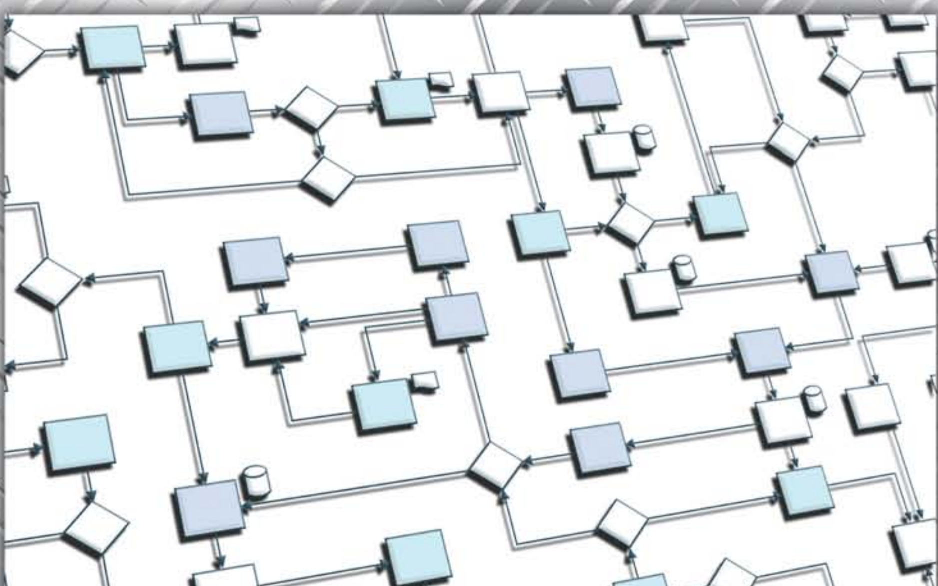
Complex and Enterprise Systems Engineering Series

Problem
Approaches
Architecture
Alternatives
Solutions
Contingencies
Implementations
Capabilities
Effectiveness
C&ESE

ARCHITECTURE AND PRINCIPLES OF SYSTEMS ENGINEERING



C.E. Dickerson
D.N. Mavris

CRC Press
Taylor & Francis Group
AN AUERBACH BOOK

ARCHITECTURE AND PRINCIPLES OF SYSTEMS ENGINEERING

COMPLEX AND ENTERPRISE SYSTEMS ENGINEERING

Series Editors: Paul R. Garvey and Brian E. White
The MITRE Corporation
www.enterprise-systems-engineering.com

Designing Complex Systems: Foundations of Design in the Functional Domain
Erik W. Aslaksen
ISBN: 978-1-4200-8753-6
Publication Date: October 2008

Architecture and Principles of Systems Engineering
Charles Dickerson and Dimitri N. Mavris
ISBN: 978-1-4200-7253-2
Publication Date: May 2009

Model-Oriented Systems Engineering Science: A Unifying Framework for Traditional and Complex Systems
Duane W. Hybertson
ISBN: 978-1-4200-7251-8
Publication Date: May 2009

Enterprise Systems Engineering: Theory and Practice
George Rebovich, Jr. and Brian E. White
ISBN: 978-1-4200-7329-4
Publication Date: October 2009

Leadership in Decentralized Organizations
Beverly G. McCarter and Brian E. White
ISBN: 978-1-4200-7417-8
Publication Date: October 2009

Complex Enterprise Systems Engineering for Operational Excellence
Kenneth C. Hoffman and Kirkor Bozdogan
ISBN: 978-1-4200-8256-2
Publication Date: November 2009

Engineering Mega-Systems: The Challenge of Systems Engineering in the Information Age
Renee Stevens
ISBN: 978-1-4200-7666-0
Publication Date: December 2009

Social and Cognitive Aspects of Engineering Practice
Stuart S. Shapiro
ISBN: 978-1-4200-7333-1
Publication Date: March 2010

RELATED BOOKS

Analytical Methods for Risk Management: A Systems Engineering Perspective
Paul R. Garvey
ISBN: 978-1-58488-637-2

Probability Methods for Cost Uncertainty Analysis: A Systems Engineering Perspective
Paul R. Garvey
ISBN: 978-0-8247-8966-4

AUERBACH PUBLICATIONS

www.auerbach-publications.com
To Order Call: 1-800-272-7737 • Fax: 1-800-374-3401
E-mail: orders@crcpress.com

ARCHITECTURE AND PRINCIPLES OF SYSTEMS ENGINEERING

C.E. Dickerson
D.N. Mavris

CRC Press
Taylor & Francis Group
Boca Raton London New York

CRC Press is an imprint of the
Taylor & Francis Group, an **informa** business
AN AUERBACH BOOK

Complex and Enterprise Systems Engineering Series

Auerbach Publications
Taylor & Francis Group
6000 Broken Sound Parkway NW, Suite 300
Boca Raton, FL 33487-2742

Printed in the United States of America on acid-free paper
10 9 8 7 6 5 4 3 2 1

International Standard Book Number: 978-1-4200-7253-2 (Hardback)

Library of Congress Cataloging-in-Publication Data

Dickerson, Charles, 1949-
Architecture and principles of systems engineering / Charles Dickerson, Dimitri N. Mavris.
p. cm. -- (Complex and enterprise systems engineering)
Includes bibliographical references and index.
ISBN 978-1-4200-7253-2 (hardcover : alk. paper)
1. Systems engineering. 2. Computer architecture. I. Mavris, Dimitri. II. Title.

TA168.D643 2009
620.001’171--dc22 2009007601

Visit the Taylor & Francis Web site at
http://www.taylorandfrancis.com

and the Auerbach Web site at
http://www.auerbach-publications.com

This book is dedicated to our families who have sacrificed their time with us so that we could finish this important work.

Contents

Acknowledgments .... **xix**

List of Principles Used for Model-Based Architecture and Systems Engineering .... **xxi**

The Authors .... **xxiii**

List of Figures by Chapter .... **xxv**

SECTION 1 FOUNDATIONS OF ARCHITECTURE AND SYSTEMS ENGINEERING

1 Introduction **3**
Reference 7

2 Logical and Scientific Approach **9**
Motivation and Background 9
Scientific Basis of Engineering 12
Experimental and Logical Basis of Science 13
Logical Modeling of Sentences 15
Relationship of Logical Modeling to Science and the Predicate Calculus 15
Details of the Procedure for the Recommended Modeling Approach 16
Detailed Illustration of the Approach to the Term "System" 18
A Suggested Model-Based Definition of Systems Engineering 23
Summary 24
References 25

3 Concepts, Standards, and Terminology **27**
Systems Engineering Standards 27
Standards-Based Definitions of Systems Engineering 29
The Systems Engineering Vee 30
Implications of the Vee Model for the Logical Diagram for System 31

Definitions and Models of Key Terms 32
- Abstraction 34
- Model 34
- Relations and Relationships 34
- Combinations and Arrangements 35
- Interactions 35
- System 36
- System of Systems (SoS) 36
- Family of Systems (FoS) 37
- Structure 38
- System Architecture 39
- System Boundary 41
- System Interface 41
- System Behavior 41
- Capability: Military Definitions 42
- Requirements Engineering 42

Meeting the Challenge of Standardizing Systems Engineering Terminology 43
Summary 43
References 44

4 Structure, Analysis, Design, and Models 45

Logical Models 46
Using Logical Models in a Document-Based Systems Engineering Process 46
System Design and Model Transforms 47
- Model Transformations in MDA™ 48
- Quality Function Deployment (QFD) in Systems Engineering 48
- A Matrix Notation for Models and Model Transformations 48
  - Matrix Notation and Conventions for Models 49
  - Matrix Notation and Conventions for Model Transformations 50
  - Model Transformation in a Simple Design Problem 50
  - Details of the Matrix Transformation in the Simple Design Problem 51

Structured Analysis and Design 52
- Structured Design 52
- Structured Analysis 53

