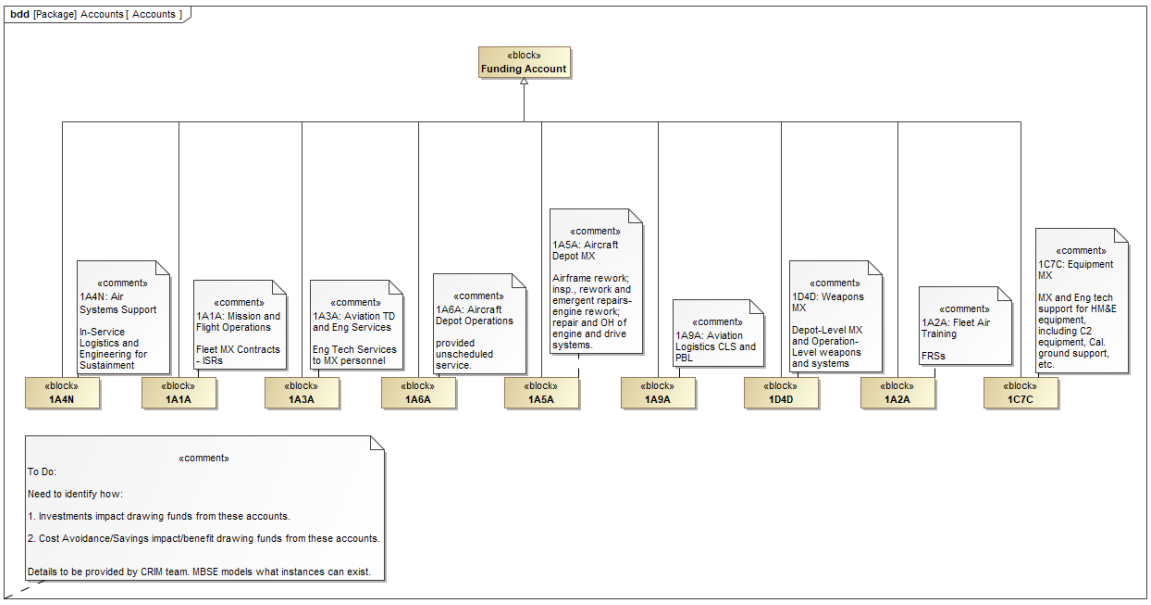


Figure 143. NAVAIR Funding Accounts view.

The following view, Figure 144, tries to provide a taxonomy that classes the various types of data. The thought is that these data types can be referenced in the data flows between elements. This view was created by ManTech modelers.

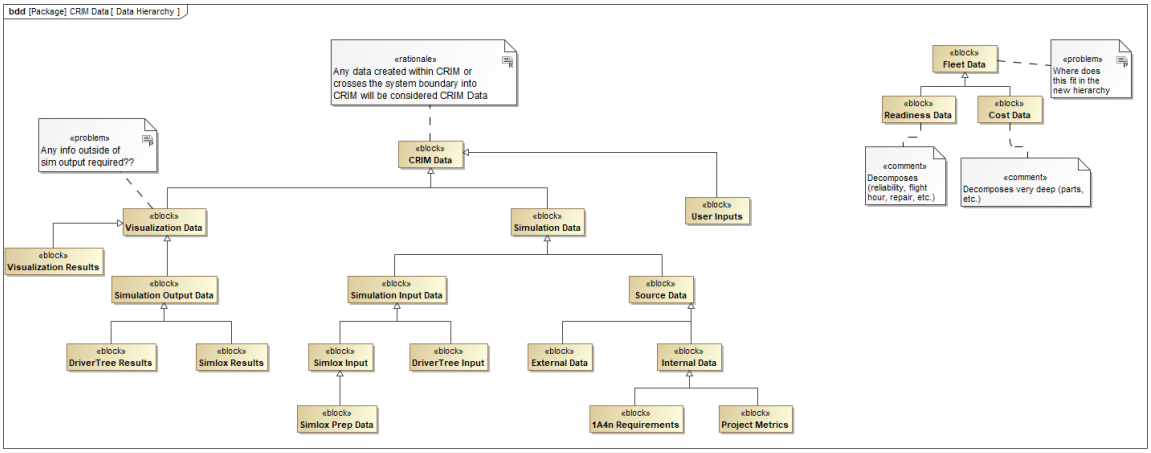


Figure 144. Taxonomy of Data classes.

The following view, Figure 145, captures the set of CRIM system actors that are expected to interact directly or indirectly with CRIM.

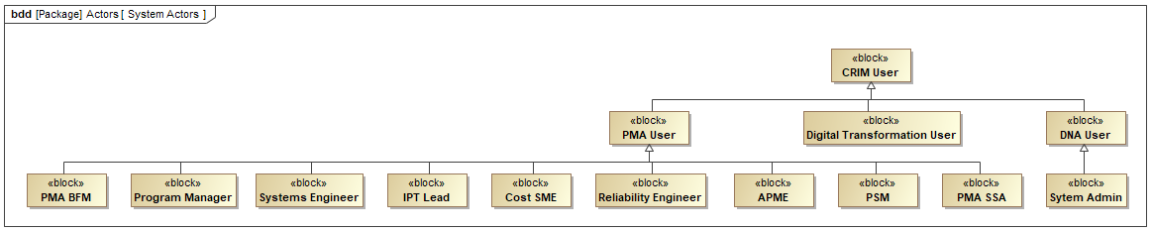


Figure 145. CRIM System Actors view

The following view, presented in Figure 146, is intended to capture stakeholders or their organizational entity and how they relate to the VV&A roles. The VV&A roles captured here: the "V&V Agent," "V&V Team," "M&S Proponents," "Validation Authority," and "Accreditation Authority" are the identified "V&V Participants" listed in

the DOD's *Documentation of Verification, Validation, and Accreditation (VV&A) for Models and Simulations, MIL-STD-3022 w/CHANGE 1* (Department of Defense 2012, 25).

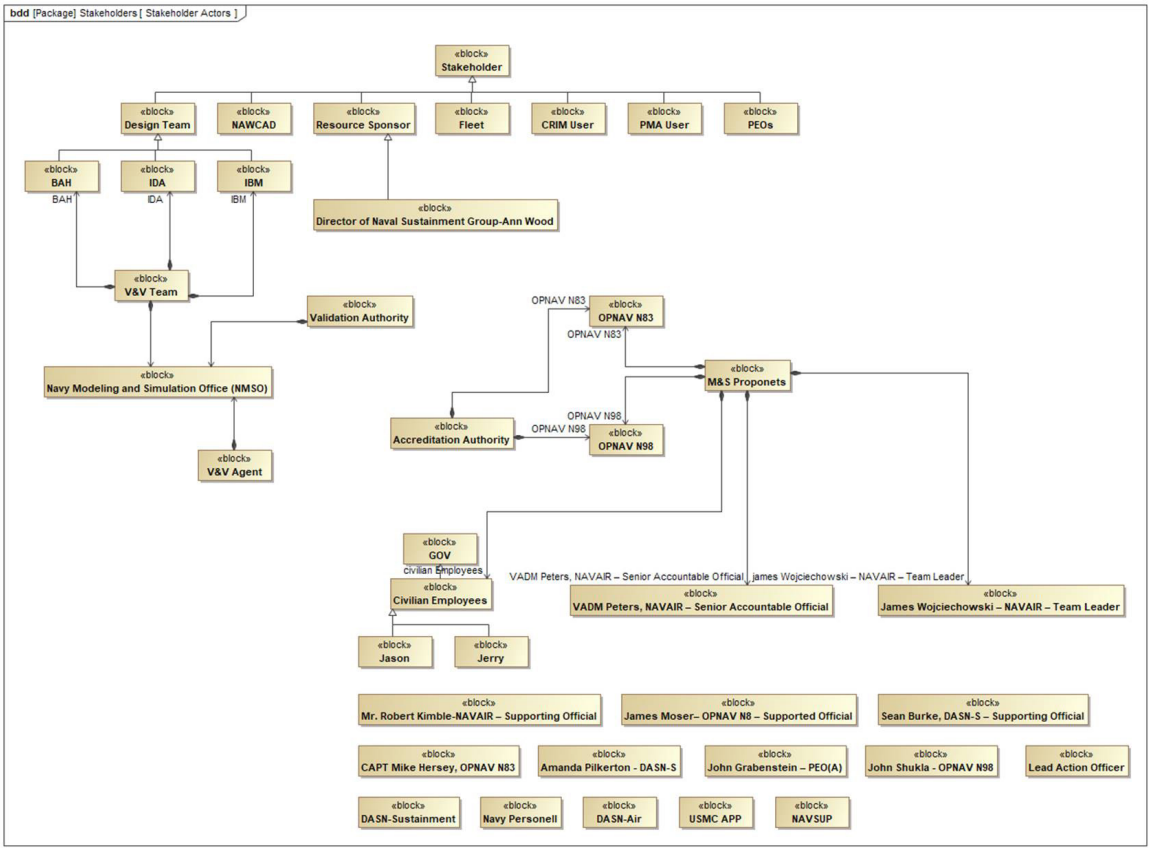


Figure 146. CRIM stakeholders view

IV. RESULTS ANALYSIS AND INTERPRETATION

This chapter analyses the results obtained from the methodology described in the previous chapter. The result of the methodology is an MBSE descriptive model of the Cost Readiness Impact Model (CRIM). Diagrams or views of the model are presented in Chapter III—these views provide the MBSE model viewers semantically rich visualizations of the CRIM system architecture that convey details about CRIM's elements and their relationships. George Box is widely known by his quote, "all models are wrong but some are useful." In this chapter the author does not analyze how the MBSE model of CRIM is "wrong," but rather, the author analyzes how the model might be useful from an M&S VV&A perspective.

This thesis research is useful in two ways:

1. This research demonstrates a generally applicable or reusable methodology for deriving an M&S VV&A framework. Currently there are no published guides that detail the framework or steps required to perform M&S VV&A. The Department of Defense Standard Practice, Documentation of Verification, Validation and Accreditation (VV&A) for Models and Simulation MIL-STD-3022 W/CHANGE 1 (Department of Defense 2012) primarily provides documentation product guidance but does not provide a methodology that can be used to develop a V&V plan or framework.
2. This research provides the resulting product from this methodology; CRIM's MBSE model. This model may effectively serve as the M&S VV&A framework for CRIM.

To prove or support how this research is useful, according to the two ways listed above, the author poses the following main analytical questions:

- General Question: Can this method be applied (re-used) to any M&S system and provide a useful M&S VV&A framework that identifies the M&S system entities and corresponding VV&A elements?

- Specific Question: Does the resulting MBSE model provide an ontological mapping or framework of the M&S system (CRIM) that structures and defines M&S entities in a manner that is helpful for V&V planning and for precisely tracing or assigning V&V methods or actions?

These analytical questions may be regarded as the two main effectiveness questions that must be answered to prove the utility of this research. Descriptive models such as the MBSE model developed in this thesis are mostly qualitative in nature, thus are suited for qualitative evaluation methods. The author developed a qualitative rubric that can be used to evaluate the results obtained in this research. The rubric provides a more structured evaluation for answering the two main effectiveness questions and aims to remove subjectivity in the evaluation. This is achieved largely by the rubric's framework which sets evaluation criteria boundaries and qualities. The MBSE model results are evaluated against these evaluation criteria and are ranked based on the rubric's scoring structure.

Due to size, the rubric is split into several parts. The overall rubric consists of 18 effectiveness criteria which are further grouped and classified by broader domain questions. The rubric parts are presented in Figures 147, 148, and 150.

A. RESULTS EVALUATION: CRITERIA 1 TO 4

This set of criteria supports answering the parent model effectiveness question of:

- Does the model provide an (ontological) mapping or framework of the M&S system that defines and structures M&S entities in a manner that is helpful for V&V planning and for precisely tracing or assigning V&V methods or actions?

The parent question for this set derives two effectiveness criteria that the model must satisfy. To satisfy these criteria, the model must:

1. Identify the M&S entities at various system boundary levels, and
2. Assign V&V methods or actions to M&S element.

The rubric further expands sub parts of these two criteria, which are captured as criteria 1 through 4, shown in Figure 147.

Effectiveness Criteria	Effectiveness Sub-criteria/Measure	Sub-criteria/Measure	Excellent	Good	Poor or Needs Work
Does the model provide an (ontological) mapping or framework of the M&S system that defines and structures M&S entities in a manner that is helpful for V&V planning and for precisely tracing or assigning V&V methods or actions?	Identify the M&S entities at various system boundary levels	1	Structural system boundaries are defined at the system, sub-system, component level and beyond, as required.	Structural system boundaries are defined at the system and sub-system, enough to roughly map and define the M&S results, which are needed for Operational (results) Validation.	The structural system boundary is defined only at the system level and represented by a black-box abstraction of the system.
		2	The model decomposes data flows at the system, sub-system, component level and beyond, as required. The objective is to remove abstraction and define the discrete input/output data flows.	The model decomposes data flows at the system, sub-system, component level and beyond, as required. Discrete input/output data flows are not defined, but model elements are created to represent or reference a more concrete artifact that does capture this level of detail.	Data flows are defined at the system level, but abstracted to a single flow showing the general type. System outputs (results) are abstracted and shown as a single entity. At this level, no information is given about the subtype of system outputs or data results.
		3	The model decomposes behavior and functions at the system, subsystem, component level and beyond to the required discrete function level. The model identifies associated discrete input/output data flows.	The model decomposes behavior and functions at the system, subsystem, and component level. The model definition of discrete level functions is not achieved.	The model defines the system's behavior at the system (black-box) level.
	Assign V&V methods or actions to M&S elements	4	V&V methods or actions are assigned at the system, subsystem, component, discrete function, and discrete input/output data flow levels.	V&V methods or actions are assigned at the system and subsystem.	The model decomposes behavior and functions at the system, subsystem, and component level. The model definition of discrete level functions is not achieved.