Using Logical Models in Structured Analysis and Design 54
- Relation of Model Transformations to Structured Analysis and Design 54

Summary 55
References 55

SECTION 2 MODELING LANGUAGES, FRAMEWORKS, AND GRAPHICAL TOOLS

5 Architecture Modeling Languages 59
Mathematical Logic 59
Propositional Calculus 60
Propositional Formulae 60
Interpretation: Truth Values 60
Predicate Calculus 61
What Is the Predicate Calculus? 61
Examples of Predicates and Interpretation 61
Scope and Interpretation of Predicates 62
Quantifier Symbols, Models, and Sentences in the Predicate Calculus 62
Example of a Model of a Sentence 63
Interpretation: Models and Validity 63
Object Management Group 63
OMG Modeling Languages 64
Unified Modeling Language (UML) 64
Use Case Diagrams 64
Class Diagrams 65
Package Diagrams 66
Object Diagrams 67
Sequence Diagrams 67
Systems Modeling Language (SysML) 68
Summary 69
References 70

6 Applications of SysML to Modeling and Simulation 71
RUSSELL PEAK
Introduction to SysML and COBs 72
OMG SysML and Parametrics 72
Motivation for SysML Parametrics 73
The Composable Object (COB) Knowledge Representation 74
Introductory Concepts and Tutorial Examples 77
Triangles and Prisms 77
Representation Using Classical COB Formulations 77
Representation Using SysML 79
Implementation and Execution 83
More about Composable Object (COB) Technology 84
Analytical Spring Systems 85
Representation Using Classical COB Formulations 85
Representation Using SysML 90

Implementation and Execution....................91
Analysis Building Block Examples....................91
System Examples....................95
Discussion....................100
Summary and Conclusions....................105
Acknowledgments....................106
References....................106

7 DFD and IDEF0 Graphical Tools....................109
Graphical Tools....................109
Data Flow Diagrams....................110
Strengths and Weaknesses of DFDs....................113
IDEF0 Standard....................114
Strengths and Weaknesses of IDEF0....................117
Summary....................117
References....................117

8 Rule and State Modeling....................119
Control Structures....................119
Rule Modeling....................121
Structured English....................121
Decision Trees....................122
Decision Tables....................123
Decision Tables or Decision Trees?....................125
State Modeling....................126
States in Engineering and Physics....................126
States and Events in Chess....................127
State Transitions and Events: Concepts and Terminology....................129
State Diagrams....................129
State Transition Diagrams....................130
State Transition Diagram Summary....................131
References....................131

9 Data and Information Modeling....................133
Entity-Relationship (E-R) Diagrams....................133
E-R Concepts and Terminology....................134
Incorporating Semantic Networks into E-R....................135
E-R Modeling Summary....................139
IDEF1x Standard....................139
IDEF1x Concepts and Terminology....................139
Entities in IDEF1x....................140
Attributes in IDEF1x....................141

Domains in IDEF1x ... 141
Relationships in IDEF1x ... 142
IDEF1x Summary ... 144
References ... 145

10 Introduction to DoDAF and MODAF Frameworks ... 147
What Are Architecture Frameworks? ... 147
History of DoDAF ... 148
DoD Architecture Framework (DoDAF) ... 149
Structure of DoDAF ... 150
DoDAF Architecture Development Principles, Expectations, and Process ... 152
History of MODAF ... 153
MOD Architecture Framework (MODAF) ... 154
Summary ... 155
References ... 155

11 DoDAF and MODAF Artifacts ... 157
KELLY GRIENDLING
Introduction ... 158
DoDAF Products ... 158
All View ... 159
Operational View ... 160
System and Services View ... 169
Technical View ... 178
MODAF ... 179
Strategic Viewpoint ... 179
Acquisition Viewpoint ... 180
Service-Oriented Viewpoint ... 181
Summary ... 182
References ... 182

12 Other Architecture Frameworks ... 185
Enterprise Architecture Frameworks ... 185
The Open Group Architecture Framework (TOGAF) ... 186
Zachman Framework ... 188
Other Architecture Frameworks ... 189
High-Level Architecture (HLA) ... 190
Summary ... 190
References ... 191

SECTION 3 USING ARCHITECTURE MODELS IN SYSTEMS ANALYSIS AND DESIGN

13 Modeling FBE-I with DoDAF 195
The Fleet Battle Experiments (FBEs) 195
Modeling NCO 196
Operational Concept for FBE-I 200
Using DoDAF Architecture to Model Targeting in FBE-I 203
Using DoDAF Architecture for Interoperability Assessments 209
Summary 210
References 213

14 Capabilities Assessment 215
Motivation 215
Concept Refinement and Capabilities Analysis 216
Process for Concept Definition and Refinement 217
Sensor Fusion Node Case Study 217
Capability Model 219
Design Change to the Sensor Architecture 221
Architecture Analysis 222
Capability Analysis 223
Concept Refinement 224
Summary of Concept Refinement and Capability Analysis 225
Analysis of Alternatives (AoA) at the FoS Level 225
Background and Attribution 225
Problem Statement 226
Analysis of Alternatives for Link 16 ICP Trades 228
Relation of the ICPs to Operational Capabilities 228
Choosing a Portfolio of ICPs from the Bundles 230
The Traditional Cost–Benefit Trade Elevated to a Capability-Based FoS Level 232
Summary of AoA at the FoS Level 232
References 234

15 Toward Systems of Systems and Network-Enabled Capabilities 235
Background 235
The Move toward Capability Engineering 236
Late 20th-Century SoS History in U.S. Defense Systems 237
Description of Forces and Technologies 237
A Revolution in Military Affairs 238
A Transition to Network Enablement and SoS 238
Network Enablement of Naval Forces 238
NCW Report to Congress 240

From Network-Centric Enablement to SoS 241
SoSE for Air Force Capability Development....242
SoS or Network Enablement? 242
A Lesson from History....243
Systems Engineering Considerations....243
Critical Roles of Information Exchange and Capabilities Integration....245
Examples of U.S. DoD Defense Applications of SoS 248
OSD SoS SE Guide 248
Future Combat System (FCS)....249
Single Integrated Air Picture (SIAP) 249
Naval Integrated Fire Control–Counter Air (NIFC-CA) 250
Commissary Advanced Resale Transaction System (CARTS) 250
Related Work and Research Opportunities....251
References....251

16 Model Driven Architecture 253
Motivation....253
MDA Concepts and Terminology....256
Transformation Example 258
Details of MDA Model Transformation 262
Relation of MDA to Systems Engineering.... 264
Relation of MDA to SoS Engineering....265
Summary.... 266
References....267

SECTION 4 AEROSPACE AND DEFENSE SYSTEMS ENGINEERING

17 Fundamentals of Systems Engineering....271
Fundamental Concepts in Systems Engineering 271
What Is Systems Engineering?....273
The "Vee" Model 274
The DoD Systems Engineering Process....276
Other Systems Engineering Processes 284
Summary....287
References....288

18 Capability-Based Acquisition Policy....289
U.S. DoD Acquisition Policy....290
The DoD 5000....290
Joint Capabilities Integration and Development System (JCIDS)...291
The Role of Systems Engineering in Acquisition....296

MOD Acquisition Policy 297
Defence Acquisition Operating Framework (AOF)........................298
Transitions in TLCM from National Capability to Lines of Development 298
References..299

19 Systems Design..301
DMITRI MAVRIS AND HERNANDO JIMENEZ
Introduction 301
A Characterization of Design..302
Design as a Phased Process 303
Problem Definition for Design Context.. 304
Requirements Analysis.. 306
System Decomposition 307
Solution Definition.. 308
Decomposition and Definition: Second Cycle 311
Additional Iterations..313
Supporting Information Transformation in Design with Quality Function Deployment..314
Deployment of the QFD House of Quality 319
Summary..321
References..322