Figure 147. Qualitative Evaluation Rubric for sub-criteria/measures 1–4 of 18 total.

1. Criteria 1. Evaluation of the CRIM MBSE Model System Boundaries

2. SCORE: Excellent

The MBSE model does an excellent job at capturing the various system boundaries or partitions of CRIM. System boundaries were defined at the system level, the subsystem, level, and the component level. These boundaries are useful to isolate sub elements of interest and provide focused information at that boundary level.

3. Criteria 2. Evaluation of the CRIM MBSE Model Data Flows/Interfaces

4. SCORE: Good

The model does a good job at capturing data/flow information. The model does a better job at making sure that the interfaces or data flow connections are identified through the various system hierarchy levels, but the data flows are abstracted, and the model does not capture the detail required to provide "discrete" flow elements. This is not a fault of the model's capabilities or features, but rather a fault of the modeler's lack of knowledge of the CRIM data flows. Given enough information, the modeling tool and language provides the ability to capture this level of detail.

"Discrete" values for inputs and outputs are essential for performing V&V. these values provide the concrete numerical measures that are needed to support some type of quantitative analysis.

5. Criteria 3. Evaluation of the CRIM MBSE Model System Behavior

6. SCORE: Good

The model does a good job at defining the system's behavior at the system, and subsystem level. The component level behavior should be refined and more completely captured for all components; right not the model captures only the components that the author considers critical.

The model did not achieve a functional decomposition level that captures the most "discrete" level functions; this is why the model did not score as excellent. The author doesn't consider discrete level functions to be as essential for supporting V&V compared

to the discrete inputs and outputs. However, having a concrete understanding of the discrete function that transforms the discrete inputs into outputs is useful for isolating that function's performance. Additionally, V&V methods can be focused on that discrete function.

7. Criteria 3. Evaluation of the CRIM MBSE Model V&V entities

8. SCORE: Good

The model does a good job at capturing V&V entities at the system and subsystem level. The author modified the MagicGrid v.1 Framework's Parameters Pillar and adopted it as the "Parameters (and V&V)" Pillar. This pillar captures V&V entities at the system, subsystem and even the component level. This approach performs a rough allocation of V&V elements to the various system hierarchy levels, but the author did not perform full traceability modeling of V&V elements to specific system elements. This type of full V&V traceability modeling can support the development of the V&V plan and the specific V&V methods and actions that must be performed on specific system elements.

B. RESULTS EVALUATION: CRITERIA 5 TO 11

This set of criteria supports the parent model effectiveness question of:

- Does the model support development of the M&S V&V plan? According to the Documentation of Verification, Validation, and Accreditation (VV&A) for Models and Simulations, MIL-STD-3022 w/CHANGE 1 (Department of Defense 2012)

This parent effectiveness question is further decomposed so that it addresses the specific V&V plan sections, outlined by the MIL-STD-3022, in its "TABLE I. Outlines of four core VV&A documents" (Department of Defense 2012, 5):

1. V&V (Plan) Problem Statement definition
2. M&S Requirements and Acceptability Criteria
3. M&S Assumptions, Capabilities, Limitations & Risks/Impacts
4. V&V methodology (similar or equivalent to the "Assign V&V methods or actions to M&S elements" effectiveness criteria; criteria number 4)
5. V&V Issues

6. Key (V&V) Participants
7. Planned V&V Resources

These seven V&V plan sections effectively derive the next set of effectiveness criteria, shown in Figure 148.

Effectiveness Criteria	Effectiveness Sub-criteria/Measure	Sub-criteria/Measure	Excellent	Good	Poor or Needs Work
Does the model support development of the M&S V&V plan? According to the Documentation of Verification, Validation, and Accreditation (VV&A) for Models and Simulations, MIL-STD-3022 w/CHANGE 1 (DoD 2012)	1. V&V (Plan) Problem Statement definition	5	The model defines the M&S problem domain: it defines the general stakeholder needs and identifies the M&S intended purpose and applications (use cases).	The model defines the stakeholder needs and intended purpose for the model, but does not capture applications (use cases) of the M&S system.	The model roughly captures the stakeholder needs.
	2. M&S Requirements and Acceptability Criteria	6	The model captures measures of effectiveness (MoEs), measures of performance (MoPs) and other M&S system attributes that satisfy some acceptability criteria. These measures provide details on *what* the system must do to be effective and *how well* it must perform the identified effectiveness measures captured.	The model captures MoEs and other supporting M&S system attributes that satisfy some acceptability criteria.	The model captures only MoEs or M&S system attributes that satisfy some acceptability criteria. The model details are high-level and need refinement.
	3. M&S Assumptions, Capabilities, Limitations & Risks/Impacts	7	The model captures M&S assumptions, capabilities, limitations and risks/impacts of the M&S system theories and the conceptual model. Assumptions, capabilities, limitations, and risks/impacts are captured at the M&S system, subsystem, and component level to provide clear mapping and traceability.	The model captures M&S assumptions, capabilities, limitations and risks/impacts at the M&S system and subsystem level.	The model does not capture assumptions, capabilities, limitations and risks/impacts.
	4. V&V methodology (similar or equivalent to the "Assign V&V methods or actions to M&S elements" effectiveness criteria)	8	The model captures V&V methodology at the system, subsystem, and component level. M&S entities are identified and associated to suitable V&V methodologies.	The model captures V&V methodology at the system and subsystem level.	The model captures V&V methodology only at the system level.
	5. V&V Issues	9	The model captures V&V Issues information at the system, subsystem and component level.	The model captures V&V Issues information at the system level.	The model does not capture V&V Issues information.
	6. Key (V&V) Participants	10	Identify typical V&V roles, such as the M&S Proponents, V&V Agent, V&V Team, Validation Authority, Data Source, and SMEs.	The model identifies the V&V team, Validation Authority, Data Sources, and SMEs.	The model identifies the Data Sources and SMEs.
	7. Planned V&V Resources	11	The model identifies the allocation of *who* (which participant) is doing *what* V&V process, what method, tools and resources are needed.	The model identifies the allocation of who (which participant) is doing what V&V process.	The model does not capture Planned V&V Resource information.

Figure 148. Qualitative Evaluation Rubric for sub-criteria/measures 5–11 of 18 total.

1. Criteria 5. Evaluation of the CRIM MBSE Model to support the V&V Plan's Problem Statement Definition

2. SCORE: Excellent

The model meets the features expressed in the "excellent" category: the model defines the M&S problem domain, it defines stakeholder needs, M&S purpose, and identifies M&S use cases. However, the author believes that these elements must be validated by the appropriate CRIM stakeholders. The author believes that the quality of details that was captured for this category's features is good, but not excellent. So, there is some subjectivity involved. The advantage of modeling these features is that even if they require refinement, they exist which lends themselves to be reviewed and refined by the appropriate stakeholders.

3. Criteria 6. Evaluation of the CRIM MBSE Model to support the V&V Plan's M&S Requirements and Acceptability Criteria

4. SCORE: Excellent

The model does an excellent job at capturing CRIM's Measures of Effectiveness (MoEs) and Measures of Performance (MoPs), as well as other system attributes that satisfy acceptability criteria. Acceptability criteria have not been formally modeled, but these are understood to be the systems MoEs and MoPs. Additionally, the stakeholder needs and system requirements are the effective M&S requirements. This information is also captured in the model and is associated with the corresponding MoEs and MoPs.

5. Criteria 7. Evaluation of the CRIM MBSE Model to support the V&V Plan's M&S Assumptions, Capabilities, Limitations and Risks/Impacts

6. SCORE: Poor or Needs Work

The model does not explicitly capture the M&S assumptions, capabilities, limitations, and risk/impacts. These can be implied from the model but are prone to subjectivity and interpretation error. Capabilities and limitations may be inferred based on the modeled use cases, MoEs and MoPs.

7. Criteria 8. Evaluation of the CRIM MBSE Model to support the V&V Plan's V&V Methodology.

8. SCORE: Good

This criterion is similar to criterion 4 "Assign V&V methods or actions to M&S elements." Assigning V&V methods to their respective M&S elements derives the V&V methodology that must be followed to fully execute the M&S V&V. The assessment for criteria 4, also applies here and is presented again here:

> The model does a good job at capturing V&V entities at the system and subsystem level. The author modified the MagicGrid v.1 Framework's Parameters Pillar and adopted it as the "Parameters (and V&V)" Pillar. This pillar captures V&V entities at the system, subsystem and even the component level. This approach performs a rough allocation of V&V elements to the various system hierarchy levels, but the author did not perform full traceability modeling of V&V elements to specific system elements. This type of full V&V traceability modeling can support the development of the V&V plan and the specific V&V methods and actions that must be performed on specific system elements.

9. Criteria 9. Evaluation of the CRIM MBSE Model to support the V&V Plan's V&V Issues

10. SCORE: Poor or Needs Work

The CRIM MBSE model does not capture V&V Issues. This is not a limitation of the model's capabilities. This information could be modeled to support the V&V plan's V&V Issues section. The author did not possess the knowledge and understanding necessary to understand what CRIM V&V issues could be. These issues have not been explicitly discussed or documented by the CRIM design team. The only main issue that has been discussed with respect to CRIM's ability to be used is *data availability* and *data quality*. It is difficult to get enough of the right data and format it to match the model's input configuration so that simulations can be performed. The author recognizes that this as an area where the MBSE model of CRIM needs improvement. To support the V&V Plan's V&V issues, the model can capture these issues, track their status, and resolutions.

11. Criteria 10. Evaluation of the CRIM MBSE Model to support the V&V Plan's Key (V&V) Participants definition.

12. SCORE: Excellent

The model does an excellent job at identifying stakeholders and their V&V roles. The author created a "Fit for Purpose" view that captures all CRIM's stakeholders and their associated V&V roles. The information in this view has not been validated by CRIM stakeholders so there may be errors. However, this is not a limitation of the model, but of the author's knowledge of these relationships.

13. Criteria 11. Evaluation of the CRIM MBSE Model to support the V&V Plan's V&V Resources

14. SCORE: Poor or Needs Work

The model does not capture planned V&V resource information. This is not a limitation of the model, but rather a limitation of the available existing information that the author could use to model V&V resource planning and allocation.

C. RESULTS EVALUATION: CRITERIA 12 TO 15

This set of criteria supports answering the parent, model effectiveness question of:

> Do M&S model elements reference or trace to the M&S entities identified by Sargent's (2013) M&S VV&A process framework?

The M&S entities identified in Sargent's (2013) circle are the: System (problem entity), Conceptual Model, and the Computerized Model. The author includes "Data Validity" as a fourth entity. Data is a component of the system entity. Data validity is a quality attribute or parameter of the data, so Data validity is not so much a process and that's why the author classified it as an entity. Figure 149 shows an illustration of Sargent's (2013) circle.

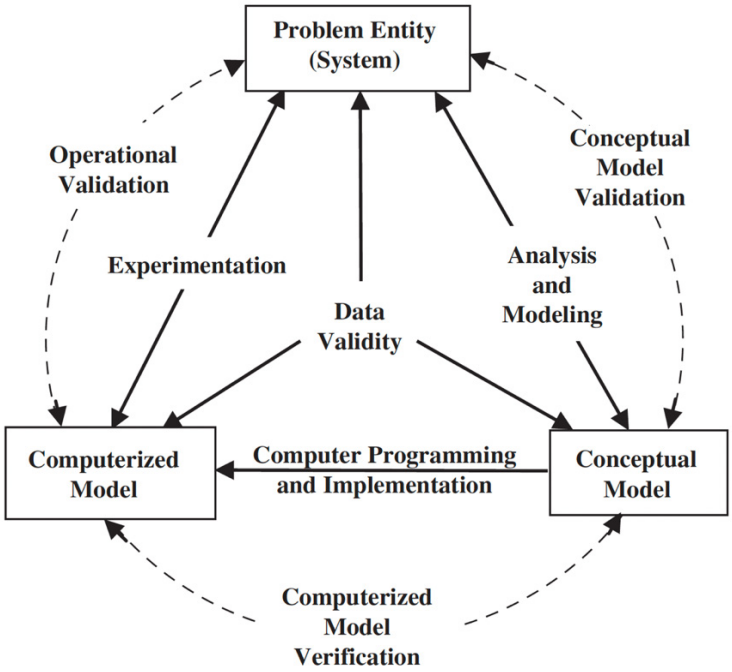


Figure 149. Sargent's simplified version of the model development process. Source: Sargent (2013, 14).

The rubric uses these M&S entities and their qualitative representation in the CRIM MBSE model as criteria for evaluating the model's effectiveness in relation to general M&S VV&A concepts. The rubric parts used for assessing the model's effectiveness in capturing the M&S entities are shown in Figure 150.