20 Advanced Design Methods..323
DMITRI MAVRIS AND HERNANDO JIMENEZ
Introduction 323
Interactive Reconfigurable Matrix of Alternatives..................................324
Formulation..324
Enabler: Multi-Attribute Decision Making....................................325
Enabler: Visual Analytics..326
Implementation 326
Genetic Algorithm Concept Space Exploration 329
Formulation..329
Enabler: Genetic Algorithm..329
Implementation 330
Sensitivity Profiling..333
Formulation..333
Enabler: Surrogate Models..333
Implementation 336
Probabilistic Design Methods..339
Formulation..339
Probabilistic Design Space Exploration...341
Technology Evaluation 347

Summary 349
References 349

21 Wicked Problems 353
ANDREW J. DAW
The Acquisition Context 354
"Wicked Problems" 357
Concepts for Managing the Wicked Problem 362
The Characterization of the Four-Blob Model: Project Style 362
The Characterization of the Four-Blob Model: Engineering Style 364
System Analysis Support 367
Acquisition Life-Cycle Considerations 368
Information Management 373
The Characterization of the Four-Blob Model: Skillset and Competencies 375
The Characterization of the Four-Blob Model: Organizational Style 377
The Characterization of the Four-Blob Model: Commercial Style 378
A Mapping of the Wicked Problem Characteristics and the Management Issues Discussed 381
Conclusions 381
Acknowledgments 386
References 387

SECTION 5 CASE STUDIES

22 Fleet Battle Experiment-India (FBE-I) Concept Model 391
Background 391
Class Exercise 392
Instructions to the Class 392
Class Assessment 393
FBE-I Concept Statement 393
Instructor's Response 393

23 Air Warfare Model 397
PHILIP JOHNSON
Background 397
Class Exercise 398
Instructions to the Class 398
Class Assessment 399
Air Warfare System Concept Statement 399
Instructor's Response 400
Executable Architectures 403

Partitioning Systems 404
Domain 404
The Implementation of xUML within Kennedy Carter's iUMLite........ 406
A Practical Example of Reusing Executable Architectures with iUMLite........ 407
Simulating the Air Warfare Architecture 410
Toward Future Executable Architectures 415
Reference 415

24 Long-Range Strike System Design Case Study 417
WILLIAM ENGLER
Class Exercise........ 417
Instructions to the Class 418
Class Assessment........ 418
Long-Range Strike System Concept Statement 419
SWARMing 419
Exercise 1 420
Exercise 1: Instructor's Response 420
Management and Planning Tools 423
Affinity Diagram 423
Exercise 2 423
Exercise 2: Instructor's Response 423
Interrelationship Digraph 424
Exercise 3 424
Exercise 3: Instructor's Response 425
Tree Diagram 425
Exercise 4 425
Exercise 4: Instructor's Response 426
Prioritization Matrix........ 426
Exercise 5 426
Exercise 5: Instructor's Response 426
Process Decision Program Chart 426
Exercise 6 428
Exercise 6: Instructor's Response 428
Activity Network........ 429
Exercise 7 429
Exercise 7: Instructor's Response........ 429
Quality Function Deployment 429
Exercise 8 430
Exercise 8: Instructor's Response 430

Creation of Alternatives 432
Functional and Physical Decomposition 434
Matrix of Alternatives 434
Exercise 9 435
IRMA 435
Exercise 9: Instructor's Response 435
Quantitative Modeling 436
Surrogate Modeling 437
Selection of Alternatives 438
Summary 438
References 439

25 Remote Monitoring 441
Background 441
Class Exercise 442
Instructions to Class 442
Class Assessment 443
Remote Monitoring System Concept 443
Technical Details 444
Architecture Development 444
The Developed Architecture 444
Architecture-Based Assessment 446
Case Study Summary 450

Index 451

Acknowledgments

The material in this course is from a collaborative effort by Professor C.E. Dickerson of Loughborough University and Professor D.N. Mavris of the Georgia Institute of Technology. We each wish to express special thanks to our Ph.D. students who have generously devoted their personal time to support the creation of this material: Phil Johnson, Hernando Jimenez, Kelly Griendling, and William Engler.

Each has directly contributed to the writing of this book, as noted by their authorship and co-authorship of individual chapters. And their overall technical contributions and insights on the course have been invaluable.

This course is based on but substantially extends the Advanced Systems Design M.Sc. course for aerospace engineering at the Georgia Institute of Technology and the previous Systems Architecture M.Sc. course taught at Loughborough University on the subject of defense systems architecture and frameworks.

The defense systems architecture course at Loughborough University had been taught by Professor Alex Levis and Dr. Lee Wagenhals, both of George Mason University in Virginia. Course materials from all three universities have been integrated into this book.

The authors have been fortunate to receive key contributions from two leading subject matter experts, one from the U.S. and one from the U.K.

Dr. Russell Peak, a Senior Research staff at the Georgia Institute of Technology, has contributed the chapter on the application of the Systems Modeling Language (SysML) to modeling and simulation. Dr. Peak is the Director of the Modeling and Simulation Lab at Georgia Tech and represents the University to the OMG SysML Task Force.

Mr. Andrew Daw, Chief System Engineer at BAE Systems for Capability Development, is a recognized thought leader for Through Life Capability Management. He has contributed the chapter on the 'wicked' aspects of acquiring defense capabilities. Mr. Daw is the past president of the U.K. Chapter of the International Council on Systems Engineering (INCOSE).

List of Principles Used for Model-Based Architecture and Systems Engineering

Conceptual Integrity and the Role of the Architect

Conceptual integrity is the most important consideration in system design. The architect should be responsible for the conceptual integrity of all aspects of the product perceivable by the user.

The Principle of Definition

One needs both a formal definition of a design, for precision, and a prose definition for comprehensibility.

Model Transformation

Model transformations relate to system design and should preserve the relationships *between the parameters being modeled*.

Reflection of Structure in System Design

The solution should reflect the inherent structure of the problem.

Modular Structured Design

Systems should be comprised of modules, each of which is highly cohesive but collectively are loosely coupled.

Structured Analysis

The specification of the problem should be separated from that of the solution.

The Authors

Professor C.E. Dickerson is the Royal Academy of Engineering Chair of Systems Engineering at Loughborough University in the United Kingdom. His research program at the university is focused on model-driven architecture and systems engineering. He has authored numerous papers as well as an internationally recognized book on military systems architecture, and has co-authored key government reports. As a member of IEEE and OMG, and as the Chair of the INCOSE Architecture Working Group, he works with the systems engineering community on systems architecture practice and standards.

Before joining Loughborough University, he was a Technical Fellow at BAE Systems, providing corporate leadership for architecture-based and system of systems engineering. Previously, as a member of MIT Lincoln Laboratory, he conducted tests and research on electromagnetic scattering. As an MIT IPA, he served as Aegis Systems engineer for the U.S. Navy Theater-Wide Program and then as the Director of Architecture for the Chief Engineer of the U.S. Navy. His aerospace experience includes air vehicle survivability and avionics design at the Lockheed Skunk Works in the Burbank (California) facility and the Northrop Advanced Systems Division, and operations analysis at the Center for Naval Analyses. He received a Ph.D. degree from Purdue University in 1980.

Dimitri Mavris is the Boeing Professor of Advanced Aerospace Systems Analysis at the Guggenheim School of Aerospace Engineering, Georgia Institute of Technology. Throughout his academic career he has authored and co-authored over 60 refereed publications, more than 300 conference papers, and more than 100 technical reports. Since his initial appointment as academic faculty in 1996, Dr. Mavris has graduated more than 110 master's degree students and 50 doctoral students.