Effectiveness Criteria	Effectiveness Sub-criteria/Measure	Sub-criteria/Measure	Excellent	Good	Poor or Needs Work
Do M&S model elements reference or trace to the Sargent's (2013) M&S VV&A process framework M&S entities?	System (problem entity)	12	The model defines the system and the system data and underlying data sources and fields. Characteristics of the data are captured in the model. System data is used to hypothesize system theories and data value distributions. The system data is also used to validate the simulation results.	The model defines the sources of system data, and the respective data fields, but does not provide characteristics of the data types or properties.	The model defines the sources of system data, but not the underlying data fields or any further information.
	Conceptual Model	13	The model captures the underlying system theories and the conceptual model. The conceptual model is decomposed into its sub parts and assumptions.	The model captures the conceptual model. The conceptual model may be defined by type (black-box level) and high-level attributes, characteristics or assumptions. These convey some implication of the conceptual model identity and working principles, but are generalized and not specific to the M&S conceptual model of interest.	The model does not capture the conceptual model. Alternatively, the conceptual model may be roughly represented by the M&S system structure described by its main logical structural components - such as "data input module", "analytical simulation model", and "M&S results".
	Data Validity	14	The model identifies the data sources and data fields. The model identifies which of these are "authoritative" sources or follow some basic data governance principles or data hygiene. The model captures information about the data source that can be used to evaluate or judge the validity of the data.	The model identifies data sources.	The model does not capture data validity information.
	Computerized Model	15	The model abstracts the computerized model (simulation model) by system hierarchy. The model captures the codes' system, subsystem, and component functions, inputs/outputs, and code (structural) entities that enable the computerized model's functions. Ideally, the model would capture details down to the algorithm or pseudo code level.	The model abstracts the computerized model (simulation model) by system hierarchy. The model captures the codes' system and subsystem inputs/outputs, and code (structural) entities.	The model captures the computerized model (simulation model) at the black-box model level.

Figure 150. Qualitative Evaluation Rubric for sub-criteria/measures 12–15 of 18 total.

1. Criteria 12. Evaluation of the CRIM MBSE Model to Support Defining the M&S System (problem) entity.

2. SCORE: Good

It should be clarified that the M&S System (problem) entity is *not* CRIM, it is the Fleet Operations and Maintenance (O&M) system. This research focused on modeling CRIM which is a quantitative analytical model of the M&S system entity. Modeling the M&S system entity was out of scope for this effort. However, to score "excellent" the M&S system should be modeled as well. The common interface or element between the M&S system entity and the CRIM system is the data sources. The same (or more) data sources would be modeled if someone developed an MBSE model of the Fleet O&S system. The CRIM MBSE model does capture the data sources and to a limited degree the data fields of these data sources, not presented in this thesis's results, but they exist in the underlying CRIM MBSE model.

3. Criteria 13. Evaluation of the CRIM MBSE Model to Support Defining the M&S Conceptual Model

4. SCORE: Good

The CRIM MBSE model captures the high-level Conceptual model entities for both the Driver Tree and the SIMLOX models that are part of CRIM. The MagicGrid v.1 Framework (W3) Logical Subsystem Communication Structure is used to capture CRIM's conceptual models. The CRIM MBSE model identifies the Driver Tree model as a type of reliability tree conceptual model, and the CRIM MBSE model identifies the SIMLOX model as a type of multi-echelon, multi-indenture discrete event Monte Carlo simulation conceptual model. These descriptions provide the high-level identity for these types of model concepts. Internal details for the two types of conceptual models were not modeled. Instead, the author pasted artifacts, such as the Driver Tree mathematical relationship model that shows the reliability tree model structure and mathematical relationships, and the author pasted artifacts that illustrate working details of the multi-echelon, multi-indenture model.

5. Criteria 14. Evaluation of the CRIM MBSE Model to Support Defining M&S Data Validity

6. SCORE: Good

The model identifies data sources, but it does not address data validity parameters for each data source and for each individual data source field. Data sources individual fields were modeled but are not presented in this thesis' results. These model elements will be useful when modeling more concrete CRIM data flow details.

7. Criteria 15. Evaluation of the CRIM MBSE Model to Support Defining the M&S Computerized Model

8. SCORE: Good

The CRIM MBSE model captures the main computer model elements, but generally the details for the computer model elements are represented at the black-box level. The abstraction of the computer model elements in the MBSE model still provides users a way to isolate and identify the uniqueness of the computer model elements.

For the MBSE model to score excellent, the MBSE model would need to capture the computerized model "code-to" specifications.

D. RESULTS EVALUATION: CRITERIA 16 TO 18

This set of criteria supports answering the parent, model effectiveness question of:

> Do M&S model elements reference or trace to Sargent's (2013) M&S VV&A process framework M&S V&V Processes?

The M&S V&V processes identified in Sargent's (2013) circle, are the: conceptual model validation, operational validation, and computerized model verification.

Effectiveness Criteria	Effectiveness Sub-criteria/Measure	Sub-criteria/Measure	Excellent	Good	Poor or Needs Work
Do M&S model elements reference or trace to the Sargent's (2013) M&S VV&A process framework M&S V&V Processes?	Conceptual Model Validation	16	The model captures conceptual model validation methods that are suitable for system level validation, such as face-validation.	The model does not capture conceptual model validation methods, but defines and structures system boundaries/partitions, or identifiable M&S elements that can be assigned a conceptual model validation method.	The model does not capture conceptual model validation methods, and the model's level of detail does not sufficiently map out the M&S system boundaries/partitions or entities that would be associated with the conceptual model and its validation.
	Operational (results) Validation	17	The model captures operational (result) validation methods that are suitable for the respecting M&S entities at the system, subsystem, and component level.	The model captures operational (result) validation methods at the M&S system level.	The model does not capture operation (results) validation methods, but defines the M&S system architecture so that SMEs can identify and assign operational validation methods.
	Computerized Model Verification	18	The model captures code development best-practices and verification methods that apply to some M&S system function or component.	The model captures code verification methods that apply to their respective M&S system functions or entities.	The model does not capture computerized model verification information.

Figure 151. Qualitative Evaluation Rubric for sub-criteria/measures 16–18 of 18 total.

1. Criteria 16. Evaluation of the CRIM MBSE Model to support M&S Conceptual Model Validation.

2. SCORE: Excellent

The MBSE model captures system and subsystem details of CRIM's conceptual models: the Driver Tree reliability simulation conceptual model, and the SIMLOX multi-echelon, multi-indenture discrete event Monte Carlo simulation conceptual model. Some component level details are also captured, but not in enough depth to fully understand which real-system physical processes are encoded into the conceptual model component's functions. However, the CRIM MBSE model provides enough details to get notion of the working principles of the conceptual models.

Face validation is a conceptual model validation method that is supported by this MBSE model. In *face validation*, "experts on the problem entity (e.g., the system) evaluate the [graphical] model [which is a representation of the conceptual model] to determine if they believe it [the conceptual model] is reasonable and correct for its intended purpose" (Sargent 1986, 239). In this case, the CRIM MBSE model is the graphical model.

3. Criteria 17. Evaluation of the CRIM MBSE Model to support M&S Operational Validation.

4. SCORE: Good

The CRIM MBSE model defines system boundaries and data flows where operational (result) validation methods can be applied. The model identifies data flow outputs of the Driver Tree and SIMLOX simulation models. The simulation results are abstracted and captured by these data flows. The author added comments to these simulation result data flows stating that operational validation methods should be applied there. The author proposed that statistical tests can be applied to these simulation results. Additionally, the author identified in the model that sensitivity analysis can be performed at the system level by varying the simulation model inputs.

The model also captures component level validation methods that can be applied. Component functions are in the functional value pathway that deliver overall system level functionality and value, therefore should also be validated and verified as appropriate. For example, the author recommends that validation methods be applied to Honeybee results. The author proposes in the model that statistical methods for validation the Honeybee generated statistical distributions for "component-level" metrics should be applied and that Turin validation tests may be applied to Honeybee generated flight profiles.

5. Criteria 18. Evaluation of the CRIM MBSE Model to support M&S Computerized Model Verification

6. SCORE: Poor or Needs Work

The model does not capture computerized model verification information. However, this is not a limitation of the model. If enough information about the computerized model code specifications is known, these specifications can be captured in

the model can be used as the authoritative source of truth for code specifications. Additionally, verification test model elements can be created and traced to the code specification model elements so that all of the computerized model verification information may be kept in the MBSE model.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

This chapter summarizes the results analysis and provides recommendations that can be used to improve the MBSE model developed for this research. This chapter presents conclusions that are supported by the results analysis and interpretation.

A. RESULTS AND RECOMMENDATIONS

1. Results Summary

The methodology produced an MBSE model of the Cost Readiness Impact Model (CRIM). The MBSE model is primarily a graphical, descriptive model. Descriptive models are mostly qualitative in nature; therefore, a qualitative analysis was appropriate for evaluating the obtained results. The author developed a rubric that was derived from two main effectiveness questions. The rubric was used to evaluate the effectiveness of resulting product, the MBSE model, and by association also validates this research's methodology used to produce the MBSE model—if it is a good product, it is a good method:

- General Question: Can this method be applied (re-used) to any M&S system and provide a useful M&S VV&A framework that identifies the M&S system entities and corresponding VV&A elements?

- Specific Question: Does the resulting MBSE model provide an ontological mapping or framework of the M&S system (CRIM) that structures and defines M&S entities in a manner that is helpful for V&V planning and for precisely tracing or assigning V&V methods or actions?

These questions are derived themselves from the primary thesis question: Can Model-Based Systems Engineering (MBSE) be leveraged to develop a re-usable framework and process for Validation Verification and Accreditation (VV&A) of Modeling and Simulation (M&S) models?

The rubric takes the "specific" effectiveness question and adds three more accompanying effectiveness questions. This set of four effectiveness questions captures the entire set of rubric criteria. The four effectiveness questions are presented below:

1. Does the model provide an (ontological) mapping or framework of the M&S system that defines and structures M&S entities in a manner that is helpful for V&V planning and for precisely tracing or assigning V&V methods or actions?
2. Does the model support development of the M&S V&V plan? According to the Documentation of Verification, Validation, and Accreditation (VV&A) for Models and Simulations, MIL-STD-3022 w/CHANGE 1 (DOD 2012)
3. Do M&S model elements reference or trace to the M&S entities identified by Sargent's (2013) M&S VV&A process framework?
4. Do M&S model elements reference or trace to Sargent's (2013) M&S VV&A process framework M&S V&V Processes?

The author went through each criterion in the rubric and provided an evaluation of the MBSE model by providing a qualitative score. The score is based on the qualitative features defined in the rubric for a particular criterion. Figure 152 shows a visual distribution of the scores that were given for each criterion.

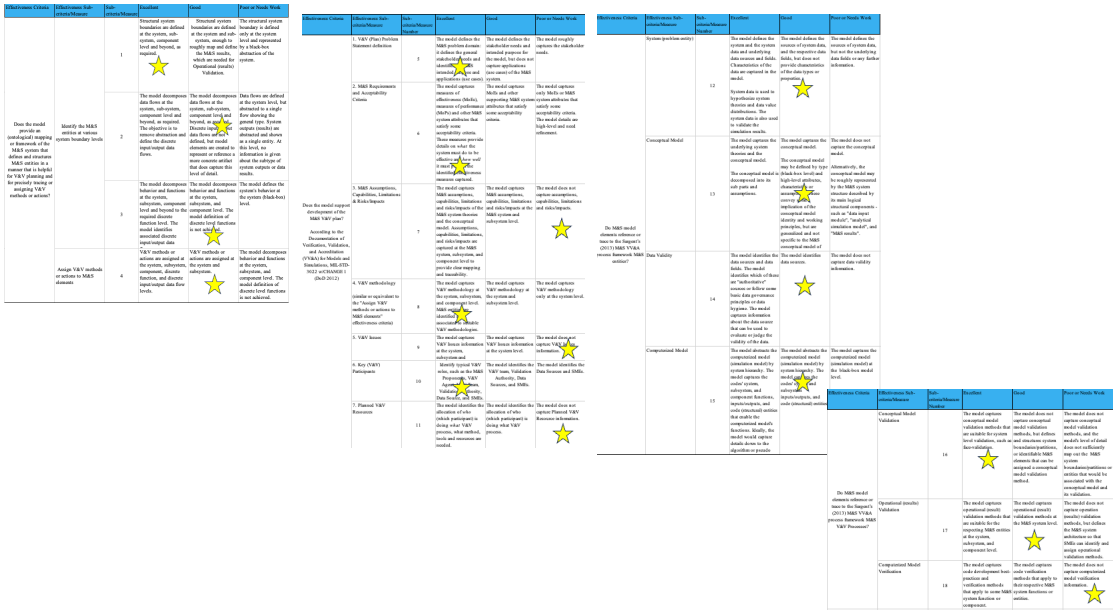


Figure 152. Overall Rubric score distribution for the CRIM MBSE model. The overall score distribution gives a rough idea of how effective the MBSE model may be for supporting VV&A efforts or deriving VV&A products.

A simple tally reveals the following counts:

1. Criteria rated as “Excellent”: 6 counts.
2. Criteria rated as “Good”: 8 counts.
3. Criteria rated as “Poor or Needs Work”: 4 counts.

It is best to evaluate the MBSE model at the individual criterion score level. This rubric was not designed to provide some sort of results aggregation that yield a quantitative effectiveness value. However, this rubric does seek to remove subjectivity from the qualitative assessment of the CRIM MBSE model. Overall, the author believes that the methodology used in this thesis is effective in producing an MBSE model of the M&S system (CRIM), and that resulting model is effective for answering and supporting M&S VV&A questions and efforts. The author believes that the method used here can be widely applicable to any M&S system, and is thus “re-usable,” as stated in the primary thesis question. Additionally, the author believes that the evaluation rubric developed for this

research can also be applied to other M&S models of various types, not just MBSE models, and thus, is also "re-usable."

The limiting factor for why the MBSE model did not score as "excellent" on all criteria was not the fault of the modeling tool or method, but rather the fault of the author's limited knowledge of the Cost Readiness Impact Model system. It is possible for the author or any other modeler to model to the degree where the MBSE model could score "excellent" in every defined criterion.

2. Recommendations

3. Method recommendation

The author recommends the method used in this research to those that must perform M&S VV&A. The author also recommends this method when designing an M&S product or system. Users can apply this method to build their MBSE model of their M&S system while the M&S system is being designed. Once the M&S product or system is implanted, the user should also have a finished MBSE model that can be used to support VV&A efforts. Other users may opt to choose their own MBSE framework or modeling methodology. The author believes that if good systems engineering and systems architecture principles are followed, the resulting M&S descriptive model will be useful in supporting VV&A.