In addition to his academic duties, Dr. Mavris is the founder and director of Georgia Tech's Aerospace Systems Design Laboratory (ASDL), a unique academic organization home to 200 researchers and graduate-level students actively pursuing research in the areas of multidisciplinary analysis, design, and optimization; MDO/A; and nondeterministic design theory. Since its inception in 1992, ASDL has been named a Center of Excellence in Robust Systems Design and Optimization under the General Electric University Strategic Alliance (GE USA), and a NASA/DoD University Research Engineering Technology Institute (URETI) on Aeropropulsion and Power Technology (UAPT). In addition, ASDL is a member of the Federal Aviation Administration's Center of Excellence under the Partnership for Air Transportation Noise and Emissions Reduction (PARTNER).

Dr. Mavris is an associate fellow of the American Institute of Aeronautics and Astronautics (AIAA) and fellow of the National Institute of Aerospace. He also serves as Deputy Director for AIAA's Aircraft Technology Integration and Operations Group and as a Chair for the AIAA Energy Optimized Aircraft and Equipment Systems Program Committee.

List of Figures

SECTION 1 FOUNDATIONS OF ARCHITECTURE AND SYSTEMS ENGINEERING

1. Introduction

There are no figures in this chapter.

2. Logical and Scientific Approach

Figure 2.1 The Collaborative Visualization Environment (CoVE).

Figure 2.2 Modeling and formalizing systems engineering.

Figure 2.3 The relationship between **system** and ***stated purposes.***

Figure 2.4 Reasonable relationships between **system**, ***combination, elements;*** and the relationship between **system** and ***stated purposes***.

Figure 2.5 The relationship between ***combinations*** and ***stated purposes.***

Figure 2.6 Logical assessment of the diagram for **system.**

Figure 2.7 Interpretation of ***parts*** in the Hitchins definition as the ***combinations*** of ***elements*** of the INCOSE definition**.**

Figure 2.8 The issue of realization of ***properties, capabilities,*** and ***behaviors*** from the combinations in the diagram for the INCOSE definition of **system.**

Figure 2.9 The issue of ***emergence*** from ***interactions.***

Figure 2.10 Initial extension of the diagram for **system.**

Figure 2.11 Final issues to be resolved in the diagram for **system.**

Figure 2.12 An extension of the diagram for **system.**

3. Concepts, Standards, and Terminology

Figure 3.1 The Systems Engineering Vee model.

Figure 3.2 What is the implication of the Vee for *system*?

Figure 3.3 Adaptation of the INCOSE and Hitchins definition to the Systems Engineering Vee.

Figure 3.4 Diagram of the term *transport*.

Figure 3.5 A model of system architecture based on IEEE Standard 610.12.

Figure 3.6 A model of the OMG MDA™ definition of system architecture.

4. Structure, Analysis, Design, and Models

Figure 4.1 Deriving a system model from systems engineering documents.

Figure 4.2 Using model transformations in system design.

SECTION 2 MODELING LANGUAGES, FRAMEWORKS, AND GRAPHICAL TOOLS

5. Architecture Modeling Languages

Figure 5.1 Use case diagram.

Figure 5.2 Class diagram.

Figure 5.3 Package diagram.

Figure 5.4 Object diagram.

Figure 5.5 Sequence diagram.

Figure 5.6 Relation between UML and SysML.

Figure 5.7 SysML diagram types.

Figure 5.8 The four pillars of SysML.

6. Applications of SysML to Modeling and Simulation

Figure 6.1 Classical lexical and graphical formulations of the COB representation: (a) COB Structure languages, and (b) COB Instance languages.

Figure 6.2 Basic COB constraint schematic notation: (a) Structure notation (-S), and (b) Instance notation (-I).

Figure 6.3 Triangles and prisms tutorial: classical COB formulations.

Figure 6.4 Triangles and prisms tutorial—SysML diagrams: (a) Triangle/prism tutorial block definition diagram, (b) RightTriangle parametric diagram, (c) TriangularPrism parametric diagram, and (d) TriangularPrism sample instance.

Figure 6.5 Sample SysML value types with specified dimensions and units.

Figure 6.6 Sample parametrics implementation in a SysML tool.

Figure 6.7 CAE solver access via XaiTools engineering Web services.

Figure 6.8 Tool architecture enabling executable SysML parametrics.

Figure 6.9 TriangularPrism instance from Figure 6.4 in XaiTools COB browser: an object-oriented noncausal spreadsheet.

Figure 6.10 Analytical springs tutorial: linear_spring classical COB structural formulations.

Figure 6.11 Multidirectional (noncausal) capabilities of a COB linear_spring instance: (a) Constraint schematic-I, and (b) lexical COB instance (COI).

Figure 6.12 Analytical springs tutorial: two_spring_system traditional math and diagram forms.

Figure 6.13 Analytical springs tutorial: two_spring_system classical COB formulations.

Figure 6.14 Analytical springs tutorial: TwoSpringSystem SysML diagrams: (a) block definition diagram, (b) LinearSpring parametric diagram, and (c) Two-SpringSystem parametric diagram.

Figure 6.15 TwoSpringSystem parametric diagram: sample instance.

Figure 6.16 COB instance generated from a SysML TwoSpringSystem model (Figure 6.15) and its execution in XaiTools.

Figure 6.17 Example mechanics of materials analysis building blocks (ABBs)—SysML parametric diagrams: (a) OneDLinear ElasticModel, (b) ExtensionalRod, and (c) TorsionalRod.

Figure 6.18 Problem overview: road scanner system using LittleEye UAVs. (From Zwemer, D.A. and Bajaj, M. 2008. SysML parametrics and progress towards multi-solvers and next-generation object-oriented spreadsheets. *Frontiers in Design & Simulation Workshop*, Georgia Tech PSLM Center, Atlanta.)

Figure 6.19 LittleEye SysML model: various diagram views. (From Zwemer, D.A. and Bajaj, M. 2008. SysML parametrics and progress towards multi-solvers and next-generation object-oriented spreadsheets. *Frontiers in Design & Simulation Workshop*, Georgia Tech PSLM Center, Atlanta.)

Figure 6.20 Solving LittleEye SysML parametrics: ParaMagic browser views. (From Zwemer, D.A. and Bajaj, M. 2008. SysML parametrics and progress towards multi-solvers and next-generation object-oriented spreadsheets. *Frontiers in Design & Simulation Workshop*, Georgia Tech PSLM Center, Atlanta.)

Figure 6.21 Problem overview: financial projections system. (From Zwemer, D.A. and Bajaj, M. 2008. SysML parametrics and progress towards multi-solvers and next-generation object-oriented spreadsheets. *Frontiers in Design & Simulation Workshop*, Georgia Tech PSLM Center, Atlanta.)

Figure 6.22 Financial projections SysML model: various diagram views. (From Zwemer, D.A. and Bajaj, M. 2008. SysML parametrics and progress towards multi-solvers and next-generation object-oriented spreadsheets. *Frontiers in Design & Simulation Workshop*, Georgia Tech PSLM Center, Atlanta.)

Figure 6.23 Solving financial projections SysML parametrics: ParaMagic browser views. (From Zwemer, D.A. and Bajaj, M. 2008. SysML parametrics and progress towards multi-solvers and next-generation object-oriented spreadsheets. *Frontiers in Design & Simulation Workshop*, Georgia Tech PSLM Center, Atlanta.)

Figure 6.24 Excavator modeling and simulation test bed: tool categories view. (From Peak, R.S., Burkhart, R.M., Friedenthal, S., Paredis, C.J.J., and McGinnis, L.M. 2008. Integrating design with simulation & analysis using SysML—Mechatronics/interoperability team status report. Presentation to INCOSE MBSE Challenge Team, Utrecht, Holland.)

Figure 6.25 Excavator operational domain: top-level context diagram. (From Peak, R.S., Burkhart, R.M., Friedenthal, S., Paredis, C.J.J., and McGinnis, L.M. 2008. Integrating design with simulation & analysis using SysML—Mechatronics/interoperability team status report. Presentation to INCOSE MBSE Challenge Team, Utrecht, Holland.)

Figure 6.26 Excavator hydraulics system simulation in Dymola. (From Johnson, T.A., Paredis, C.J.J., and Burkhart, R.M. 2008. Integrating models and simulations of continuous dynamics into SysML. *Proc. 6th Intl. Modelica Conf.*)

Figure 6.27 SysML-based panorama for high diversity CAD–CAE interoperability: flap linkage tutorial. (From Peak, R.S., Burkhart, R.M., Friedenthal, S., Wilson, M.W., Bajaj, M., and Kim, I. 2007b. Simulation-based design using SysML—Part 2: Celebrating diversity by example. *INCOSE Int. Symposium*, San Diego.)

7. DFD and IDEF0 Graphical Tools

Figure 7.1 Typical symbols for DFD elements.

Figure 7.2 Context diagram (Environmental Model).

Figure 7.3 0-diagram (Behavioral Model).

Figure 7.4 IDEF0 basic notation.

Figure 7.5 IDEF0 top-level context diagram (A0).

Figure 7.6 IDEF0 diagram example.

8. Rule and State Modeling

Figure 8.1 Control structures for sequences and decisions.

Figure 8.2 Loops in control structures.

Figure 8.3 Decision tree example.

Figure 8.4 The four parts of a decision table.

Figure 8.5 Rule specification example.

Figure 8.6 Relation of events, actions, and states.

Figure 8.7 Example of state transition diagram.

9. Data and Information Modeling

Figure 9.1 Symbols for E-R graphical elements.

Figure 9.2 Aggregation structure.

Figure 9.3 Example of generalization–specialization structure.

Figure 9.4 Example of association abstraction.

Figure 9.5 Example of cardinality expressed in E-R.

Figure 9.6 Example of E-R recursive relationships.

Figure 9.7 Graphical representation of entities in IDEF1x.

Figure 9.8 Attribute syntax.

Figure 9.9 Example of base and typed domains.

Figure 9.10 Entity-Relationship Model diagram in IDEF1x.

10. Introduction to DoDAF and MODAF Frameworks

Figure 10.1 DoDAF layers. (Adapted from Department of Defense. 2007a. Version 1.5, *Department of Defense Architecture Framework (DODAF), Volume I: Definitions and Guidelines*.)

Figure 10.2 DoDAF views and their relationships. (Adapted from the Department of Defense. 2004. *Department of Defense Architecture Framework (DoDAF), Deskbook*.)

Figure 10.3 DoDAF architecture description process. (Adapted from Department of Defense. 2007a. Version 1.5, *Department of Defense Architecture Framework (DODAF), Volume I: Definitions and Guidelines.*)

Figure 10.4 Relationship between DoDAF and MODAF. (Adapted from Ministry of Defence (MOD). 2008. *The Ministry of Defence Architecture Framework Version 1.2.* http://www.modaf.org.uk/ [accessed June 26, 2008].)

11. DoDAF and MODAF Artifacts

Figure 11.1 Overview of DoDAF products. (Reproduced from Hardy, D., OMG UML Profile for the DoD and MoD Architecture Frameworks. http://syseng.omg.org/UPDM%20for%20DODAF%20WGFADO-INCOSE%20AWG-070130.ppt.)

Figure 11.2 Notional AV-1.

Figure 11.3 Example OV-1 in UML.

Figure 11.4 Example OV-2 in UML.

Figure 11.5 Example OV-3 in UML.

Figure 11.6 Example OV-4 in UML.

Figure 11.7 Example OV-5 in UML.

Figure 11.8 Example OV-5 in UML with annotations.

Figure 11.9 Example OV-6a in UML.

Figure 11.10 Example OV-6b in UML.

Figure 11.11 Example OV-6c in UML.

Figure 11.12 Example OV-7 in UML.

Figure 11.13 Example SV-1 in UML.

Figure 11.14 Example SV-2 in UML.

12. Other Architecture Frameworks
There are no figures in this chapter.

SECTION 3 USING ARCHITECTURE MODELS IN SYSTEMS ANALYSIS AND DESIGN

13. Modeling FBE-I with DoDAF

Figure 13.1 Key elements of the operational concept for warfare.

Figure 13.2 System of systems (SoS) integration through networks.

Figure 13.3 Network-centric aspect of the military force.

Figure 13.4 Illustrative Joint Targeting Doctrine.

Figure 13.5 Joint Targeting Doctrine augmented for TST.

Figure 13.6 FBE-I operational concept.

Figure 13.7 OV-4 organizational relationships chart and OV-2 operational node connectivity.

Figure 13.8 Functional decomposition (high-level OV-5 hierarchy).

Figure 13.9 OV-6c event trace diagram (an architectural model of capabilities).

Figure 13.10 Relationships of architecture, interoperability, and capability.

Figure 13.11 Top-level FBE-I TCS SV-1.

Figure 13.12 Scenario-WESTPAC TACSIT-4 (F-S).

Figure 13.13 Example of a detailed SV-6.

Figure 13.14 Scenario-WESTPAC TACSIT-4 (F-S) interoperability assessment.

Figure 13.15 Net-centric readiness checklists.

14. Capabilities Assessment

Figure 14.1 Realizing network-enabled operational capabilities.

Figure 14.2 Capability model F2T.

Figure 14.3 Sensor fusion node description (external view of the node).

Figure 14.4 Communications architecture supports fusion node.

Figure 14.5 Sensor fusion node (internal view of the node).

Figure 14.6 Measures of performance.

Figure 14.7 Assessment of alternatives (example of Link 16 ICP interoperability trades).

Figure 14.8 Multiple fixes can be required for each gap.

Figure 14.9 Choosing a portfolio from bundles.

Figure 14.10 Using system bundles for FoS level trades.

15. Toward Systems of Systems and Network-Enabled Capabilities

Figure 15.1 System of systems integration through networks.

Figure 15.2 Network-centric region of the information domain.

Figure 15.3 A logical model of system.

Figure 15.4 Fleet Battle Experiment-India operational concept.

16. Model Driven Architecture

Figure 16.1 MDA™ cost savings over program life: Codagen testimonial to OMG.

Figure 16.2 A simple view of the MDA™ approach. (Adapted from Jones, V., van Halteren, A., Konstantas, D., Widya, I., and Bults, R. [2007]. *International Journal of Business Process Integration and Management*, 2(3):215–229).

Figure 16.3 Logical model of what the payroll system must do.

Figure 16.4 A DFD model of what the payroll system must do.

Figure 16.5 Implementation issue for the payroll system.

Figure 16.6 Example of a type mapping in a flow transform.

Figure 16.7 Example of an instance mapping in a flow transform.

SECTION 4 AEROSPACE AND DEFENSE SYSTEMS ENGINEERING

17. Fundamentals of Systems Engineering

Figure 17.1 The Systems Engineering "Vee." (Adapted from Forsberg and Mooz "vee" model.)

Figure 17.2 The U.S. Department of Defense systems engineering process. (From Department of Defense (DoD), United States. 2001. *Systems Engineering Fundamentals*.)

Figure 17.3 Notional functional decomposition and allocation for an aircraft.

Figure 17.4 Notional physical decomposition for an aircraft.

Figure 17.5 Inputs and outputs of DoD systems engineering process sequence.

Figure 17.6 Summary diagram of the DoD systems engineering process. (From Department of Defense (DoD), United States. 2001. *Systems Engineering Fundamentals*.)

Figure 17.7 Waterfall model for large software development programs. (From Royce, Winston. 1970. Managing the development of large software systems. *Proceedings of the IEEE WESCON*: 1–9.)

Figure 17.8 Spiral development model. (Boehm, B.W. 1988. A spiral model of software development and enhancement. *Computer* 21(5):61–72.)

18. Capability-Based Acquisition Policy

Figure 18.1 DoD's acquisition strategy. (Adapted from JCIDS 2007.)

Figure 18.2 JCIDS process through Milestone A.

Figure 18.3 JCIDS process from Milestone A to Milestone B.