4. Evaluation Recommendation

The author recommends that others use the evaluation rubric developed for this research or that others design their own based on their defined effectiveness criteria. The effectives criteria used in this rubric should be broad enough for re-use, but it still addresses essential M&S VV&A elements and attributes. If it is not used as-is, it can provide a good foundation for modification.

B. CONCLUSIONS

The primary and secondary research questions are stated below so that the conclusions can be presented in the context of the research questions:

The primary research question is:

- Can Model-Based Systems Engineering (MBSE) be leveraged to develop a re-usable framework and process for Validation Verification and Accreditation (VV&A) of Modeling and Simulation (M&S) models?

The secondary research question is:

- If a new M&S model is being conceptualized or developed, can the framework developed by this thesis help guide and facilitate the design and development of M&S architecture and system requirements?

Considering the overall research accomplishments, the author believes that the simple answer to both questions is "yes." However, it is important to clarify what "framework and process" means. Maybe it is necessary to refine the primary research question in light of the research and the results. This thesis research presents an MBSE methodology, which uses a modeling framework (MagicGrid) to produce an M&S MBSE model. The M&S MBSE model has a framework that structures defined M&S V&V methods and their allocations to M&S entities. The M&S MBSE model can be used to support M&S VV&A by providing information that can be used to derive the V&V plan, which outlines the V&V processes and methods that will be applied to the M&S product or system. Therefore, the "framework and process" means two things: (1) The framework and process that are part of this thesis research methodology, and (2) the M&S VV&A framework and process that is derived from the M&S MBSE model produced by the methodology. These concepts are illustrated in Figure 153. Perhaps the primary research question should be refined to reflect the direct context of the research work and findings. In retrospect of the original question, the primary thesis question could be refined and restated as such:

- Can an MBSE model of the M&S product be useful for supporting that product's VV&A efforts?

VV&A efforts would include the derived processes and products. The secondary research question can be left alone—the essence of it is, "can MBSE be used to help design

the M&S product/system such that the model supports M&S systems architecting, definition of requirements, and definition of VV&A elements and relationships to the M&S product/system such that these will be useful when VV&A of the M&S product happens.

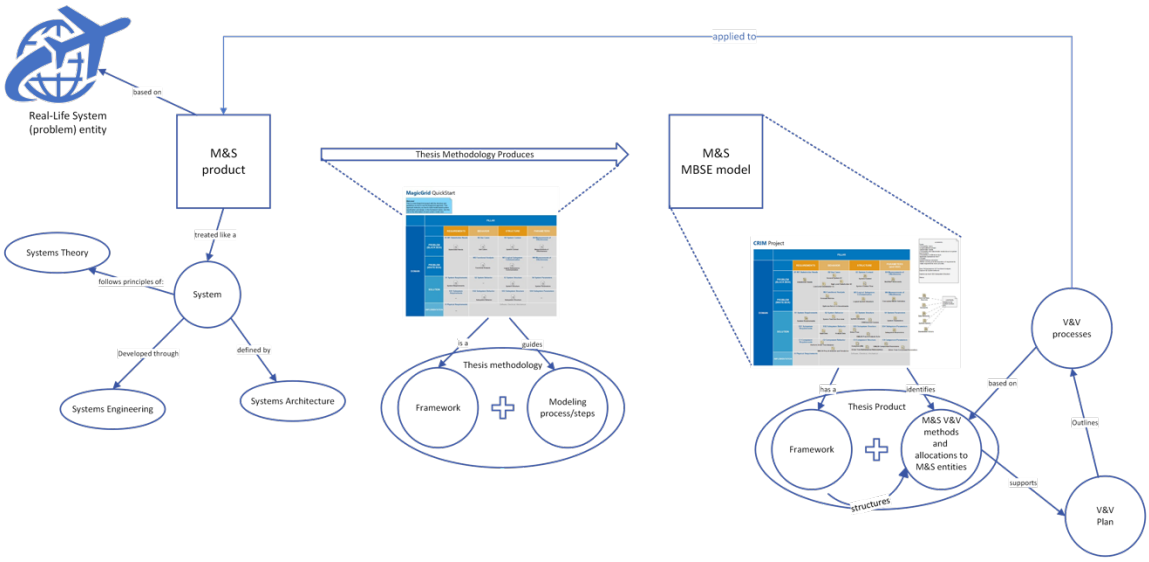


Figure 153. Thesis research question conceptual model

This thesis methodology is not limited to the subject of interest, CRIM. It can be applied to any M&S product, therefore it is "re-usable." The MagicGrid framework is useful for defining the M&S product in therms of systems architecture. The systems architecture defines system boundaries and inputs and outputs. Therefore, the M&S product can be described in a lot of detail. The MagicGrid framework supports capturing other useful infromation that helps define VV&A elements and V&V methods and their relations to the M&S system elements. The reulting product is a model that can be used to support M&S VV&A.

This method can be used to reverse engineering an M&S product, as was demonstrated in this thesis, thus addressing the primary research question, or it can be used to forward engineer an M&S product, thus addressing the secondary research question.

The limitations the product that can be produced are not limitations of the method or the model, but rather limitations of the modeler's ability to gather the necessary information to model to the degree necessary to support M&S VV&A efforts.

1. Future Research

Chapter III has a "Future Work" section at the end of each modeling task. The author recommends that the items identified in these sections be carried out. In general, the author believes that further refinement of the model is needed to describe all of the discrete inputs, outputs, and functions of the M&S system.

Future research or work should be done on the development of a CRIM V&V plan derived from this CRIM MBSE model. The CRIM V&V plan will be very useful for CRIM stakeholders and NAVAIR overall.

Future research should be done to develop a descriptive model of the real-life system (problem) entity, i.e., the Fleet Operations and Maintenance (O&M) system. This would be a valuable accompanying product that can be used as an authoritative source of truth that defines the fleet's O&M entities, relationships, and physical processes. There is not a single place or product that captures all this information (to the author's knowledge). Such a model can help evaluate existing M&S products based on the fleet or this model can support development of new M&S produces.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

This appendix contains slightly larger images than some of the ones shown in the main body of this document. Readers can zoom in on these images to see smaller details.

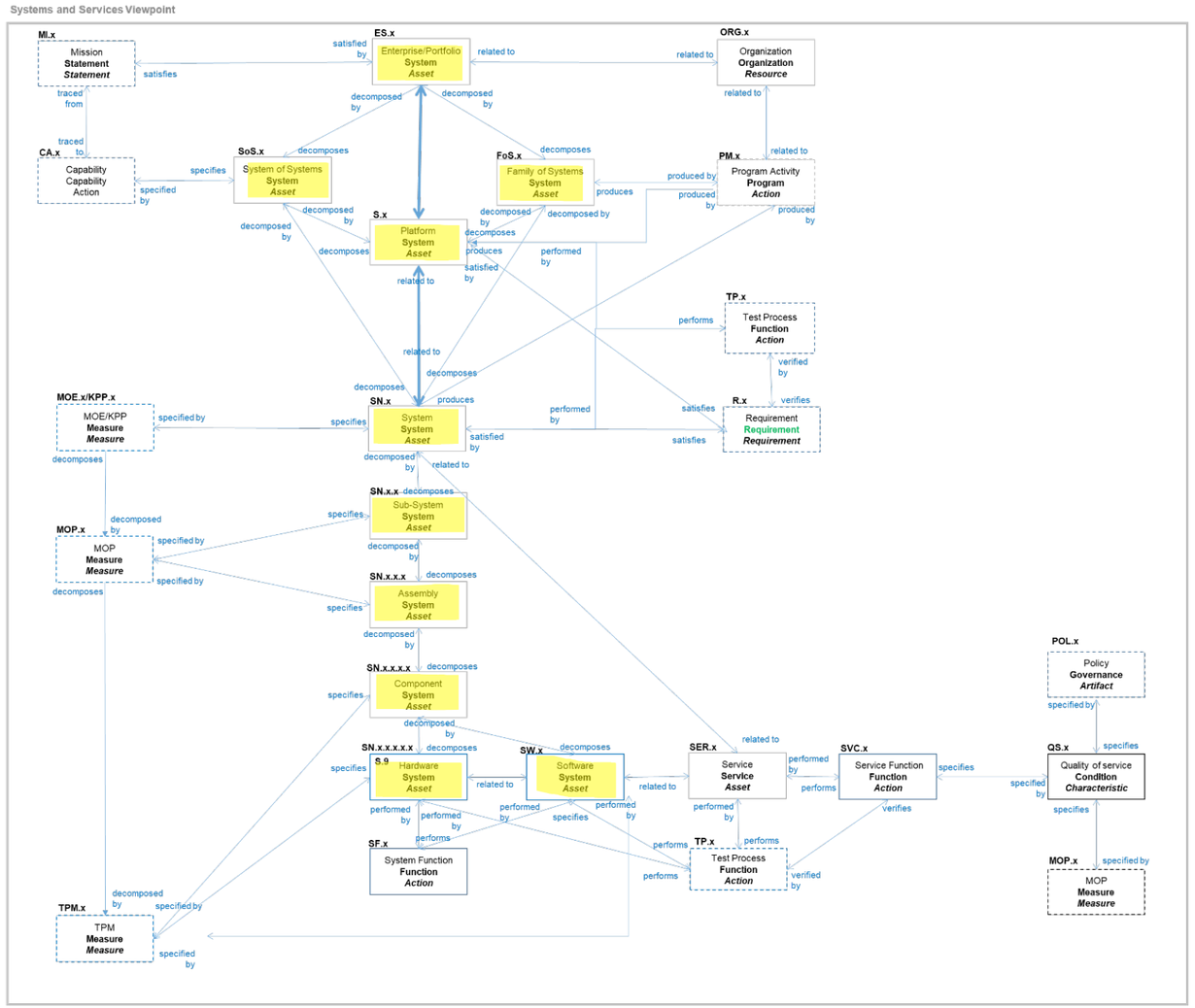


Figure 154. Systems and Services Viewpoint from the Marine Corp Systems Command (MCSC) Ontology and Conceptual Data Model (CDM). Zoom in to see smaller details. This image is a slightly larger version of Figure 4. Source: Vaneman et al. (2022,41).

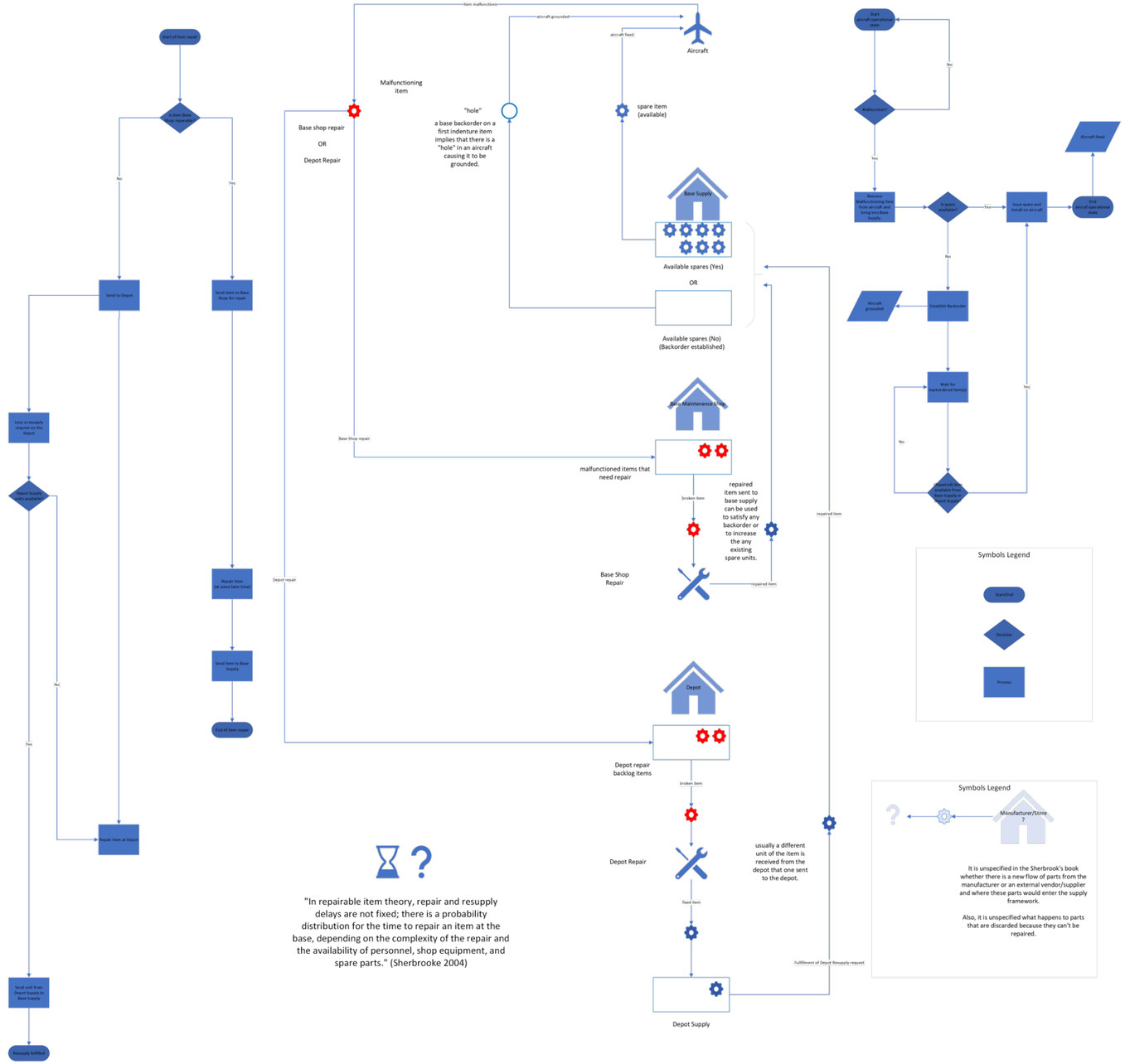


Figure 155. "Physics" of the Problem: Diagram synthesized and translated from Sherbrooke's (2004) textual description. Zoom in to see smaller details. This image is a slightly larger version of Figure 78.