Figure 18.4 JCIDS process from Milestone B to Milestone C.

Figure 18.5 Summary of the DoD acquisition process.

Figure 18.6 Systems engineering, requirements analysis, and defense acquisition.

Figure 18.7 Defence Acquisition Operating Framework (AOF).

Figure 18.8 Through Life Capability Management.

19. Systems Design

Figure 19.1 Matrix of alternatives for notional supersonic transport.

Figure 19.2 Compatibility matrix for notional supersonic transport subsystems.

Figure 19.3 Main elements of the QFD house of quality.

Figure 19.4 Supporting tools for Quality Function Deployment.

Figure 19.5 Sample correlation matrix.

Figure 19.6 Deployment progression in QFD.

Figure 19.7 Comparison of DoD systems engineering design process sequence and QFD progression.

20. Advanced Design Methods

Figure 20.1 IRMA concept in down-selection process. (From Engler III, W.O., Biltgen, P., and Mavris, D.N. 2007. Paper presented at the *45th AIAA Aerospace Sciences Meeting and Exhibit*, January 8–11, in Reno, NV.)

Figure 20.2 Summary of the genetic algorithm procedure.

Figure 20.3 Genetic algorithm setup for concept space exploration.

Figure 20.4 Notional visualization of genetic algorithm concept space exploration.

Figure 20.5 Notional display of sensitivity profilers.

Figure 20.6 Sample implementation of a designed experiment.

Figure 20.7 Wing attributes for sample sensitivity profiles.

Figure 20.8 Sensitivity profiles for sample aircraft.

Figure 20.9 Changing sensitivity profiles with attribute settings.

Figure 20.10 Probabilistic design space exploration process.

Figure 20.11 Notional design space exploration via joint probability distributions.

Figure 20.12 Notional display of a filtered Monte Carlo design space exploration.

Figure 20.13 Relational structure and model transforms in matrix notation, QFD, and multivariate plots.

Figure 20.14a Multivariate scatter plot for sample surrogate model.

Figure 20.14b Statistical correlation matrix for sample surrogate model.

Figure 20.15 K-factor distributions for different levels of technology maturity.

21. Wicked Problems

Figure 21.1 U.K. Joint Doctrine and Concepts Centre "Defence Capability Framework," expressing the seven "verbs" of operational capability.

Figure 21.2 The five resource classes of the industrial definition of capability.

Figure 21.3 Capability Value Chain. (MOD AOF [U.K. Ministry of Defence 2007].)

Figure 21.4 Considering a complexity perspective for project style/type. (From PA Consulting Group. 2006. Early lessons for establishing through life capability based programmes. Paper presented at the *RUSI Defence Project Management Conference*, October 10–11, London. Copyright PA Knowledge Ltd 2006. All rights reserved.)

Figure 21.5 A (systems) engineering overlay of process styles. (From PA Consulting Group. 2006. Paper presented at the *RUSI Defence Project Management Conference*, Oct. 10–11, London.)

Figure 21.6 Traditional "V" diagram systems engineering process life cycle.

Figure 21.7 Reaction Chamber Systems Engineering Process Lifecycle. (Adapted from Price S.N. and John, P. 2002. Paper presented at *IFORS 2002*, Royal Military College of Science, Shrivenham, U.K.)

Figure 21.8 Established Nine-Layer Model of synthetic environments.

Figure 21.9 Exemplar new life-cycle approach to early defense acquisition activities.

Figure 21.10a Staircase model of through life acquisition.

Figure 21.10b Combined Gantt and capability aggregation representation.

Figure 21.11 Information considerations of the interfaces.

Figure 21.12 Emergence model techniques and skills.

Figure 21.13 A structure of commercial models.

Figure 21.14 A transition from the Capability Value Chain to a defense acquisition capability model.

SECTION 5 CASE STUDIES

22. Fleet Battle Experiment-India (FBE-I) Concept Model

Figure 22.1 FBE-I operational concept.

Figure 22.2 Instructor's model of FBE-I STOM concept.

23. Air Warfare Model

Figure 23.1 The air warfare system.

Figure 23.2 The gas station system.

Figure 23.3 Functional versus domain partitioning. (From Raistrick, C. et al. 2004. *Model Driven Architecture with Executable UML*. New York: Cambridge University Press.)

Figure 23.4 Air warfare domain model.

Figure 23.5 Air warfare domain model with viewpoints.

Figure 23.6 xUML model layers.

Figure 23.7 Executable UML (xUML).

Figure 23.8 The role of sequence diagrams.

Figure 23.9 Use case: threat engagement and weapon delivery.

Figure 23.10 Sequence diagram: air warfare system engages threat.

Figure 23.11 Class diagram: air warfare control.

Figure 23.12 Class collaboration diagram: air warfare control.

Figure 23.13 Example State Machine ASL code translation.

Figure 23.14 iUMLite Simulator screenshot.

24. Long-Range Strike System Design Case Study

Figure 24.1 Range and speed of existing aircraft.

Figure 24.2 Operating radii from Diego Garcia.

Figure 24.3 Table of sortie rates.

Figure 24.4 Affinity diagram.

Figure 24.5 Interrelationship digraph.

Figure 24.6 Tree diagram.

Figure 24.7 Prioritization matrix.

Figure 24.8 Process decision program chart.

Figure 24.9 Activity network.

Figure 24.10 Long range strike QFD.

Figure 24.11 LRS engineering characteristics importance ranking.

Figure 24.12 LRS QFD 2.

Figure 24.13 Long range strike interactive reconfigurable matrix of alternatives.

25. Remote Monitoring

Figure 25.1 Dependencies in the UML package diagram.

Figure 25.2 MDA domain model.

Figure 25.3 Relation to MDA-specified model types.

Figure 25.4 Domain model throughput assessment.

Figure 25.5 Wireless technology solution space.

Figure 25.6 Domain model: software architecture highlighted.

Figure 25.7 Solution space with data compression.

1

FOUNDATIONS OF ARCHITECTURE AND SYSTEMS ENGINEERING

Chapter 1

Introduction

Why does the community need another book on architecture and systems engineering? How will this book help you personally gain a better understanding of these subjects and develop new skills that can help you in the practice or research of systems architecture and engineering?

Today, the systems engineering community is actively rethinking its concepts and practices as it undergoes dramatic growth. However, tensions exist across the various systems disciplines. In the domain of software engineering and information technology, Moore's law leads to an order-of-magnitude improvement in technical capability every four to six years, a timeframe that aerospace and defense systems enterprises can require to develop a single new system, such as an air vehicle. The development of major new systems can take even longer.

The emergence of Model Driven Architecture (MDA™) and recent initiatives for model-based systems engineering (MBSE) will play an important role in determining how the practice of architecture and systems engineering evolves over the next several years. How will this affect you as an architect or engineering professional? Major changes have occurred in the practice of software engineering over the past two decades. The times will demand that major changes occur in systems engineering in the years to come.

This book will give students and readers the foundation and elementary methods to step into the domain of model-based architecture and systems engineering practices. A special attractiveness of the approach in this book is that it is as widely applicable as the interests and needs of the practitioner, and it can be inserted at any point into any organization's practice of systems architecting and engineering. The concepts, standards, and terminology provided in this book embody the emerging model-based approaches, but are rooted in the long-standing practices of engineering, science, and mathematics. Each of the authors brings substantial experience from academic, government, and commercial research and development.

Fundamental questions are addressed. What is systems architecture? How does it relate to systems engineering? What is the role of a systems architect? How should systems architecture be practiced?

The expectation of the authors is to change how you think about architecture and systems. Our goal is to give you new skills that you can take back to your workplace or research program. However, your ability to reason in new ways from this book is more important than the details of any information that you might gain from it.

Readers of this book should ask themselves a basic question that is embodied in a comparison of an ancient Greek philosopher and one of the greatest 20th-century inventors: Diogenes and Thomas Edison. They might appear to have been on two separate paths. However, are these paths in conflict or do they have a common root that makes them complementary?