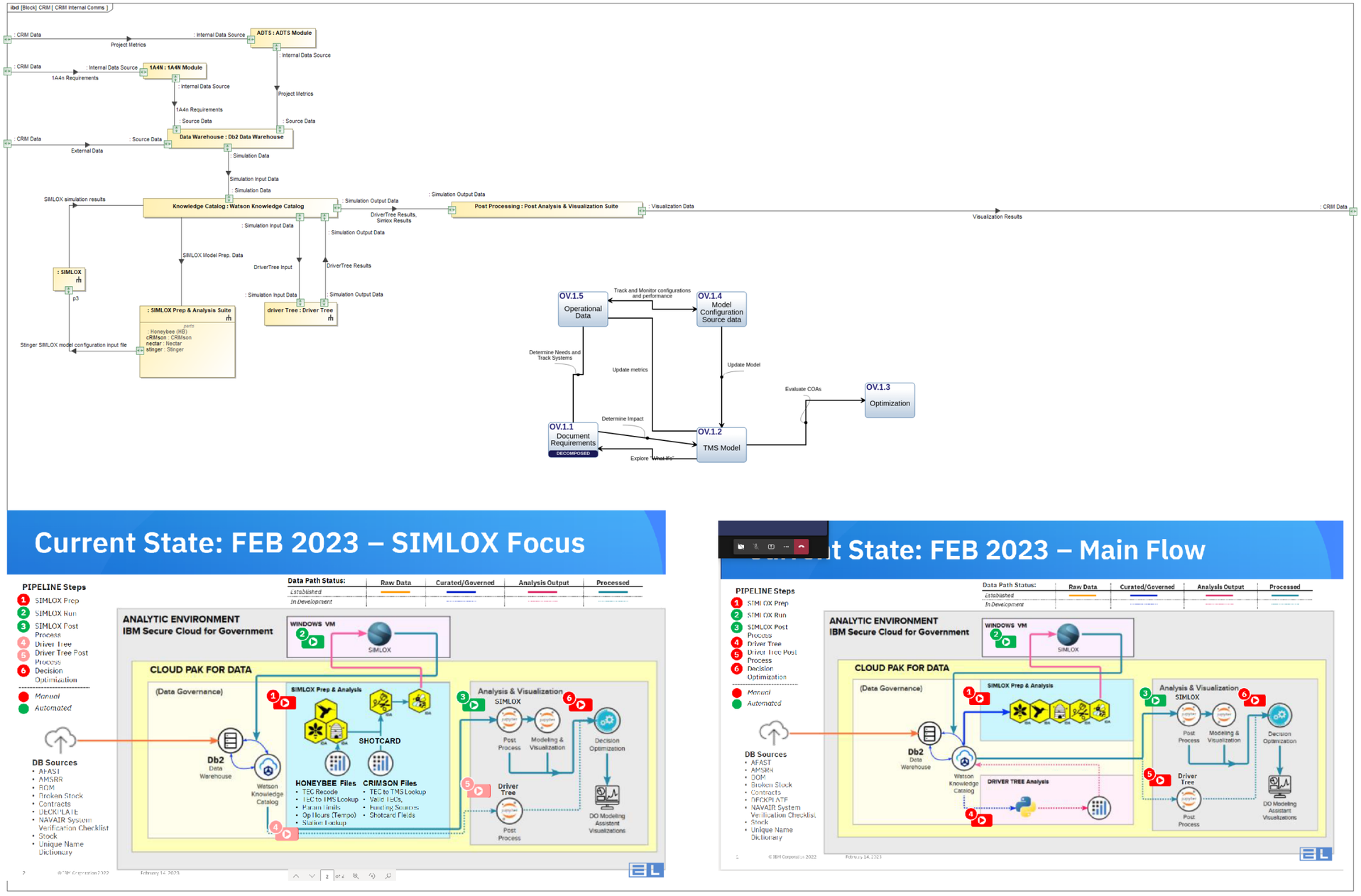


Figure 156. CRIM's internal block diagram with pasted artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 113.

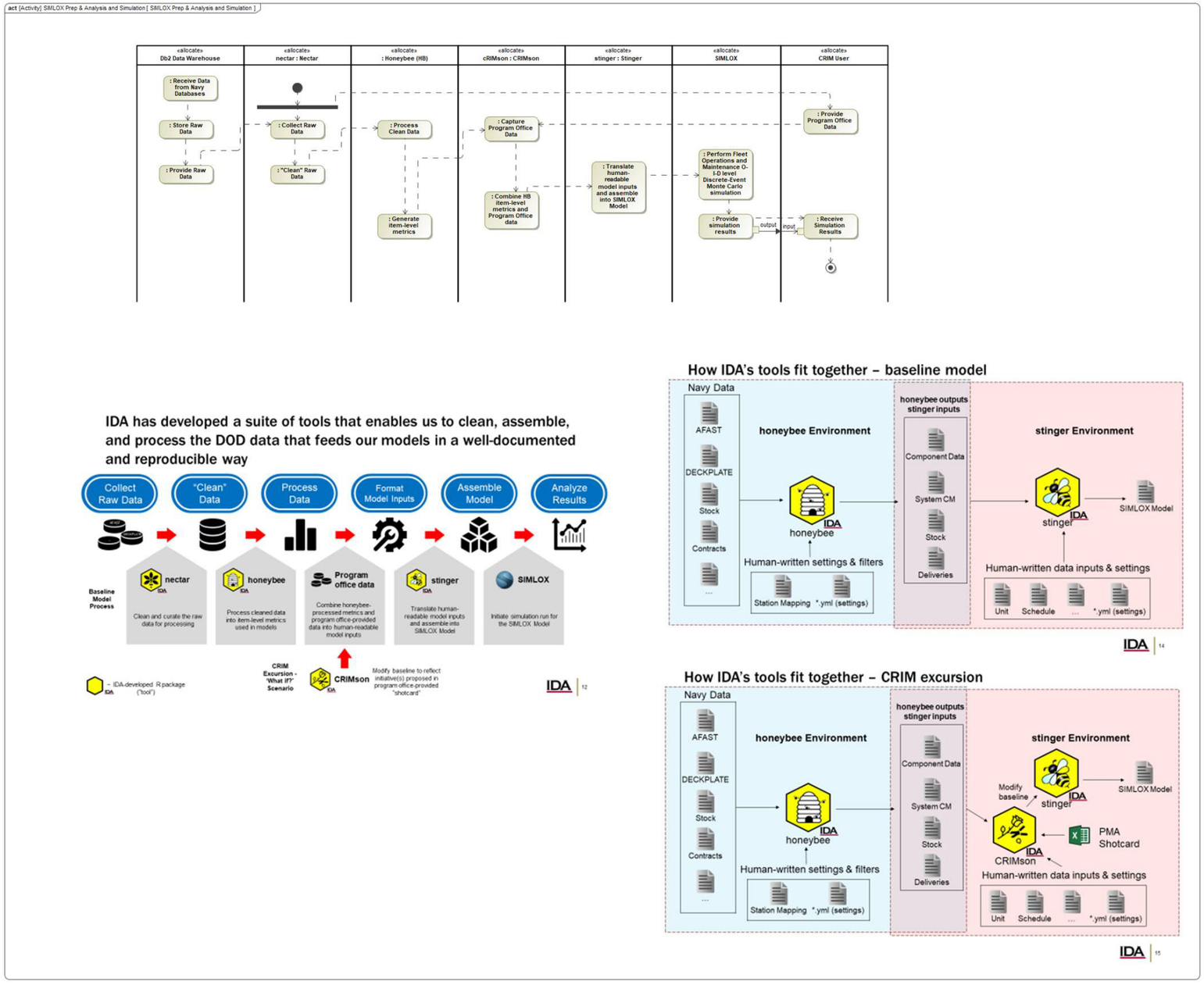
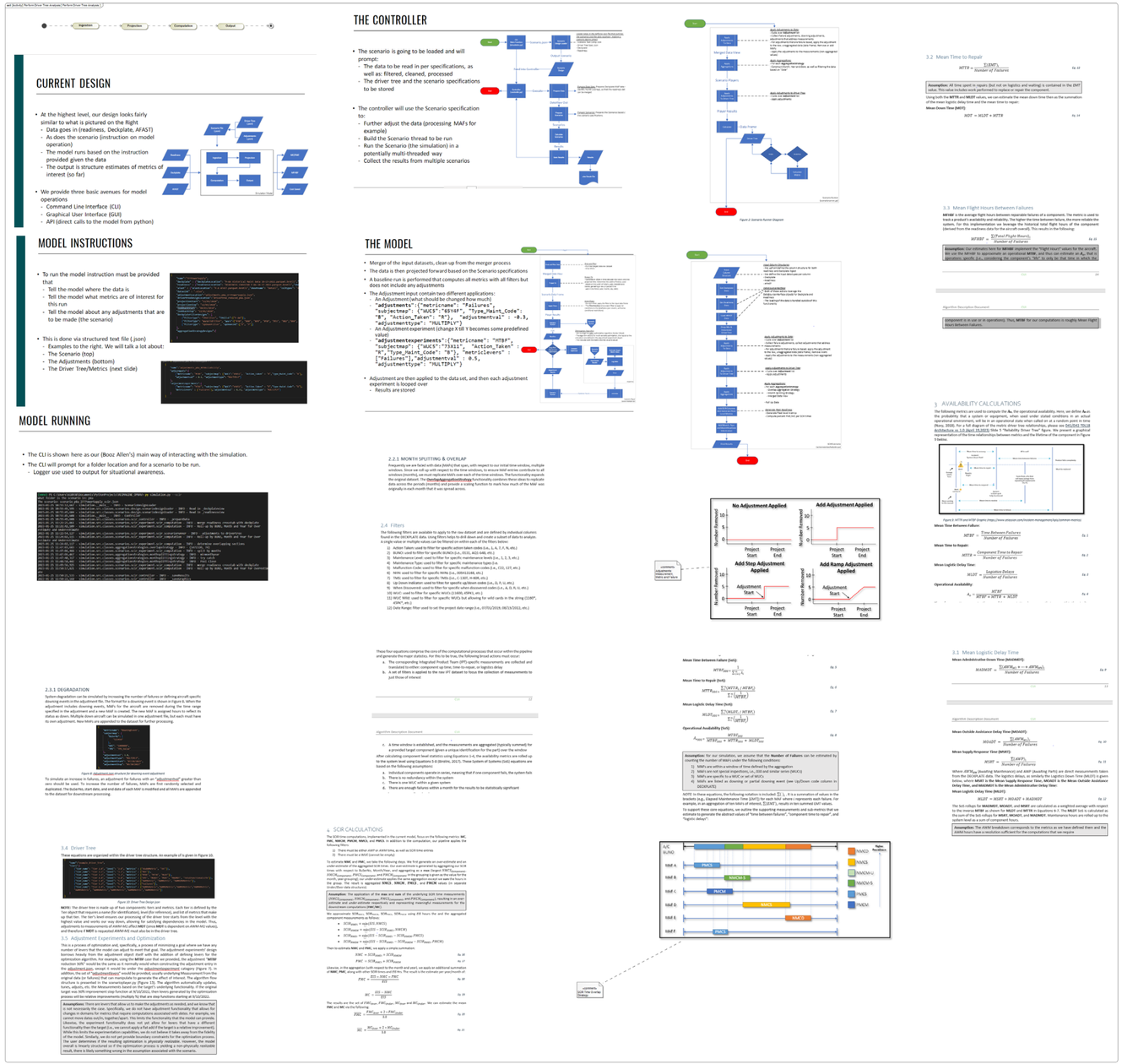
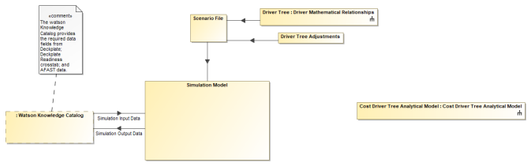


Figure 157. SIMLOX Prep & Analysis and Simulation activity diagram with pasted reference artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 124.


ingestion
Projection
Computations
Output
CURRENT DESIGN
At the highest level, our design looks fairly similar to what is pictured on the Right
Data goes in (readiness, Deckplate, AFAST)
As does the scenario (instruction on model operation)
The model runs based on the instruction provided given the data
The output is structure estimates of metrics of interest (so far)
We provide three basic avenues for model operations
Command Line Interface (CLI)
Graphical User Interface (GUI)
API (direct calls to the model from python)
Readiness
Aircraft
ADAF
Scenario
Projection
Analytics
Output
MCFHR
MFHR
Cost Center
MODEL INSTRUCTIONS
To run the model instruction must be provided that
Tell the model where the data is
Tell the model what metrics are of interest for this run
Tell the model about any adjustments that are to be made (the scenario)
This is done via structured text file (.json)
Examples to the right. We will talk a lot about:
The Scenario (top)
The Adjustments (bottom)
The Driver Tree/Metrics (next slide)
MODEL RUNNING
The CLI is shown here as our (Booz Allen's) main way of interacting with the simulation.
The CLI will prompt for a folder location and for a scenario to be run.
Logger use used to output for situational awareness.
THE CONTROLLER
The scenario is going to be loaded and will prompt:
The data to be read in per specifications, as well as filtered, cleaned, processed
The driver tree and the scenario specifications to be stored
The controller will use the Scenario specification to:
Further adjust the data (processing MAFs for example)
Build the Scenario thread to be run
Run the Scenario (the simulation) in a potentially multi-threaded way
Collect the results from multiple scenarios
THE MODEL
Merger of the input datasets, clean up from the merger process
The data is then projected forward based on the Scenario specifications
A baseline run is performed that computes all metrics with all filters but does not include any adjustments
The Adjustment input contain two different applications:
An Adjustment (what should be changed how much)
"adjustments":{"metricname": "failures", "sub:ectmq": "A6C3","6S*##", "Type_Maint_Code": "B", "Action_Taken": "R"}, "adjustmentval" : 0.3, "adjustmenttype": "MULTIPLY"}
An adjustment experiment (change X till Y becomes some predefined value)
"adjustmentexperiments":{"metricname": "MTB#", "sub:ectmq": ["A6C5","7731t", "Action_Taken": "R","Type_Maint_Code": "B"], "metriclevers" : ["failures"],"adjustmentval" : 0.5, "adjustmenttype": "MULTIPLY"}
Adjustment are then applied to the data set, and then each adjustment experiment is looped over
Results are stored
2.2.1 MONTH SPLITTING & OVERLAP
Frequently we are faced with data (mainly that type, with respect to our initial time-window, multiple windows. Since we roll up with respect to the time windows, to ensure MAF entries contribute to all windows (months), we must replicate MAFs over each of the time windows. The functionality expands the original dataset. The OverlapAggregateMonthsMaps functionality combines these ideas to replicate data across the periods (months), and provide a saving function to mark how much of the MAF was originally in each month that it was spread across.
2.4 Filters
The following filters are available to apply to the raw dataset and are defined by individual columns found in the DECKPLATE data. Using filters helps to drill down and create a subset of data to analyze. A single column can have multiple filters applied to it. Available filters are as follows:
1) Action Taken: used to filter for specific action taken codes (i.e., D, L, T, N, etc.)
2) BUNO: used to filter for specific BUNOs (i.e., 03151, 4CD 648, etc.)
3) Maintenance Level: used to filter for specific maintenance levels (i.e., 1, 2, 3, or 4.)
4) Maintenance Type: used to filter for specific maintenance types (i.e.
5) Malfunction Code: used to filter for specific malfunction codes (i.e., 172, 137, etc.)
6) NHA: used to filter for specific NHAs (i.e., 0004121048, etc.)
7) Type Maint Code: used to filter for specific maintenance codes (i.e.
8) Up Down Indicator: used to filter for specific up/down codes (i.e., D, U, etc.)
9) When Discovered: used to filter for specific when discovered codes (i.e., A, D, N, U, etc.)
10) WUC: used to filter for specific WUCs (12400, 45041, etc.)
11) WUC MSN: used to filter for specific WUCs but allowing for wild cards in the string (1180*, 4104*, etc.)
12) Date Range: filter used to set the project date range (i.e., 07/01/2019, 09/15/2022, etc.)
These four equations comprise the core of the computational processes that occur within the pipeline and generate the major statistics. For this to be true, the following broad actions must occur:
a. The corresponding Deckplate Product Tuple (PT) specific measurements are collected and translated to either component up time, time-to-repair, or logistics delay
b. A set of filters is applied to the raw PT dataset to focus the collection of measurements to just those of interest
c. A time window is established, and the measurements are aggregated (typically summed) for a provided target component (given a unique identification for the part) over the window
After calculating component level statistics using Equations 1-4, the availability metrics are rolled up to the system level using Equations 5-8 (Jenkins, 2017). These System of Systems (SoS) equations are built on the following assumptions:
a. Individual components operate in series, meaning that if one component fails, the system fails
b. There is no redundancy within the system
c. There is one MUC within a given system
d. There are enough failures within a month for the results to be statistically significant
3.4 Driver Tree
These equations are organized within the driver tree structure. An example of is given in Figure 10.
Figure 10 Driver Tree Design.json
NOTE: The driver tree is made up of two components: Kees and metrics. Each tier is defined by the Tier object that requires a name (for identification), level (for reference), and list of metrics that make up that tier. The Tier's level ensures our processing of the driver tree starts from the level with the lowest value and works its way up to the highest level. The metrics themselves are the nodes in the model. Thus, adjustments to measurements of AWM-A/G (since MAF) (since MAF) is dependent on AWM-A/G values), and therefore if MAF is requested AWM-A/G must also be in the driver tree.
3.5 Adjustment Experiments and Optimization
This is a process of optimization and, specifically, a process of minimizing a goal where we have any number of levers that the model can adjust to meet that goal. The adjustment experiments' design borrows heavily from the adjustment object itself with the addition of defining levers for the optimization algorithm. For example, using the MFHR case that we provided, the adjustment "MFHR reduction 50%" would be the same as it normally would when constructing the adjustment entry in the adjustments.json file (see Figure 11). The adjustment experiment, however, would look like: In addition, the set of "adjustmentlevers" would be provided, usually containing Measurement from the original data (or failures) that can manipulate to generate the effect of interest. The algorithm flow structure is presented in the scenarioRunner.py (Figure 12). The algorithm automatically updates, tunes, adjusts, etc. the Measurements based on the target's underlying functionality. If the original MFHR is 0.5 hours and the adjustment is to reduce it by 50%, then the optimization and adjustment process will be relative improvements (multiply by 1%) that are step functions starting at 0.5(1)(1)(1)...
Assumptions: There are levers that allow us to make the adjustments as needed, and we know that it will monotonically the case. Specifically, we do not have adjustment functionality that allows for changes in domains for metrics that require computations associated with dates. For example, we cannot move dates (each, together/apart. This limits the functionality that the model can provide. Likewise, the experiment functionality does not yet allow for levers that have a different adjustment type (multiply, add, etc.) associated with each lever. This is a relatively easy change. While this limits the experimentation capabilities, we do not believe it takes away from the fidelity of the model. Similarly, we do not yet provide boundary constraints for the optimization process. The user determines if the resulting optimization is physically realizable. However, the model overall is linearly structured in if the optimization process is yielding a non-physically realizable result, there is likely something wrong in the assumption associated with the scenario.
4. SCR CALCULATIONS
The SCR time computations, implemented in the current model, focus on the following metrics: MC, FMC, NMCM, PMCM, NMCS, and PMCS. In addition to the computation, our pipeline applies the following filters:
(1) There must be either AWP or AWM time, as well as SCR time entries
(2) There must be a MUC (cannot be empty)
To estimate NMC and FMC, we take the following steps: We first generate an over-estimate and an under-estimate of the aggregated SCR times. Our overestimate is generated by aggregating our SCR times with respect to ByAerNo, Month/Year, and aggregating as a max (largest SCR[Sustainment Time]). Our underestimate is generated by aggregating with respect to ByAerNo, Month/Year, and month_year grouping. Our underestimate applies the same aggregation except we seek the least in the group. The result is aggregated NMCS, NMCM, PMCS, and PMCM values (in separate scalars/floor data structures).
Assumption: The application of the max and sum of the underlying SCR time measurements (SCR[Sustainment Time]) to estimate NMC and FMC, respectively, is a reasonable estimate of AWM/AG and AWP/AVIONICS time-to-repair (TTR) aggregate measurements for the downstream computations (FMC/MC).
We exponentiate SCR_{max}, SCR_{min}, SCR_{mean}, and SCR_{std} using 10 hours the and the aggregated component measurements as follows:
$SCR'_{max} = \exp(\ln(10) \cdot SCR_{max})$
$SCR'_{min} = \exp(\ln(10) \cdot SCR_{min})$, $SCR'_{min} \leq SCR'_{max}$
$SCR'_{mean} = \exp(\ln(10) \cdot SCR_{mean} - \frac{\ln(10) \cdot SCR_{std}}{2})$, $SCR'_{mean} \leq SCR'_{max}$
$SCR'_{std} = \exp(\ln(10) \cdot SCR_{mean} - \ln(10) \cdot SCR_{mean} - \ln(10) \cdot SCR_{std})$, $SCR'_{std} \leq SCR'_{max}$
Then to estimate NMC and FMC, we apply a simple summation:
$NMC = SCR'_{max} + SCR'_{min}$ Eq 18
$FMC = SCR'_{max} + SCR'_{mean}$ Eq 19
Likewise, in the aggregation (with respect to the month and year), we apply an additional summation of NMC, FMC, along with other SCR times and ER hrs. The result is the estimate per year/month of:
$FMC' = \frac{SSS - NMC' \cdot PMC'}{SS}$ Eq 18
$MC' = \frac{SSS - NMC'}{SS}$ Eq 19
The results are the set of FMC_{under}, FMC_{over}, MC_{over} and MC_{under}. We can estimate the mean FMC and MC via the following:
$\overline{FMC} = \frac{FMC_{over} + FMC_{under}}{2}$ Eq 20
$\overline{MC} = \frac{MC_{over} + MC_{under}}{2}$ Eq 21
3.2 Mean Time to Repair
$MTTR = \frac{SUMT_{ct}}{Number\ of\ Failures}$ Eq 13
Assumption: All time spent in repairs (but not on logistics and waiting) is contained in the ENT value. This value includes work performed to replace or repair the component.
Using both the MTTR and MDT values, we can estimate the mean down time then as the summation of the mean logistic delay time and the mean time to repair:
Mean Down Time (MDT)
$MDT = MLDT + MTTR$ Eq 14
3.3 Mean Flight Hours Between Failures
MFHBF is the average flight hours between reparable failures of a component. The metric is used to track a product's availability and reliability. The higher the time between failure, the more reliable the system. For this implementation we leverage the historical total flight hours of the component (derived from the readiness data for the aircraft overall). This results in the following:
$MFHBF = \frac{SUM\ of\ Flight\ Hours_{ct}}{Number\ of\ Failures}$ Eq 15
Assumption: Our estimation here for MFHBF (registered the "Flight Hours" values for the aircraft. We use the MFHBF to approximate an operational MTBF, and thus can estimate an A_o that is operations specific (e.g., considering the component's "life" to only be that time in which the
Algorithm Description Document
component is in use vs in operations). Thus, MFHBF for our computations in roughly Mean Flight Hours Between Failures.
3 AVAILABILITY CALCULATIONS
The following metrics are used to compute the A_o the operational availability. Here, we define A_o as the probability that a system or equipment, when used under stated conditions in an actual operational environment, will be in an operational state when called on at a random point in time (Reno, 2018). For a full diagram of the metric driver tree relationships, please see DOS/DAS TDL18 Architecture vs 1.0 (April 19,2021) Slide 5 "Reliability Driver Tree" figure. We present a graphical representation of the time relationships between metrics and the lifetime of the component in figure 9 below:
Figure 9. MTTR and MTBF (Graphic) https://www.atlassian.com/incident-management/kpis/common-metrics
Mean Time Between Failure:
$MTBF = \frac{Time\ Between\ Failures}{Number\ of\ Failures}$ Eq 1
Mean Time to Repair:
$MTTR = \frac{Component\ Time\ to\ Repair}{Number\ of\ Failures}$ Eq 2
Mean Logistic Delay Time:
$MLDT = \frac{Logistics\ Delays}{Number\ of\ Failures}$ Eq 3
Operational Availability:
$A_o = \frac{MTBF}{MTBF + MTTR + MLDT}$ Eq 4
3.1 Mean Logistic Delay Time
Mean Administrative Down Time (MADMDT):
$MADMDT = \frac{\sum(ADT_{M_1} + \cdots + ADT_{M_n})}{Number\ of\ Failures}$ Eq 9
Mean Supply Response Time (MSRT):
$MSRT = \frac{\sum(AWP)}{Number\ of\ Failures}$ Eq 11
Where ADT_{M_n} (leading Maintenance) and AWP (awaiting Parts) are direct measurements taken from the DECKPLATE data. The logistics delays, as similarly the Logistics Down Time (MLDT) is given below, where MSRT is the Mean Supply Response Time, MMANDT is the Mean Outside Assistance Delay Time, and MAMDMDT is the Mean Administrative Delay Time:
Mean Logistic Delay Time (MLDT):
$MLDT = MSRT + MOADT + MADMDT$ Eq 12
The SoS rollups for MADMDT, MOADT, and MSRT are calculated as a weighted average with respect to the inverse MFBF as shown for MLDT and MTTR in Equations 6-7. The MLDT SoS is calculated as the sum of the SoS rollups for MSRT, MOADT, and MADMDT. Maintenance hours are rolled up to the system level as a sum.
Assumption: The AWM breakdown corresponds to the metrics as we have defined them and the AWM hours have a resolution sufficient for the computations that we require.
Figure 158. Perform Driver Tree Analysis activity diagram with pasted reference artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 126.