Diogenes was a Greek philosopher, who was a second-generation disciple of Socrates, one of the greatest seekers of truth in the history of Western civilization. Diogenes has been characterized as an old man with a lantern who used its light to look for truth. In fact, he chose to live a life of poverty on the streets of ancient Greece, sleeping in a large tub. He was probably considered an annoyance because, as a philosophical cynic, he always asked the hard questions that most of us do not want to answer.

The question "What is truth?" is thousands of years old. However, the pursuit of the answer to this question crosses many boundaries. In scientific terms, the answer to this question by Western philosophers, ancient and modern, necessarily leads to the consideration of mathematical logic and the role of models in logic and science, which are discussed in detail in Chapter 2 ("Logical and Scientific Approach") and Chapter 5 ("Architecture Modeling Languages").

Thomas Edison, on the other hand, was a great inventor in modern times. Remembered for inventing the electric light bulb, he became a wealthy and recognized technology leader of his time. His pursuit of "truth" two millennia after Diogenes, in the modern times of technology, could be considered the pursuit of valuable intellectual capital.

He did not live on the streets as did Diogenes, but did he have anything in common with Diogenes? Do you have anything in common with either of these great minds?

At the age of 22, Edison received his first patent: an electronic vote recorder to be used by legislative bodies. It was a simple concept but very advanced for its time. Votes could be recorded by the flip of a switch. However, legislators preferred to cast their votes by voice; Edison could not *sell* the invention. The inventor's response was:

> Anything that won't sell, I don't want to invent. Its sale is a proof of *utility*, and utility is success.

In the years to come, he applied this principle to his engineering practice of invention. He became very wealthy and some would say that he gave up the passion of technology for the pursuit of money. However, others might consider that he valued the utility of his inventions and focused on the utility of the technologies emerging in his day rather than the technologies. It might be considered that he was an early "venture capitalist." His viewpoint coupled with his passion for technology could not help but make him a very wealthy businessman.

Edison himself said that invention was 99 percent perspiration and 1 percent inspiration. It has been said of him that often he was found late at night sleeping on one of the laboratory benches, in between running experiments. Is that so different from sleeping on the streets, as did Diogenes?

Who best fits your personal model? If Diogenes and Edison seem too remote, then consider two contemporary great business leaders, Steve Jobs and Bill Gates. Some might consider that Steve Jobs is technology driven, a revolutionary (e.g., iPod/iTunes) who charges premium prices, and a thought leader who is focused on aesthetics. They might also consider that Bill Gates, on the other hand, is market driven, evolutionary, charges commodity prices, and is a business leader who is more focused on the utility of his products.

It is not a coincidence that the two contemporary leaders were both from the computer industry. This industry has seen dramatic technological growth and generated opportunities for wealth over the past decades, but it has also been the scene of dramatic failures. The evolution of the computer business architecture from the years 1980–2000 saw dramatic, if not devastating, changes for the major computer companies of the time. At the beginning of the 1980s, the industry was organized vertically. Each major company in the computer industry had its own sales and distribution network, application software, operating system, computing platform, and even its own microchips. A scant 20 years later, the industry had been turned on its head. All but one of the major companies that were prosperous in the 1980s had gone the way of the dinosaurs. What happened? Driven by open architecture, the industry became characterized by horizontal integration. The industry had also shifted to a commodity market.

Given the fast cycle of Moore's law, the computer and software industry is necessarily compelled to adapt to change much faster than many other industries. Will the systems engineering practices of the next decade see the same dramatic changes as did the computer and software engineering industry? Shouldn't the systems engineering community be carefully considering what has happened and is happening in the computer and software engineering communities? Does the systems engineering community hold the same narrow views that the dinosaurs of the computer industry held through the 1980s and 1990s?

Multiple views and viewpoints are part of everyday life, but not all of us recognize this perspective. Legitimate but differing points of view abound. If you observe the daily world around you, especially during travel through airports, you will see many signs of the commercial realization of the importance of viewpoints in the form of advertisements. Using pictures of everyday people and objects, challenging questions are asked. For example, what is a full moon? It may be a symbol of lunacy to one person while another may view it as a symbol of romance. Which is the "correct" point of view?

Those readers who are engineers or have a background in science may be thinking that these types of point–counterpoint contrasts belong to nontechnical domains, where regions of grayness reside. In science and engineering, we think that we know what is and what is not. However, serious differences can occur in science and engineering too!

In science the quest for determinism failed in the 20th century with the advent of quantum theory. But Albert Einstein, for example, intellectually struggled against the concepts of quantum theory, saying that he did not believe God played with dice (in the design of the universe). Why did Einstein have a viewpoint that was so different from that of his contemporaries?

In Chapter 3 ("Concepts, Standards, and Terminology"), a few of the myriad concepts and terminologies of architecture and systems engineering will be reviewed. There are over 100 definitions of architecture alone. With so many viewpoints abounding in the precepts of architecture and systems engineering, there is a challenge as to how broadly the concepts and terminology of architecture and systems engineering should be treated in this book. The approach to the material in this book is to keep it simple. There are many points of view. The viewpoints of the authors will focus on just one role of an architect and a commonly practiced role of a systems engineer. Your situation will undoubtedly be different, but the approach presented in this book will give you a way of reasoning and a common thread end to end that you can apply to your own situation. Although this book will provide a standard introduction to the methods and practice of systems architecture, it is important to recognize that new ways of reasoning are more important than the details of practice.

These views are based on the authors' successful professional experiences. The architecture viewpoint is based on personal experience as the Director of Architecture (2000–2003) for the (1st–3rd) Chief Engineer of the U.S. Navy. In this role, the architect was positioned between the acquisition authority (the Assistant Secretary of the Navy) and the government leaders of system development (the system commands). This is not unlike the position that many systems architects find themselves in aerospace and commercial practice.

The systems engineering viewpoint is based on the personal experience of the director of a large academic aerospace systems engineering laboratory at the Georgia Institute of Technology. Often facing ambiguous and ill-defined problems from which clear systems understanding and crisp conclusions must be produced, this role embodies a balancing act between spearheading the dynamic interaction with industry and government customers, and leading the analytical efforts that yield relevant systems knowledge.

However, this book is not just a reiteration of past achievements. It is our view of the path to the future, based on what we know has worked and the power of model-based approaches to architecture and systems engineering.

What is the role of a systems architect in systems engineering? The view taken in this book is succinctly expressed in *The Mythical Man Month* (Brooks 1995):

> Conceptual integrity is the most important consideration in system design. The architect should be responsible for the conceptual integrity of all aspects of the product perceivable by the user.

This view is supported by personal professional experience. It is the basis of this book from the viewpoint of the systems architect.

Key to conceptual integrity are modeling and modeling languages, architecture frameworks, and systems engineering standards. The practice of systems architecture requires communication skills, tools, experience, and knowledge of case studies.

The material in this book has been taught as a one-semester course in Systems Architecture at the M.Sc. and Ph.D. level. However, there is sufficient material to support two semesters of study, one being more focused on systems architecture but with an exposure to software engineering and the other being more focused on aerospace and defense systems engineering. If the material is spread over two semesters, it is recommended that the instructor spend additional time on laboratory practicum so that the students can gain greater proficiency in one or more of the software tools available today for the practice of systems architecture.

Reference

Brooks, F.P. 1995. *The Mythical Man Month.* Boston: Addison Wesley Longman.

Chapter 2

Logical and Scientific Approach

What are the formal languages and methods of systems engineering? The answer to this question is at the heart of the approach described in this chapter. Mathematics, science, and the engineering disciplines each use formal languages, methods, and models. Logic and science provide a sound foundation for a model-based approach to architecture and systems engineering.