CURRENT DESIGN

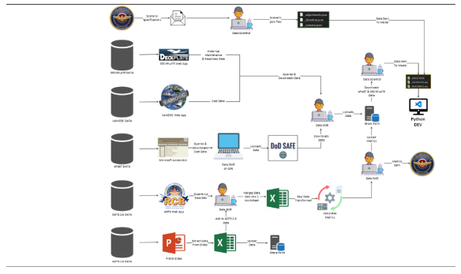
- At the highest level, our design looks fairly similar to what is pictured on the Right
  - Data goes in (readiness, Deckplate, AFAST)
  - As does the scenario (instruction on model operation)
  - The model runs based on the instruction provided given the data
  - The output is structure estimates of metrics of interest (so far)

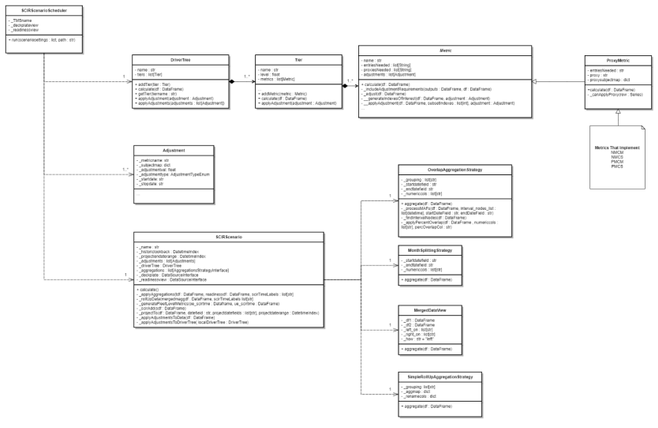
- We provide three basic avenues for model operations
  - Command Line Interface (CLI)
  - Graphical User Interface (GUI)
  - API (direct calls to the model from python)



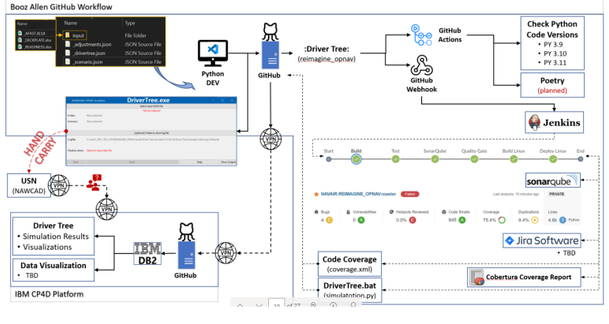


TDL 18 DATA FLOW DIAGRAM





TDL 18 CI/CD ARCHITECTURE



TDL 18 DATA FIELDS AND TABLE JOINS

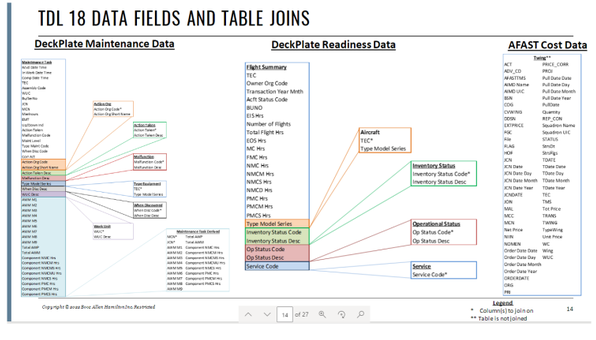


Figure 159. Driver Tree Subsystem Structure with pasted artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 129.

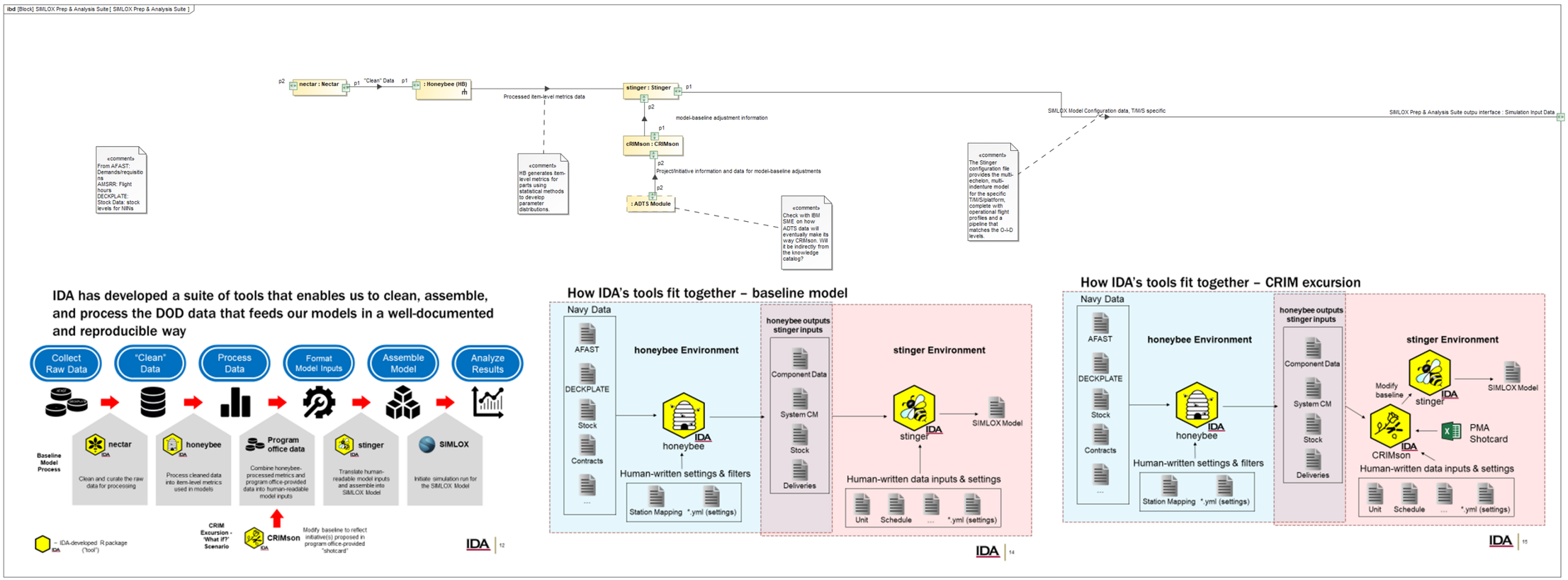


Figure 160. SIMLOX Prep. & Analysis Suite Subsystem Structure with pasted reference artifacts. Zoom in to see smaller details. This image is a slightly larger version of Figure 131.

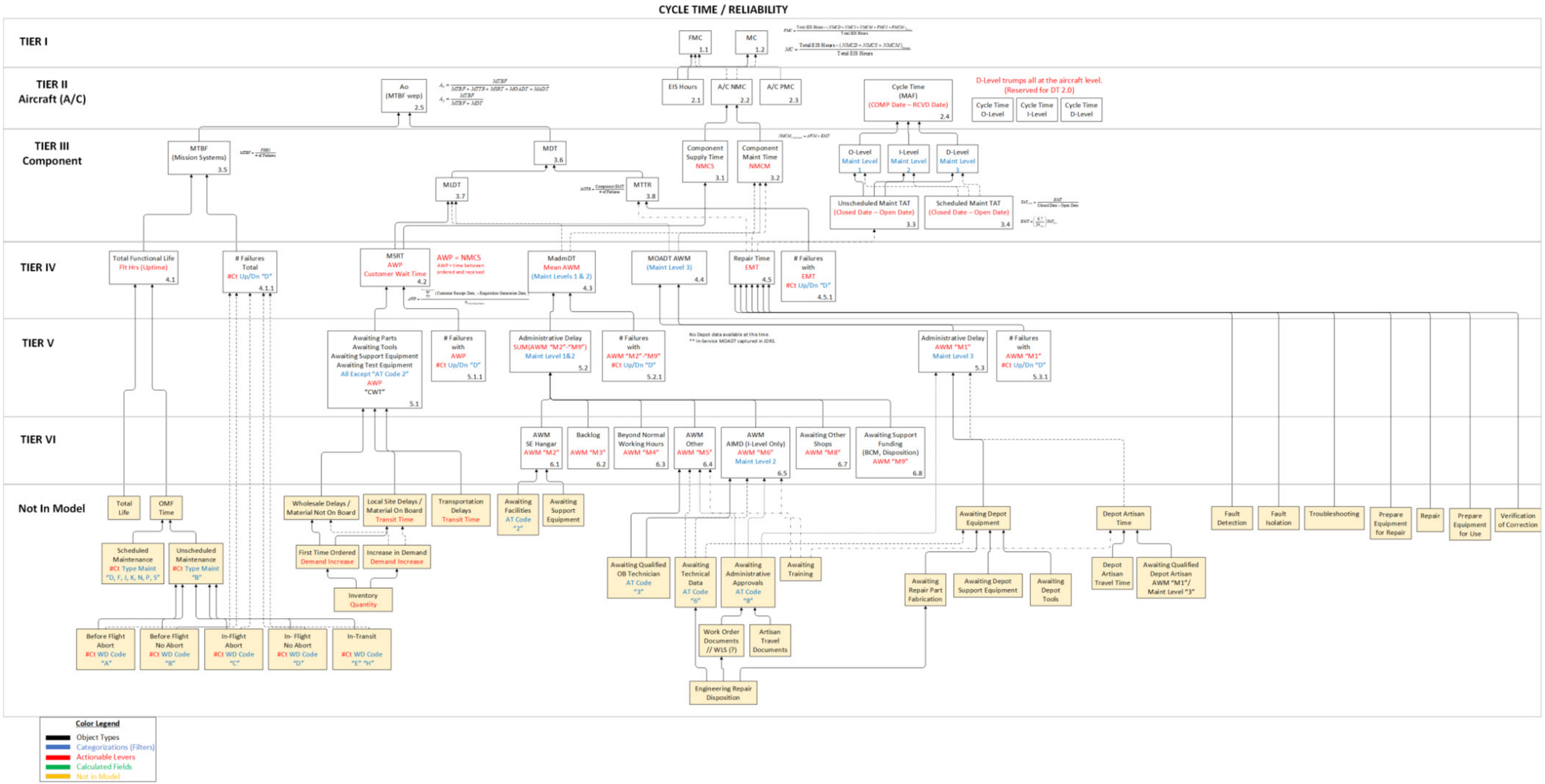


Figure 161. Driver Tree Mathematical Relationships. Zoom in to see smaller details. This image is a slightly larger, cropped version of Figure 135.

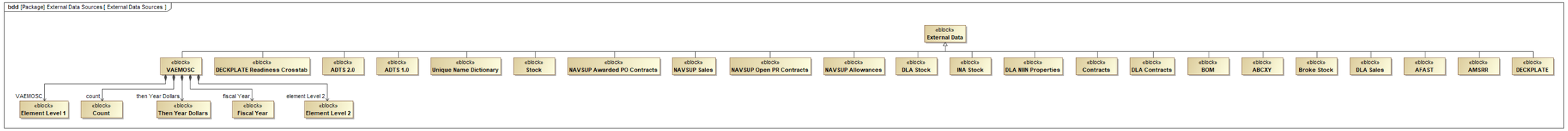


Figure 162. Navy External Data Sources view. Zoom in to see smaller details. This image is a slightly larger version of Figure 142.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

AcqNotes. 2017 “Non-Materiel Solution.” Last modified July 12, 2017. https://acqnotes.com/acqnote/acquisitions/non-materiel-solution.

———. 2021 “Systems Engineering Technical Review Process.” Last modified June 30, 2021. https://acqnotes.com/acqnote/careerfields/systems-engineering-technical-review-process.

———.2023 “Measures of Effectiveness (MOE).” Last modified August 15, 2023. https://acqnotes.com/acqnote/tasks/measures-of-effectivenessrequirements.

Aleksandraviciene, Aiste, and Aurelijus Morkevicius. 2018. *MagicGrid Book of Knowledge A Practical Guide to Systems Modeling Using MagicGrid from No Magic*. Kaunas: Vitae Litera.

———. 2021. *MagicGrid® Book of Knowledge A Practical Guide to Systems Modeling Using MagicGrid from Dassault Systemes*. 2nd ed. Kaunas: Vitae Litera.

Beisbart, Claus, and Nicole J. Saam. 2019. *Computer Simulation Validation: Fundamental Concepts, Methodological Frameworks, and Philosophical Perspectives*. Simulation Foundations, Methods and Applications. Cham, Switzerland: Springer International Publishing AG. https://doi.org/10.1007/978-3-319-70766-2.

Blanchard, Benjamin S., and Wolter J. Fabrycky. 2011. *Systems Engineering and Analysis*. 5th ed. Upper Saddle River, NJ: Prentice Hall.

Buede, Dennis M. 2009. *The Engineering Design of Systems: Models and Methods*. 2nd ed. Vol. 55. Wiley Series in Systems Engineering and Management. Hoboken, NJ: John Wiley & Sons, Incorporated.

Buede, Dennis M., and William D. Miller. 2016. *The Engineering Design of Systems: Models and Methods*. 3rd ed. Wiley Series in Systems Engineering and Management. New York: Wiley.

BYJU’S. n.d. “What Are Variables, Constants, Terms in Algebraic Expressions?” BYJUS. Accessed August 26, 2023. https://byjus.com/maths/variables-and-constants-in-algebraic-expressions/.

Carr, Steven. n.d. “Natural Classification & Phylogeny.” Accessed September 15, 2023. https://www.mun.ca/biology/scarr/Natural_Classification.htm.

Cambridge Semantics. n.d. “Owl 101.” *Cambridge Semantics* (blog). Accessed August 13, 2023. https://cambridgesemantics.com/blog/semantic-university/learn-owl-rdfs/owl-101/.

Cantot, Pascal, and Dominique Luzeaux. 2013. *Simulation and Modeling of Systems of Systems*. 1st ed. Hoboken, NJ: John Wiley & Sons, Ltd. https://doi.org/10.1002/9781118616727.

Chief Information Officer. n.d.a "Background." Accessed August 30, 2023. https://dodcio.defense.gov/Library/DOD-Architecture-Framework/dodaf20_background/.

———. n.d.b "DODAF Viewpoints and Models." Accessed August 13, 2023. https://dodcio.defense.gov/Library/DOD-Architecture-Framework/dodaf20_viewpoints/.

———. n.d.c "DODAF Architecture Presentation Techniques." Accessed August 13, 2023. https://dodcio.defense.gov/Library/DOD-Architecture-Framework/dodaf20_presentation/.

College of Research and Engineering. 2019a. "Architecture Views in the SE Process." Lecture, Systems Engineering Boot Camp, Module 3, The Systems Engineering Process (CORE_411_115_SEBC_103).

———. 2019b. "Process Flow Details." Lecture, Systems Engineering Boot Camp, Module 3, The Systems Engineering Process (CORE_411_115_SEBC_103).

———. 2019c. "System Technical Baselines and System Engineering Technical Review (SETR) Events." Lecture, Systems Engineering Boot Camp, Module 9, Baselines, Requirements & Technical Reviews (CORE_411_115_SEBC_109).

———. 2019d. "System Technical Baselines Specification Tree." Lecture, Systems Engineering Boot Camp, Module 16, Systems Engineering Plan (CORE_411_115_SEBC_116).

———. 2019e. "The Systems Engineering 'Vee' with Technical and Technical Management Processes, and Systems Engineering Technical Review (SETR) Events." Lecture, Systems Engineering Boot Camp, Module 9, Baselines, Requirements & Technical Reviews (CORE_411_115_SEBC_109).

Commander Naval Air Force. 2022. *The Naval Aviation Maintenance Program (NAMP)*. COMNAVAIRFORINST 4790.2D CH-1. San Diego, CA: Department of the Navy. https://www.navair.navy.mil/Naval-Aviation-Maintenance-Program.

Crawley, Edward, Bruce Cameron, and Daniel Selva. 2016. *Systems Architecture: Strategy and Product Development for Complex Systems*. 1st ed. Essex, England: Pearson Education Limited.

Defense Acquisition University. n.d.a "Design Considerations." Lecture (online) material from ENG 201 Applied Systems Engineering in Defense Acquisition Part 1, Lesson 2.1.

———. n.d.b “Technical Management Processes Overview.” Accessed August 13, 2023. https://www.dau.edu:443/tools/se-brainbook/Pages/Management%20Processes/Overview.aspx.

———. n.d.c “Technical Processes Overview.” Accessed August 13, 2023. https://www.dau.edu:443/tools/se-brainbook/Pages/Technical%20Processes/Technical-Processes-Overview.aspx.

Delligatti, Lenny. 2013. *SysML Distilled: A Brief Guide to the Systems Modeling Language*. 1st ed. Addison-Wesley Professional.

Department of Defense. 2012. *Department of Defense Standard Practice Documentation of Verification, Validation, and Accreditation (VV&A) for Models and Simulations*. MIL-STD-3022 w/CHANGE1. Washington, DC: Department of Defense. http://everyspec.com/MIL-STD/MIL-STD-3000-9999/MIL-STD-3022_CHG-1_46836/

———. 2018. *Department of Defense Digital Engineering Strategy*. Washington, DC: Office of the Deputy Assistant Secretary of Defense for Systems Engineering. https://man.fas.org/eprint/digeng-2018.pdf

———. 2022. *Systems Engineering Guidebook*. Washington, DC.: Department of Defense. https://ac.cto.mil/wp-content/uploads/2022/02/Systems-Eng-Guidebook_Feb2022-Cleared-slp.pdf.

Department of the Navy. 2020. *United States Navy and Marine Corps Digital Systems Engineering Transformation Strategy*. Washington, DC: Secretary of the Navy. https://nps.edu/documents/112507827/0/2020+Dist+A+DON+Digital+Sys+Eng+Transformation+Strategy+2+Jun+2020.pdf/

Faisandier, Alan, and Garry Roedler. 2023. “Logical Architecture.” Systems Engineering Body of Knowledge. Last modified May 17, 2023. https://www.sebokwiki.org/wiki/Logical_Architecture.

Fonseca i Casas, Pau. 2023. “A Continuous Process for Validation, Verification, and Accreditation of Simulation Models.” *Mathematics* 11 (4): 845–865. https://doi.org/10.3390/math11040845.

Friedenthal, Sanford, Alan Moore, and Rick Steiner. 2015. *A Practical Guide to SysML The Systems Modeling Language*. 3rd ed. Waltham, MA: The Morgan Kaufmann/Object Management Group Press.

Gansch, Roman, and Ahmad Adee. 2020. “System Theoretic View on Uncertainties.” In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1345–50. Grenoble, France: IEEE. https://doi.org/10.23919/DATE48585.2020.9116472.

Goldin, Claudia, and Lawrence F. Katz. 1999. "The Shaping of Higher Educations: The Formative Years in the United States, 1890 to 1940." *Journal of Economic Perspectives* 13 (1) (Winter): 37–62. https://doi.org/10.1257/jep.13.1.37.

Green, John M. 2001. *Establishing System Measures of Effectiveness*. San Diego, CA: Raytheon Naval & Maritime Integrated Systems. https://acqnotes.com/wp-content/uploads/2014/09/Establishing-System-Measures-of-Effectiveness-by-John-Green.pdf.

Holt, Jon, and Simon Perry. 2019. *SysML for Systems Engineering: A Model-Based Approach: A Model-Based Approach*. 3rd ed. London, UK: The Institution of Engineering and Technology.

Horton, Denntrick A. 2023. "Verification and Validation of Naval Air Warfare Center Aircraft Division (NAWCAD) F/A-18 Sustainment Model." Master's thesis, Naval Postgraduate School. https://hdl.handle.net/10945/72011.

INCOSE. 2007. *Systems Engineering Vision 2020*. Document No. INCOSE-TP-2004-004-02. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/http://www.ccose.org/media/upload/SEVision2020_20071003_v2_03.pdf.

———. n.d. "System and SE Definitions." Accessed July 15, 2023. https://www.incose.org/about-systems-engineering/system-and-se-definition/system-and-se-definitions

Institute for Defense Analyses. 2023a. "O-I-D Sustainment Structure, Honeybee Tutorial." Unpublished training material for operators of the Cost Readiness Impact Model, July 12, 2023.

———. 2023b. "Simplified Supply Cycle, Tutorial Intro." Unpublished training material for operators of the Cost Readiness Impact Model, July 12, 2023.

Kaplan, David L., and M. Claire Casey. 1958. *Occupational Trends in the United States, 1900 to 1950*. Washington, D.C.: United States Census Bureau. https://www.census.gov/library/publications/1958/demo/occupational-trends-1958.html.

Kevles, Daniel. 1979. "The Physics, Mathematics, and Chemistry Communities: A Comparative Analysis". In *The Organization of Knowledge in Modern America, 1860-1920*, edited by Alexandra Oleson and John Voss, 139–72. Baltimore: Johns Hopkins University Press. http://resolver.caltech.edu/CaltechAUTHORS:20171026-145452734

Kossiakoff, Alexander, Samuel J. Seymour, David A. Flanigan, and Steven M. Biemer. 2020. *Systems Engineering Principles and Practice*. 3rd ed. Hoboken, NJ: Wiley.

McBee, Nate. 2022. "What Is Model-Based Systems Engineering (MBSE)?" *Ansys* (blog). May 25, 2022. https://www.ansys.com/blog/model-based-systems-engineering-explained.

Modarres, Mohammad. 2016. *Risk Analysis in Engineering: Techniques, Tools, and Trends*. Boca Raton, FL: CRC Press. https://doi.org/10.1201/b21429.

Moschler, Joe. 2019. "DAU News - NAVAIR Systems Engineering Transformation." *Defense Acquisition University* (blog). September 1, 2019. https://www.dau.edu:443/library/defense-atl/blog?TermStoreId=dce24bd1-67d5-449a-a845-034b26f02d5e&TermSetId=8f7816db-bdf4-4718-9235-765b669106cc&TermId=14ebb14e-a7d6-4156-8c1d-baac16c70f63&UrlSuffix=NAVAIR--Systems-Engineering-Transformation.

NAVAIR News. 2022. "Speed of Capability and Affordability Rely on Strong Industry Partnerships." March 4, 2022. https://www.navair.navy.mil/news/Speed-capability-and-affordability-rely-strong-industry-partnerships/Fri-03042022-1347.

No Magic. n.d. "Generalization." Accessed July 25, 2023. https://docs.nomagic.com/display/MD190/Generalization.

Oberkampf, William L., and Christopher J. Roy. 2010. *Verification and Validation in Scientific Computing*. New York, NY: Cambridge University Press.

Pace, Dale. 2004. "Modeling and Simulation Verification and Validation Challenges." *Johns Hopkins Apl Technical Digest* 25 (2) (2004): 163-172. https://www.semanticscholar.org/paper/Modeling-and-simulation-verification-and-validation-Pace/ca3ec5113da698a9bdb12f5bdea01d58fa2f33ba.

Pyster, Art, and Dave Olwell. 2023. "Systems Engineering and Quality Attributes." Systems Engineering Body of Knowledge. Last modified May 25, 2023. https://sebokwiki.org/wiki/Systems_Engineering_and_Quality_Attributes#:~:text=Every%20system%20has%20a%20set,interference%2C%20usability%2C%20and%20resilience.

Ruskin, Steve. 2017. "The Hunt for Vulcan…And How Albert Einstein Destroyed a Planet, Discovered Relativity, and Deciphered the Universe." *American Journal of Physics* 85 (2): 159–60. https://doi.org/10.1119/1.4968558.

Sabine, P., and C. Plumpton. 1985. *Statistics*. London: Palgrave Macmillan UK. https://doi.org/10.1007/978-1-349-07668-0.

Schlesinger, S. 1979. "Terminology for Model Credibility." *SIMULATION* 32 (3): 103–4. https://doi.org/10.1177/003754977903200304.

Sargent, Robert. 1986. "The Use of Graphical Models in Model Validation." In *Proceedings of the 18th Conference on Winter Simulation – WSC '86*, 237–41. Washington, D.C., United States: ACM Press. https://doi.org/10.1145/318242.318312.

Sargent, Robert. 2013. "Verification and Validation of Simulation Models." *Journal of Simulation* 7 (1): 12–24. https://doi.org/10.1057/jos.2012.20.

Secretary of the Navy. 2019. *Department of the Navy Modeling, Simulation, Verification, Validation, and Accreditation Management*. SECNAVINST 5200.46. Washington, DC: Department of the Navy.

Sherbrooke, Craig C. 1968. "Metric: A Multi-Echelon Technique for Recoverable Item Control." *Operations Research* 16 (1): 122–41. https://doi.org/10.1287/opre.16.1.122.

———. 2004. *Optimal Inventory Modeling of Systems Multi-Echelon Techniques*. Boston, MA: Springer. https://doi.org/10.1007/b109856.

Snoderly, John, and Alan Faisandier. 2023. "System Verification" Systems Engineering Body of Knowledge. Last modified May 17, 2023. https://sebokwiki.org/wiki/System_Verification.

Stanford University. n.d. "What Is an Ontology?" Accessed August 13, 2023. http://www-ksl.stanford.edu/kst/what-is-an-ontology.html.

Swarthmore College. 2009. "Deterministic System." Last modified July 9, 2009. https://mathimages.swarthmore.edu/index.php/Talk:Deterministic_system

SysML.org. n.d. "SysML Specifications: Current Version - OMG SysML 1.6." Accessed August 15, 2023. https://sysml.org/sysml-specs/index.html.

Systems Management College, Department of Defense. 2001. *Systems Engineering Fundamentals*. Fort Belvoir, VA: Defense Acquisition University Press.

Tamr Core. n.d. "Managing a Taxonomy." Tamr Core Documentation. Accessed September 15, 2023. https://docs.tamr.com.

Thomas, Jason. 2023. "Digital Transformation Is Changing How We Do Business at NAWCAD." *NAWCAD Communicator*, June 2023. https://flankspeed.sharepoint-mil.us/sites/NAVAIR_NAWCAD_HQ_PAO/SitePages/Digital-Transformation-is-changing-how-we-do-business-at-NAWCAD.aspx?sw=auth.

Ulrich, Karl T., and Steven D. Eppinger. 2012. *Product Design and Development*. New York, NY: McGraw-Hill.

Vaneman, Warren, Ronald R. Carlson, Ronald E. Giachetti, and Corina L. White. 2022. "Developing a Model-Based Systems Engineering (MBSE) Land Domain Construct for Marine Corps Systems Command." Technical Report. Monterey, California: Naval Postgraduate School. https://calhoun.nps.edu/handle/10945/71800.

Weilkiens, Tim. 2007. *Systems Engineering with SysML/UML Modeling, Analysis, Design*. 1st edition. Amsterdam : Morgan Kaufmann OMG Press/Elsevier.

Weilkiens, Tim, Jesko G. Lamm, Stephan Roth, and Markus Walker. 2015. *Model-Based System Architecture*. Newark, NJ: Wiley. https://doi.org/10.1002/9781119051930.

Wikipedia. 2023a. S.v. "There Are Unknown Unknowns." Last modified July 26, 2023. https://en.wikipedia.org/w/index.php?title=There_are_unknown_unknowns&oldid=1167257988.

———. 2023b. "Johari Window." Last modified July 28, 2023. https://en.wikipedia.org/w/index.php?title=Johari_window&oldid=1167507790.

Xue, Xingsi, Qi Wu, Miao Ye, and Jianhui Lv. 2022. "Efficient Ontology Meta-Matching Based on Interpolation Model Assisted Evolutionary Algorithm." *Mathematics* 10 (September): 3212. https://doi.org/10.3390/math10173212.

Zachman, John A. n.d. "About the Zachman Framework." Zachman International – FEAC Institute. Accessed August 13, 2023. https://zachman-feac.com/zachman/about-the-zachman-framework.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California



DUDLEY KNOX LIBRARY

NAVAL POSTGRADUATE SCHOOL

WWW.NPS.EDU

WHERE SCIENCE MEETS THE ART OF WARFARE