Key Concepts

Formal languages and methods
Integrity and consistency of terms
Scientific models
Logical models

Motivation and Background

Modern mathematics, science, and engineering enjoy the benefit of thousands of years of human thought, experience, and practice. And over the past century, with the advent of large-scale systems that today are commonplace, systems engineering has emerged as a distinct engineering discipline, but from any historical perspective, it must be considered still young and maturing. When compared with mathematics, science, and the historical academic disciplines of engineering, we see that

- Mathematics uses
  - Formal logic (the predicate calculus) for description
  - Logical methods and mathematical induction for reasoning

- Science uses
  - Mathematics (a formal language) for description
  - Models and experimental methods for reasoning
- Engineering uses
  - Science and mathematics for description and reasoning
  - Methods, tools, and prototypes for design and decision

The languages, tools, and formalized methods of systems engineering are beginning to emerge. Because the systems viewpoint seeks to reconcile the differences between stakeholders, collaborative systems engineering and design methods are becoming more prevalent. Figure 2.1 depicts the Collaborative Visualization Environment (CoVE) at the Georgia Institute of Technology, which has enjoyed significant success as an academic environment for research in collaborative systems engineering and design. Most large aerospace companies today have collaborative systems engineering environments with substantial visualization capabilities.

But what are the formal languages of systems engineering? Without the precision of a formal language, large-scale tools cannot be commercially developed. Systems engineering has undergone substantial growth over the past decades, but broadly accepted formal languages for systems engineering have not. By comparison, the rise of electronic computation was accompanied by the development of several formal languages, which in recent years have converged in the broadly accepted Unified Modeling Language (UML) and certain computer languages, the standards for which are managed by the Object Management Group (OMG), an international open group. And over the past decade, the OMG Model Driven Architecture (MDA™) has emerged as a new model-based approach to software design and development.

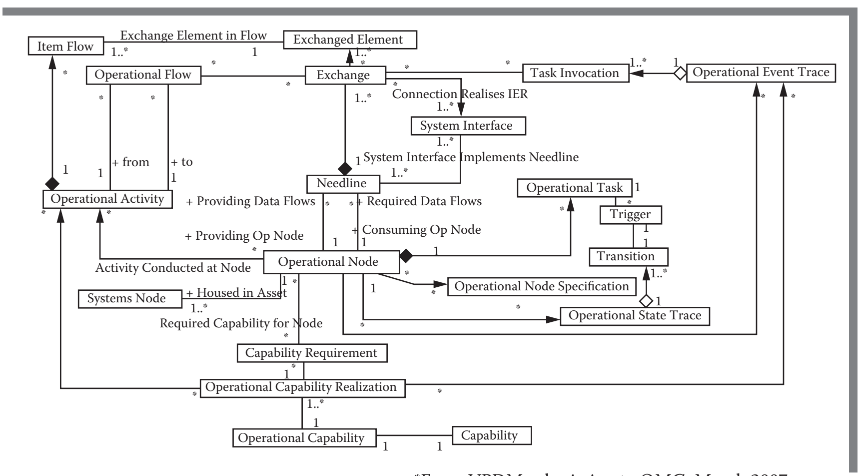
Figure 2.1 The Collaborative Visualization Environment (CoVE).

Therefore, as with the comparison to mathematics, science, and the academic disciplines of engineering made earlier, we see that software engineering indeed does have formal languages, methods, and tools. Specifically:

- Software engineering uses
  - The Unified Modeling Language (UML)
  - Computer-aided software engineering (CASE) tools
- The OMG Model Driven Architecture (MDA) uses
  - UML to create models independent of technology
  - Model transformations for design and development

Efforts to apply UML to systems engineering have been ongoing over the past decade. Figure 2.2 illustrates the meta model for the UML profile for two defense systems architecture frameworks: the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF). The UML Profile for DoDAF and MODAF (UPDM) is a recently approved OMG standard that is a significant step towards establishing architecture standards for defense systems engineering. The UPDM also takes advantage of a recently approved systems engineering extension to UML, which is called the Systems Modeling Language (SysML). Hatley (2000) has also developed a UML diagram, which was a model of his process for requirements engineering and

UPDM Meta Model Based on UML*



*From UPDM submission to OMG, March 2007

Figure 2.2 Modeling and formalizing systems engineering.

system architecture. This model was developed while UML was in its early stages but has been put into practice commercially.

We see then that although systems engineering currently lacks a broadly accepted formal language, it could use UML and the Systems Modeling Language (SysML) and recent advances in architecture frameworks and emerging tools to take steps forward that would better formalize the systems engineering discipline.

Where can all of this be expected to lead the practitioners of systems engineering and the businesses that rely on its practice? This is difficult to say, but given that the world of computer systems has developed at a faster pace than that of large-scale systems, it might be worthwhile to leverage the advances in software engineering over the past decade, especially the use of system models and model transforms.

Will the systems engineering community undergo these types of changes? This is also difficult to say, but when the languages, commercially available tools, and methods of systems engineering are firmly established and broadly accepted, the practice of systems engineering cannot be expected to be the same as it is today. If trends in the software engineering industry, such as UML and MDA, are any indication, then we should expect that systems engineering will become a model-based discipline over the next decade, rather than the document-based discipline that it is today.

Scientific Basis of Engineering

If systems engineering is to be properly understood, then it must be understood in the general context of engineering. What is engineering? The following definition will be used for the purposes of this book:

> *Engineering* is the most primitive level of concept realization where the relationships between function, behavior, and structure for the purpose of solving a problem can be described using the laws of science.

Although there are no doubt many other definitions of this common term, this definition is well suited to the architecture and systems engineering approach taken by this book.

A simple example from structural engineering can be used to illustrate the relationship between science and engineering:

- Problem: How to best use bricks to build a bridge with a planar surface.
- Purpose: The bridge enables transport across a gap in the terrain.
- The underlying physics:
  - Concept of a moment in mechanics.
  - Principle of dispersion of energy in a structure.

- Implications for the engineering of a bridge:
  - The bridge must transfer sufficiently large moments of energy from the plane of transport to the base of the bridge on the ground.
  - If the bricks are arranged in an arch, then the moments of energy will be evenly dispersed through the structure and transferred to the base.
  - Among the benefits of the design is that the number of bricks required is greatly reduced and the resulting structure can be visually attractive.

It has been considered (Finch 1951) that in the evolution of any engineering discipline in the context of Western civilization, the discipline always starts as a craft but when demands for large volumes of production cannot be met by craftsmen, the practice of the craft must evolve. Successful (and profitable!) large-scale production of goods and services relies on science, for its depth of understanding, precision, and repeatability. The honing of a craft using science to achieve commercial development results in the practice of engineering.

This short intuitive description of engineering should not lull the reader into a false sense that good engineering comes easily. Notwithstanding the commercial and organizational problems that can challenge the engineer, reliably understanding and predicting the relationships between function, behavior, and structure is a serious technical problem. The ubiquitous engineering term *emergent behavior* can work both for as well as against the engineer. Murphy's law is always at work.

For those who may consider modern science and engineering to be absolutes, the catastrophic collapse of the Tacoma Narrows Bridge in 1940 stands as one of the great counterexamples (University of Washington 2006). Harmonic oscillations, which are well understood in science and engineering, were the cause of the collapse. More recently, the Millennium Bridge in London suffered a similar design flaw, which fortunately was corrected before a catastrophic event occurred (but after the bridge went into service). See, for example, Arup 2008.

Experimental and Logical Basis of Science

What is science? If engineering in general and systems engineering in particular are to be founded on science, then there needs to be an agreement on what is meant by *science*. The following perspective is offered as a simple aid to understanding the term:

- The Latin word *scientia* means knowledge.
- The English word *science* refers to any systematic body of knowledge, but more commonly refers to one that is based on the scientific method.
- The scientific method consists of
  - Characterization of observables (by definition and measurement).
  - Formulation of hypotheses (i.e., interpretations of models), which are tested by comparing: