

SYSTEMS OF SYSTEMS ENGINEERING

Principles
and Applications

SYSTEMS OF SYSTEMS ENGINEERING

Principles
and Applications

Edited by
Mo Jamshidi



CRC Press
Taylor & Francis Group
Boca Raton London New York

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

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

© 2009 by Taylor & Francis Group, LLC
CRC Press is an imprint of Taylor & Francis Group, an Informa business

No claim to original U.S. Government works
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-13: 978-1-4200-6588-6 (Hardcover)

This book contains information obtained from authentic and highly regarded sources. Reasonable efforts have been made to publish reliable data and information, but the author and publisher cannot assume responsibility for the validity of all materials or the consequences of their use. The authors and publishers have attempted to trace the copyright holders of all material reproduced in this publication and apologize to copyright holders if permission to publish in this form has not been obtained. If any copyright material has not been acknowledged please write and let us know so we may rectify in any future reprint.

Except as permitted under U.S. Copyright Law, no part of this book may be reprinted, reproduced, transmitted, or utilized in any form by any electronic, mechanical, or other means, now known or hereafter invented, including photocopying, microfilming, and recording, or in any information storage or retrieval system, without written permission from the publishers.

For permission to photocopy or use material electronically from this work, please access www.copyright.com (<http://www.copyright.com/>) or contact the Copyright Clearance Center, Inc. (CCC), 222 Rosewood Drive, Danvers, MA 01923, 978-750-8400. CCC is a not-for-profit organization that provides licenses and registration for a variety of users. For organizations that have been granted a photocopy license by the CCC, a separate system of payment has been arranged.

Trademark Notice: Product or corporate names may be trademarks or registered trademarks, and are used only for identification and explanation without intent to infringe.

Library of Congress Cataloging-in-Publication Data

Systems of systems engineering : principles and applications / editor, Mo
Jamshidi. -- 1st ed.

p. cm.

Includes bibliographical references and index.

ISBN 978-1-4200-6588-6 (alk. paper)

1. Systems engineering--Technological innovations. 2. Large scale systems. I.
Jamshidi, Mohammad. II. Title.

TA168.S88854 2008
620.001'171--dc22

2008019820

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

and the CRC Press Web site at
<http://www.crcpress.com>

Dedication

*To my family
Jila, Ava, and Nima*

Contents

About the Editor.....	.ix
Contributors.....	.xi
Chapter 1 Introduction to system of systems	1
<i>Mo Jamshidi</i>	
Chapter 2 SoS architecture	37
<i>Reggie Cole</i>	
Chapter 3 Emergence of SoS, sociocognitive aspects.....	71
<i>Beverly Gay McCarter and Brian E. White</i>	
Chapter 4 A system-of-systems simulation framework and its applications	107
<i>Ferat Sahin, Mo Jamshidi, and Prasanna Sridhar</i>	
Chapter 5 Technology evaluation for system of systems	133
<i>Patrick T. Biltgen</i>	
Chapter 6 Enterprise system of systems	165
<i>George Rebovich, Jr.</i>	
Chapter 7 Definition, classification, and methodological issues of system of systems.....	191
<i>Marcus Bjelkemyr, Daniel T. Semere, and Bengt Lindberg</i>	
Chapter 8 Policymaking to reduce carbon emissions: An application of system-of-systems perspective	207
<i>Datu Buyung Agusdinata, Lars Dittmar, and Daniel DeLaurentis</i>	

Chapter 9 Medical and health management system of systems	233
<i>Yutaka Hata, Syoji Kobashi, and Hiroshi Nakajima</i>	
Chapter 10 The microgrid as a system of systems	251
<i>Laurence R. Phillips</i>	
Chapter 11 An integrated intelligent decision support system based on sensor and computer networks.....	281
<i>Qishi Wu, Mengxia Zhu, Nageswara S. V. Rao, S. Sitharama Iyengar, Richard R. Brooks, and Min Meng</i>	
Chapter 12 Defense applications of SoS	319
<i>Charles E. Dickerson</i>	
Chapter 13 System of air vehicles.....	339
<i>Richard Colgren</i>	
Chapter 14 System of autonomous rovers and their applications	365
<i>Ferat Sahin, Ben Horan, Saeid Nahavandi, Vikraman Raghavan, and Mo Jamshidi</i>	
Chapter 15 Space applications of system of systems.....	385
<i>Dale S. Caffall and James Bret Michael</i>	
Chapter 16 Airport operations: A system-of-systems approach	403
<i>Saeid Nahavandi, Doug Creighton, Michael Johnstone, and Vu T. Le</i>	
Chapter 17 Knowledge amplification by structured expert randomization—KASERs in SoS design.....	421
<i>Stuart H. Rubin</i>	
Chapter 18 System-of-systems standards	451
<i>Mark A. Johnson</i>	

About the Editor

Mo M. Jamshidi (fellow IEEE, fellow ASME, associate fellow AIAA, fellow AAAS, fellow TWAS, fellow NYAS) received the Ph.D. degree in electrical engineering from the University of Illinois at Urbana-Champaign in February 1971. He holds three honorary doctorate degrees from Azerbaijan National University, Baku, Azerbaijan, 1999, University of Waterloo, Canada, and Technical University of Crete, Greece in 2004. Currently, he is Lutcher Brown Endowed Chaired Professor at the University of Texas, San Antonio. He is also the Regents Professor Emeritus of Electrical and Computer Engineering, the AT&T Professor of Manufacturing Engineering and founding director of Center for Autonomous Control Engineering (ACE) at the University of New Mexico, Albuquerque. He has been a consultant and special government employee with U.S. Department of Energy, NASA Headquarters and Jet Propulsion Laboratory, and U.S. Air Force Research Laboratory for a combined 25-year period. He has worked in various academic and industrial positions at various national and international locations including with IBM and GM Corporations. In 1999, he was a NATO Distinguished Professor in Portugal conducting lectures on intelligent systems and control. He has over 600 technical publications including 62 books (12 textbooks) and edited volumes. Six of his books have been translated into at least one foreign language. He is the founding editor or cofounding editor or editor-in-chief of many journals (including Elsevier's *International Journal of Computers and Electrical Engineering*, Elsevier, UK; *Intelligent Automation and Soft Computing*, TSI Press, USA) and one magazine (*IEEE Control Systems Magazine*). He is editor-in-chief of the new *IEEE Systems Journal* (inaugurated in 2007) and coeditor-in-Chief of the *International Journal on Control and Automation*. He has been the general chairman of World Automation Congress (WAC) from its inception. He has been active within the IEEE for 42 years. Dr. Jamshidi is a fellow of the IEEE for contributions to "large-scale systems theory and applications and engineering education," a fellow of the ASME for contributions to "control of robotic and manufacturing systems," fellow of the AAAS (the American Association for the Advancement of Science) for contributions to "complex large-scale systems and their applications to controls and optimization," a fellow of Academy of Developing Nations (Trieste, Italy), member of Russian Academy of Nonlinear Sciences, associate fellow, Hungarian Academy of

Engineering, a fellow of the New York Academy of Sciences, and recipient of the IEEE Centennial Medal and IEEE Control Systems Society Distinguished Member Award and the IEEE CSS Millennium Award. In October 2005 he was awarded the IEEE SMC Society's Norbert Weiner Research Achievement Award, and in October 2006 he received the IEEE SMC Society Outstanding Contribution Award. As an OSU Alumni, he was inducted into Oregon State University's Academy of Distinguished Engineers in February 2007. He founded and has been chair of the IEEE International Conference on System of Systems Engineering since 2006.

Contributors

Datu Buyung Agusdinata received his Ph.D. from Delft University of Technology, the Netherlands, in 2008. He will be with the System-of-Systems Signature Area group at Purdue University, West Lafayette, Indiana. His research interests are in developing and applying quantitative methods for dealing with uncertainty in complex problems in the field of energy, transport, infrastructure, and environment.

Patrick T. Biltgen is a senior principal systems engineer at BAE Systems in Arlington, Virginia. His research focuses on the use of modeling and simulation tools to perform quantitative capability-based analysis of systems-of-systems. He previously held the position of Research Engineer at the Georgia Institute of Technology, where he performed research on the use of agent-based models for technology evaluation. His areas of interest also include systems engineering methods development, aircraft design, and visual analytics.

Marcus Bjelkemyr is a Ph.D. student in production systems engineering at the Royal Institute of Technology (KTH), Stockholm, Sweden. His research focuses on large and complex socio-technical systems, particularly applicability to production systems. In 2004/2005 Marcus Bjelkemyr studied under Professor Nam Suh at MIT (Massachusetts Institute of Technology), Cambridge. In addition to research, Marcus Bjelkemyr also lectures and supervises at the Department of Production Systems at KTH.

Richard R. Brooks is an associate professor of electrical and computer engineering at Clemson University, Clemson, South Carolina, with a B.A. in mathematical science from Johns Hopkins University and Ph.D. in computer science from Louisiana State University. He was head of Pennsylvania State University Applied Research Laboratory Distributed Systems Department and affiliated with Pennsylvania State University IE Department from 1996 to 2004.

Dale S. "Butch" Caffall is director of the National Aeronautic and Space Administration's Independent Verification and Validation Facility, Fairmont,

West Virginia. His area of interest is software engineering, with a particular focus on methods, practices, and tools for software and systems assurance.

Reggie Cole is a principal engineer with the Lockheed Martin Corporation. He has been a program chief architect, program chief engineer, and engineering manager. His areas of interest include enterprise and SoS architecture, analysis of alternatives, capital investment analysis, technology investment analysis, and technology replacement analysis.

Richard Colgren is an associate professor of aerospace engineering and is director of the UAV Laboratory at the University of Kansas in Lawrence, Kansas. Previously he was a senior staff engineer at the Lockheed Martin Aeronautics Company. His areas of interest include flight vehicle conceptual design, development, and flight testing and the development and testing of their autonomous and robust flight control systems.

Doug Creighton received his B.Sc. in physics and B.E. in systems engineering from the Australian National University, and a Ph.D. in simulation-based optimization from Deakin University in Australia. He is currently employed as a post-doctoral research fellow and leads the process modeling and analysis team. His primary areas of research include simulation-based optimization, agent-based architectures, visualization methodologies, augmented reality, haptics technologies, and discrete event simulation methodologies, tools, and applications.

Daniel DeLaurentis is assistant professor of aeronautics and astronautics at Purdue University, West Lafayette, Indiana, joining the University in 2004 under the System-of-Systems Signature Area. His areas of interests are system of systems modeling and analysis methodologies and advanced design techniques applied to air/space transportation systems and civil infrastructure systems.

Charles E. Dickerson is chair of systems engineering at Loughborough University and previously was a technical fellow at BAE Systems. Before joining BAE, he was at MIT Lincoln Laboratory and he was the director of architecture for the chief engineer of the U.S. Navy. He is a signatory to the IEEE ICSOS and chair of the INCOSE Architecture Working Group. His aerospace experience includes air vehicle survivability and design at the Lockheed Skunk Works and Northrop's Advanced Systems Division, and operations analysis at the Center for Naval Analyses. He received his Ph.D. from Purdue University in 1980.

Lars Dittmar is affiliated with the Department of Science, Technology and Society at Utrecht University in the Netherlands and works at the Energy Systems department of Berlin University of Technology, Germany. His research

interests cover the whole spectrum of energy-economic modeling, energy policy, and sustainable development.

Yutaka Hata is a professor of Division of Computer Engineering, Graduate School of Engineering, University of Hyogo, Japan. His areas of interests are medical imaging, ultrasonic systems, and biosensor systems with fuzzy logic technology. He is also a visiting professor of Immunology Frontier Research Center, Osaka University, Japan.

Ben Horan is currently a lecturer in electronics and robotics at Deakin University, Australia, and is in the final stages of his Ph.D. in the haptics and robotics research area. Ben spent 2006 at the ACE Center at UTSA with Professor Mo Jamshidi. His research interests include haptics, haptic teleoperation, mobile robotics, and immersive operator interfaces.

S. Sitharama Iyengar is the chairman and Roy Paul Daniels Chaired Professor of computer science at Louisiana State University, Baton Rouge, and is also the Satish Dhawan Chaired Professor at the Indian Institute of Science, Bangalore. His publications include six textbooks, five edited books, and over 380 research papers. His research interests include high-performance algorithms, data structures, sensor fusion, data mining, and intelligent systems. He is a world-class expert in computational aspects of sensor networks.

Mo Jamshidi is Lutcher Brown Endowed Chaired Professor of Electrical and Computer Engineering, University of Texas, San Antonio. His areas of interests are system of systems simulation, architecture, and control with application to land, sea, and air rovers.

Mark A. Johnson is a senior project engineer with the Aerospace Corporation. He received his Ph.D. from the University of New Mexico in 2002. He has over 37 years of experience in space technology; communications and cryptographic systems; systems engineering; research and development; modeling; simulation and analysis; acquisition; operations; and cost and risk management. He has extensive experience working with NASA, the U.S. Air Force, and Aerospace Corporation.

Michael Johnstone received his B.E. (Hons) from Deakin University (Australia) and is currently a post-graduate student with Deakin's School of Engineering and IT. His research is directed toward simulation and control of complex networks, aiming toward the creation of algorithms to determine efficient flows through the networks under varying operational conditions. He has experience in varied simulation studies, baggage handling systems, logistics, and warehousing, and in all phases in the management of a simulation study.

Syoji Kobashi is an associate professor of the Graduate School of Engineering, University of Hyogo, Hyogo, Japan. His areas of interests are computer-aided diagnosis system (CAD) in medicine and medical image and signal processing. He has proposed many CAD systems for diagnosing brain, lung, pancreatic duct, hip joint, knee joint, etc.

Vu T. Le received a B.E. degree (Hons) in mechanical engineering from Royal Melbourne Institute of Technology (Australia), and an M.E. in production planning and scheduling from Deakin University (Australia). He is currently a Ph.D. research student in complex network system analysis at Deakin University. His research interests include scheduling, complex system modeling, discrete event simulation, and optimization of manufacturing and material handling systems.

Bengt Lindberg has been a professor in production systems engineering since 2000, and dean of the School of Industrial Engineering and Management at the Royal Institute of Technology (KTH), Stockholm, Sweden, since 2005. His research area includes manufacturing system configuration design, digital projecting, as well as manufacturing processes and equipment. Bengt Lindberg has 15 years of experience at the Scania Truck Company. His industrial career covers responsibilities from production engineering and engine production to development tools for the product realization processes.

Beverly Gay McCarter has an M.S. degree in counseling psychology and human systems from Florida State University and an M.F.A. degree in fine art. She is certified in self-organizing systems for complex environments focusing on human dynamics and consciousness, and is Principal, Human Mosaic Systems, LLC. Recent clients have included the Smithsonian Institution and Immigration and Customs Enforcement.

Min Meng is a Ph.D. student with the Department of Computer Science at The University of Memphis. Her main research interest is in sensor networks.

James Bret Michael is a professor of computer science and electrical and computer engineering at the Naval Postgraduate School, Monterey, California. His area of interest is methods for attaining high levels of dependability and trust in software-intensive mission- and safety-critical systems.

Saeid Nahavandi received a B.Sc. (Hons), M.Sc., and Ph.D. in automation and control from Durham University (U.K.). Professor Nahavandi holds the title of Alfred Deakin Professor, Chair of Engineering, and is the leader for intelligent systems research at Deakin University (Australia). His research interests include modeling of complex systems, simulation-based optimization, robotics, haptics, and augmented reality.

Hiroshi Nakajima is a senior advisory technology researcher at OMRON Corporation, Kyoto, Japan. His areas of interests are causality-based modeling methods, human-machine collaborative systems, and smart management systems for healthcare, energy consumption, and machine maintenance.

Laurence R. Phillips is a principal member of the technical staff at Sandia National Laboratories. His current research interest is analysis of risk due to cyber attack for critical infrastructures. Other areas of interest include artificial intelligence and agent-based microgrid operation.

Vikraman Raghavan is an engineer with Houston Oil Company in Houston, Texas. He received his B.S. degree in electronic technology from Chennai, India, and his M.S. degree from the Department of Electrical and Computer Engineering at the University of Texas, San Antonio, in June 2007.

Nageswara S. V. Rao is a corporate fellow at Oak Ridge National Laboratory in Computer Science and Mathematics Division. His research interests include high-performance networking, sensor networks, and information fusion.

George Rebovich, Jr., is a senior principal engineer in the MITRE Corporation's Command & Control Center. He has held a variety of systems engineering positions at MITRE that include leading technical departments and groups, direct sponsor projects, and technology projects and serving as chief system engineer for a C4I System program.

Stuart H. Rubin is a senior scientist in the Information, Surveillance, and Reconnaissance (ISR) Department and director of the Knowledge Amplification by Structured Expert Randomization (KASER) Project, SSC Pacific, San Diego. His areas of interest include computational creativity and knowledge-discovery systems, emphasizing heuristic, evolutionary, fuzzy, and randomization-based methodologies.

Ferat Sahin is associate professor of electrical engineering, Rochester Institute of Technology, Rochester, New York. His areas of interests are swarm robotics, multiagent systems, system of systems simulation for autonomous rovers, and MEMS-based microrobots.

Daniel T. Semere is a senior researcher and lecturer in the Department of Production Systems Engineering at the Royal Institute of Technology (KTH), Stockholm, Sweden. His research focuses on distributed systems architecture with particular emphasis to autonomous distributed systems (ADS). He has a number of publications on this and related fields. His biography is also included in Marquis Who's Who 2007 edition. In addition to his research, Dr. Semere also has lectured and supervised thesis projects in the production systems graduate program at the Royal Institute of Technology since 2005.

Prasanna Sridhar received his B.E. in computer science and engineering from Bangalore University, India, in 2000, his M.S. degree in computer science in 2003, and his Ph.D. in computer engineering in 2007, both from the University of New Mexico. In 2006, he joined the University of Texas at San Antonio as a research scientist assistant. His current research interests are embedded sensor networks, mobile robotics, modeling, and simulation, and computational intelligence. Currently, he is with Microsoft Corp.

Brian E. White received his Ph.D. from the University of Wisconsin and electrical engineering degrees from M.I.T. He worked at Lincoln Laboratory and Signaltron, Inc. In his 26+ years at The MITRE Corporation, he's had senior technical staff and project/resource management positions, most recently as director of MITRE's Systems Engineering Process Office.

Qishi Wu is an assistant professor with the Department of Computer Science at the University of Memphis. His research interests include sensor networks, computer networking, scientific visualization, and distributed high-performance computing.

Mengxia Zhu is an assistant professor with the Computer Science Department at Southern Illinois University, Carbondale. Her research interests include distributed and high-performance computing, bioinformatics, and distributed sensor networks.

chapter one

Introduction to system of systems

Mo Jamshidi

Contents

1.1	Introduction.....	2
1.2	System of systems definitions.....	3
1.3	Problems in system of systems.....	4
1.3.1	Theoretical problems	4
1.3.1.1	Architecting system-of-systems solutions	4
1.3.1.2	Emergence of SoS, sociocognitive aspects.....	5
1.3.1.3	SoS simulation	6
1.3.1.4	Enterprise systems of systems	7
1.3.1.5	Definition, classification, and methodological issues of system of systems	8
1.3.2	Implementation problems	8
1.3.2.1	Policymaking to reduce carbon emissions.....	8
1.3.2.2	Medical and health management system of systems.....	9
1.3.2.3	The microgrid as a system of systems	9
1.3.2.4	Intelligent decision support system based on sensor and computer networks	10
1.3.2.5	Defense applications of SoS.....	11
1.3.2.6	Systems of air vehicles	11
1.3.2.7	System of autonomous rovers and their applications.....	12
1.3.2.8	Space applications of system of systems	12
1.3.2.9	Airport operations: a system-of-systems approach	13
1.3.2.10	KASERS in SoS design	13
1.3.2.11	System-of-systems standards.....	14
1.4	Other SoSE issues	15
1.4.1	Open systems approach to system of systems engineering.....	15
1.4.2	SoS integration.....	16
1.4.3	Engineering of SoS.....	16
1.4.4	SoS management: the governance of paradox	17

1.4.5	Deepwater coastguard program	18
1.4.6	Future combat missions.....	19
1.4.7	Systems engineering for the Department of Defense system of systems	20
1.4.8	Sensor networks	20
1.4.9	Healthcare systems	20
1.4.10	Global Earth Observation System of Systems	21
1.4.11	E-enabling and SoS aircraft design via SoSE	22
1.4.12	A system-of-systems perspective on infrastructures.....	24
1.4.13	A system-of-systems view of services	25
1.4.14	System of systems engineering in space exploration	25
1.4.15	Robotic swarms as an SoS.....	26
1.4.16	Communication and navigation in space SoS.....	26
1.4.17	National security	27
1.4.18	Electric power systems grids as SoS.....	28
1.4.19	SoS approach for renewable energy	29
1.4.20	Sustainable environmental management from a system of systems engineering perspective.....	29
1.4.21	Transportation systems	30
1.4.22	System of land, sea, and air vehicles as SoS	31
1.5	Conclusions	32
	References	32

1.1 Introduction

Recently, there has been a growing interest in a class of complex systems whose constituents are themselves complex. Performance optimization, robustness, and reliability among an emerging group of heterogeneous systems in order to realize a common goal has become the focus of various applications including military, security, aerospace, space, manufacturing, service industry, environmental systems, and disaster management, to name a few [Crossley, 2006; Lopez, 2006; Wojcik and Hoffman, 2006]. There is an increasing interest in achieving synergy between these independent systems to achieve the desired overall system performance [Azarnoosh et al. 2006]. In the literature, researchers have addressed the issue of coordination and interoperability in a system of systems (SoS) [Abel and Sukkarieh, 2006; DiMario, 2006]. SoS technology is believed to more effectively implement and analyze large, complex, independent, and *heterogeneous* systems working (or made to work) cooperatively [Abel and Sukkarieh, 2006]. The main thrust behind the desire to view the systems as an SoS is to obtain higher capabilities and performance than would be possible with a traditional system view. The SoS concept presents a high-level viewpoint and explains the interactions between each of the independent systems. However, the SoS concept is still at its developing stages [Meilich, 2006; Abbott, 2006].

The next section will present some definitions out of many possible definitions of SoS. However, a practical definition may be that a system of systems is a super system comprised of other elements which themselves are independent complex operational systems and interact among themselves to achieve a common goal. Each element of an SoS achieves well-substantiated goals even if they are detached from the rest of the SoS. For example a Boeing 747 airplane, as an element of an SoS, is not SoS, but an airport is an SoS, or a rover on Mars is not an SoS, but a robotic colony (or a robotic swarm) exploring the red planet, or any other place, is an SoS. As will be illustrated shortly, associated with SoS, there are numerous problems and open-ended issues which need a great deal of fundamental advances in theory and verifications. It is hoped that this volume will be a first effort toward bridging the gaps between an *idea* and a *practice*.

1.2 System of systems definitions

Based on the literature survey on system of systems, there are numerous definitions whose detailed discussion is beyond the space allotted to this chapter [Jamshidi, 2005; Sage and Cuppen, 2001; Kotov, 1997; Carlock and Fenton, 2001; Pei, 2000; Luskasik, 1998]. Here we enumerate only six of many potential definitions:

Definition 1: Enterprise systems of systems engineering (SoSE) is focused on coupling traditional systems engineering activities with enterprise activities of strategic planning and investment analysis [Carlock and Fenton, 2001].

Definition 2: System-of-systems integration is a method to pursue development, integration, interoperability, and optimization of systems to enhance performance in future battlefield scenarios [Pei, 2000].

Definition 3: Systems of systems exist when there is a presence of a majority of the following five characteristics: operational and managerial independence, geographic distribution, emergent behavior, and evolutionary development [Jamshidi, 2005].

Definition 4: Systems of systems are large-scale concurrent and distributed systems that are comprised of complex systems [Jamshidi, 2005; Carlock and Fenton, 2001].

Definition 5: In relation to joint war-fighting, system of systems is concerned with interoperability and synergism of Command, Control, Computers, Communications, and Information (C4I) and Intelligence, Surveillance, and Reconnaissance (ISR) Systems [Manthorpe, 1996].

Definition 6: SoSE involves the integration of systems into systems of systems that ultimately contribute to evolution of the social infrastructure [Luskasik, 1998].

Detailed literature survey and discussions on these definitions are given in [Jamshidi, 2005, 2008]. Various definitions of SoS have their own merits, depending on their application.

The favorite definition of this author and the volume's editor is systems of systems are large-scale integrated systems which are heterogeneous and independently operable on their own, but are networked together for a common goal. The goal, as mentioned before, may be cost, performance, robustness, etc.

1.3 Problems in system of systems

In the realm of open problems in SoS, just about anywhere one touches, there is an unsolved problem, and immense attention is needed by many engineers and scientists. No engineering field is more urgently needed in tackling SoS problems than SE—system engineering. On top of the list of engineering issues in SoS is the engineering of SoS, leading to a new field of SoSE [Wells and Sage, 2008]. How does one extend SE concepts like analysis, control, estimation, design, modeling, controllability, observability, stability, filtering, simulation, etc. so that they can be applied to SoS? Among numerous open questions are how one can model and simulate such systems (see Chapter 4 by Sahin et al.). In almost all cases a chapter in this volume will accommodate the topic raised. Additional references are given to enhance a search by the interested reader.

1.3.1 Theoretical problems

In this section a number of urgent problems facing SoS and SoSE are discussed. The major issue here is that a merger between SoS and engineering needs to be made. In other words, systems engineering (SE) needs to undergo a number of innovative changes to accommodate and encompass SoS.

1.3.1.1 Architecting system-of-systems solutions

Cole, in Chapter 2, presents solution ideas for SoS architectures. The author indicates that, along with defining the problem, one of the most important jobs of the systems engineer is to partition the problem into smaller, more manageable problems and make critical decisions about the solution. One of the most critical decisions is the architecture—the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution [ANSI/IEEE-1472, 2000]. While it is impossible to understand all the characteristics and consequences of the architecture at the time the system is designed, it is possible to produce a system architecture that maximizes the ability of the system to meet user needs while minimizing the unintentional consequences.

In one sense, architecting a complex system that is comprised of a number of collaborating independent systems is no different than designing a simple

system. Both start with definition of a problem and conception of solution. Both warrant consideration of environmental context. Both involve analysis of factors related to effectiveness. And both require design compromises and balancing of competing priorities. The basic process is the same. In fact, it has been well documented for nearly 50 years [Asimow, 1962]. But compared to the design of simple systems, the complexity of designing a system-of-systems (SoS) solution is daunting, and the design process must be approached with that complexity in mind. Details are found here in Chapter 2.

Dagli and Ergin [2008] also provide a framework for SoS architectures. As the world is moving toward a networked society, the authors assert, the business and government applications require integrated systems that exhibit intelligent behavior. The dynamically changing environmental and operational conditions necessitate a need for system architectures that will be effective for the duration of the mission but evolve to new system architectures as the mission changes. This new challenging demand has led to a new operational style; instead of designing or subcontracting systems from scratch, business or government gets the best systems the industry develops and focuses on becoming the lead system integrator to provide a system of systems (SoS). SoS is a set of interdependent systems that are related or connected to provide a common mission. In the SoS environment, architectural constraints imposed by existing systems have a major effect on the system capabilities, requirements, and behavior. This fact is important, as it complicates the systems architecting activities. Hence, architecture becomes a dominating but confusing concept in capability development. There is a need to push system architecting research to meet the challenges imposed by new demands of the SoS environment. This chapter focuses on system-of-systems architecting in terms of creating meta-architectures from collections of different systems. Several examples are provided to clarify system-of-systems architecting concepts. Since the technology base, organizational needs, and human needs are changing, the system-of-system architecting becomes an evolutionary process. Components and functions are added, removed, and modified as owners of the SoS experience and use the system. Thus an evolutionary system architecting is described, and the challenges are identified for this process. Finally, the authors discuss the possible use of artificial life tools for the design and architecting of SoS. Artificial life tools such as swarm intelligence, evolutionary computation, and multiagent systems have been successfully used for the analysis of complex adaptive systems. The potential use of these tools for SoS analysis and architecting is discussed, by the authors, using several domain-application-specific examples.

1.3.1.2 *Emergence of SoS, sociocognitive aspects*

McCarter and White, in Chapter 3, offer a human-centric treatment of the concepts of multiscale analysis and emergence in system of systems (SoS) engineering, or more generally, complex systems engineering (CSE) [Gladwell, 2005]. This includes a characterization of what an individual might do in

conceptualizing a given systems engineering situation. The authors suggest fresh interpretations of the terms *scale* and *emergence* that will contribute to a more collaborative approach to improving the CSE practice. Because other authors use “scale” in several different ways, the authors propose “view” instead. Here a given view is defined as a combination of “scope,” “granularity,” “mindset,” and “timeframe.” While “emergence” has a rich spectrum of definitions in the literature, the authors prefer to emphasize the unexpected, especially “surprising,” flavor of emergence. Sociocognitive aspects are paramount, and small group dynamics are becoming more significant. Human behaviors that impact information sharing, productivity, and system interoperability need closer examination as organizations increasingly rely on decentralization. Therefore, this chapter also highlights important dynamic social issues, although the authors do not focus on trying to solve them. Rather, areas of further research are suggested which can lead to better CSE, when people are considered part of any (complex) system in the SoS.

1.3.1.3 SoS simulation

Sahin et al., in Chapter 4, provide a framework for simulation of SoS. They have presented an SoS architecture based on extensible markup language (XML) in order to wrap data coming from different systems in a common way. The XML can be used to describe each component of the SoS and their data in a unifying way. If XML-based data architecture is used in an SoS, the only requirement is for the SoS components to understand/parse XML file received from the components of the SoS. In XML, data can be represented in addition to the properties of the data such as source name, data type, importance of the data, and so on. Thus, it not only represents data but also gives useful information which can be used in the SoS to take better actions and to understand the situation better. The XML language has a hierarchical structure where an environment can be described with a standard and without a huge overhead. Each entity can be defined by the user in the XML in terms of its visualization and functionality. As a case study in this effort (see also [Mittal et al., 2008]) a master-scout rover combination represents an SoS where first a sensor detects a fire in a field. The fire is detected by the master rover, which commands the scout rover to verify the existence of the fire. It is important to note that such an architecture and simulation does not need any mathematical model for members of the systems.

Biltgen, in Chapter 5, provides a fundamental view of technology and its evaluation for SoS. The author notes that technology evaluation is the assessment of the relative benefit of a proposed technology for a particular purpose with respect to one or more metrics. While a number of quantitative assessment techniques have been developed to support engineering and design of *systems*, a flexible, traceable, rapid method for technology

evaluation for systems of systems is needed. The primary difficulty in the assessment process is the complexity associated with large-scale systems of systems and the integrated nature of models required to study complex phenomena. The author goes on to note that many qualitative techniques exist for resource allocation; an approach based on modeling and simulation is usually required to quantify technology potential with respect to measures of effectiveness (MoEs) at the systems-of-systems level. The modeling and simulation environment relies on a linked suite of variable fidelity models that calculate the impact of technologies across a system-of-systems hierarchy. Unfortunately, the run time of this suite of tools is often prohibitive for exploratory analysis and domain-spanning optimization applications. In this chapter, the concept of surrogate models is introduced to speed up the analysis process, provide a measure of transparency to the underlying physical phenomena, and enable the execution of design of experiments across a range of technology parameters at all levels of the system-of-systems hierarchy. While polynomial surrogate models have exhibited much promise in systems-design applications, neural network surrogates are introduced in this chapter due to their ability to capture the nonlinearities and discontinuities often present in systems-of-systems problems. Techniques for data farming and visualization of results are also introduced as a means to understand the intricacies of complex design spaces, and an approach to inverse design, where any variable can be treated as an independent variable, is demonstrated. The inverse design technique allows the user to set thresholds for capability-based MoEs and decompose the problem to identify the critical technology factors that are most significant to the responses in question. Finally, insight is provided into the value of the surrogate-based approach for understanding sensitivities and quantifying the relative benefit of proposed technologies with respect to the appropriate MoEs at the system-of-systems level.

1.3.1.4 *Enterprise systems of systems*

Rebovich, in Chapter 6, takes on the enterprise system of systems engineering. The author notes that the twenty-first century is an exciting time for the field of systems engineering. Advances in our understanding of the traditional discipline are being made. At the same time, new modes of systems engineering are emerging to address the engineering challenges of systems of systems and enterprise systems. Even at this early point in their evolution, these new modes of systems engineering are evincing their own principles, processes and practices. Some are different in degree than engineering at the system level, while others are different in kind.

While it is impossible to predict how the traditional and new forms of systems engineering will evolve in the future, it is clear even now that there is a long and robust future for all three. Increases in technology complexity

have led to new challenges in architecture, networks, hardware and software engineering, and human systems integration. At the same time, the scale at which systems are engineered is exceeding levels that could have been imagined only a short time ago. As a consequence, all three forms of systems engineering will be needed to solve the engineering problems of the future, sometimes separately but increasingly in combination with each other.

This chapter defines three modes of systems engineering, discusses the challenge space each addresses, describes how they differ from and complement each other, and suggests what their interrelationships should be in solving engineering problems of the future.

1.3.1.5 Definition, classification, and methodological issues of system of systems

Bjelkemyr et al., in Chapter 7, present a number of basic issues in SoS, similar to those discussed in this chapter. They present a generic definition of the term system of systems. This definition is based both on common definitions of SoS and on the characteristics that systems that are usually labeled SoS exhibit. Due to the inclusive nature of this generic definition, a wide range of sociotechnical systems of very different size and complexity are included in the concept of SoS. In an attempt to delimit the concept, a classification is suggested based on the redundancy of higher-level subsystems. Most systems that traditionally are considered SoS are labeled *SoS type I* (e.g., International Space Station, an integrated defense system, national power supply networks, transportation systems, larger infrastructure constructions), and systems with nonredundant subsystems are simply labeled *SoS type II*, in this chapter exemplified by a production system. The purpose of this classification is partly to advance knowledge of both SoS characteristics and how to address them, and partly to improve transferring of this knowledge from the area of traditional SoS to other areas where SoS characteristics are present.

1.3.2 Implementation problems

Aside from many theoretical and essential difficulties with SoS, there are many implementation challenges facing SoS. Here some of these implementation problems are briefly discussed, and references are made to some with their full coverage.

1.3.2.1 Policymaking to reduce carbon emissions

Agusdinata et al., in Chapter 8, present a system-of-systems perspective combined with exploratory modeling and analysis method as an approach to deal with complexity and uncertainty in policymaking. The test case is reduction of CO₂ (carbon) emissions, which stems from the interactions of many independent players involved and the multiple aspects. The uncertainty about the physical and socioeconomic systems is compounded by the

long time horizon the policymaking covers. Using a case of the long-term carbon emissions in Dutch residential sector, the SoS perspective is used as a way to decompose the policy issue into interdependent relevant policy systems. This representation of the policy system provides a framework to test a large number of hypotheses about the evolution of the system's performance by way of computational experiments, which forms the core of the exploratory modeling and analysis (EMA) method. In particular, in a situation where the realized emission level misses the intermediate target in the year 2025, policies can be adapted, among others, by increasing the subsidy on energy efficiency measures and tightening the standard for dwelling energy performance. As some of the system uncertainties are resolved, we test whether adapting policies can lead to meeting the policy target in the year 2050. Our approach shows the extent to which the constraints imposed on the system of systems should be relaxed in order to bring the emission level back to the intended target. The relaxation of the constraints, which include among others energy prices, investment behavior, demographic, and technological developments also point to the different policy designs that decision makers can envision to influence the performance of a system of systems.

1.3.2.2 *Medical and health management system of systems*

In Chapter 9, Hata et al. have bridged a gap between medical systems and SoS. Such a system is widely used in diagnosis, treatment, and management for patients. Human health management for healthy persons has considerable attention as a new application domain in system engineering. In this chapter, we focus on ultrasonic surgery support, medical imaging, and health management system of systems engineering. The application domains discussed here are indeed broad and essential in daily clinical practice and health management. The first one is systems of systems in medical ultrasonics. Current modern ultrasonic systems require an integrated system of systems engineering, i.e., integrated hardware system and modern software system. This section introduces a novel ultrasonic surgery support system in orthopedic surgery. Second is system of systems in medical imaging. This section introduces an approach to embed medical expert knowledge into image processing based on fuzzy logic. To demonstrate the effectiveness of the new approach, applications to human brain MR images and orthopedic kinematic analysis are introduced. Third is system of systems in human health management. An idea of health management is introduced and discussed from the view of system of systems. As its application to human health, sensing and control technology during sleep is focused on because quality and quantity of sleep has serious influence on our body health.

1.3.2.3 *The microgrid as a system of systems*

Phillips, in Chapter 10, has introduced electric microgrids as an SoS. The author points out that microgrids offer exciting possibilities for efficient,

uninterruptible power. A microgrid is a collection of small, non-collocated electric power sources, storage devices, and power conditioners interconnected to meet the power consumption needs of a designated community. The general vision is that a microgrid might produce power for a small to midsize neighborhood, industrial park, or commercial enclave. A microgrid is different from a power plant primarily because the generators are not collocated, so it cannot easily be operated, managed, or controlled in a unified way by a single committee. At the same time, a microgrid, because of the interconnectivity, can be operated more efficiently than independent sources of similar capacity; more of the sources at any given point can be operated at peak efficiency, transmission losses are minimized, and—perhaps the most enticing thought—significant coproduced heat can be used for space and water heating and a myriad of industrial uses because the sources are near the loads. All well and good, but if this is such a great idea, where are all the microgrids? Unfortunately, fielding a microgrid requires a relatively complex system of systems, only part of which is the already complicated electric power substrate. In addition, to have significant impact, microgrids need to operate in conjunction with the primary power grid; this adds to operational difficulty and forces economic considerations to be taken into account operationally. In this chapter we delineate and discuss the communication, management, decision making, and other systems needed for microgrid success.

1.3.2.4 Intelligent decision support system based on sensor and computer networks

Wu et al., in Chapter 11, have presented an intelligent decision support system based on sensor and computer networks that incorporates various subsystems for sensor deployment, data routing, distributed computing, and information fusion. The integrated system is deployed in a distributed environment composed of both wireless sensor networks for data collection and wired computer networks for data processing. For these subsystems, we formulate the analytical problems and develop approximate or exact solutions: (1) sensor deployment strategy based on a two-dimensional genetic algorithm to achieve maximum coverage with cost constraints; (2) data routing scheme to achieve maximum signal strength with minimum path loss, energy efficiency, and fault tolerance; (3) network mapping method based on dynamic programming to assign computing modules to network nodes for distributed sensor data processing; and (4) binary decision fusion rule that derive threshold bounds to improve system hit rate and false alarm rate. These subsystems are implemented and evaluated through either experiments or simulations in various application scenarios. The extensive results demonstrate that these component solutions imbue the integrated system with the desirable and useful quality of intelligence.

1.3.2.5 Defense applications of SoS

Dickerson, in Chapter 12, takes on the defense applications of SoS. The author notes that the defense community continues to move toward increasingly complex weapon systems that must support joint and coalition operations, and the need for system of systems (SoS) engineering becomes more critical to the achievement of military capabilities. The U.S. Department of Defense (DoD) and the U.K. Ministry of Defence (MOD) continue to face a critical challenge: the integration of multiple capabilities across developing and often disparate legacy systems that must support multiple warfare areas. Over the past decade, the need for SoS-enabled mission capabilities has led to changes in policy for the acquisition of systems and the assemblage of battle forces. Defense acquisition structures have been reorganizing in order to integrate acquisition activities in a way that leads to the achievement of capabilities through a system-of-systems approach rather than from just the performance of individual systems or platforms. Earlier efforts to meet the challenge faced by the defense community have included solutions using network centric operations (NCO) and Network Enabled Capabilities (NEC). Although progress has been made, future progress can be expected to depend upon the advancement of the practice of systems engineering at the SoS level. System of systems engineering (SoSE) processes and practices must enable the acquisition of systems of systems that support the integration of multiple capabilities across multiple warfare areas. Recently there has been extensive debate across a broad community of government, industry, and academic stakeholders about systems engineering processes and practices at the SoS level. But whatever form SoSE takes for defense systems, the technical processes of SoSE must support the realization of mission capabilities attainable from a system of systems that cannot be attained from a single system. Chapter 12 will focus on SoS initiatives and SoSE technical processes being used for the development and acquisition of defense systems of systems. It will begin with the revolution in military affairs that led to network-centric warfare then show how the concepts of SoSE and network enablement are intertwined. Together they not only enable mission capabilities but also are key to the integration of capabilities. Key SoS initiatives in the DoD are discussed. Chapter 12 will conclude with some of the current activities and emerging research opportunities in SoSE.

1.3.2.6 Systems of air vehicles

Colgren, in Chapter 13, has considered a system of air vehicles as a natural test bed for an SoS. The author reminds us that a longstanding example of such a system, developed long before the term system of systems was coined, is the international airspace system intended for the safe and efficient flight control of air vehicles. The International Civil Aviation Organization (ICAO), which manages the international air traffic control (ATC) system, has been

in operation since April 1947. It was built around preexisting international agreements and regulations as an agency of the United Nations (Ref. 1). It codifies the principles and techniques of international air navigation and fosters the planning and development of international air transport to ensure safe and orderly growth (Ref. 2). It insures standards, such as the international use of English for communications and the specific frequency ranges to be used for these and all other required command and control operations. The ICAO Council adopts standards and recommended practices for its 190 member states concerning air navigation, the prevention of unlawful interference, and for facilitating border crossings for international civil aviation. In addition, the ICAO defines the protocols for air accident investigations used by transport safety authorities in the signatory countries, also commonly known as the Chicago Convention. The United States air traffic control system is managed by the Federal Aviation Administration (FAA) to provide safe separation of aircraft within U.S. airspace and in and out of U.S. airports (Ref. 3). While the basic design of the air traffic control system dates back to the 1950s, this design has been adapted as the demands on the system's capacity have continued to rise. The three critical components of the ATC—communications, navigation, and surveillance—must continuously be modernized to maintain the safety and efficiency of the air transportation system. Such updates must be accomplished under international agreements, maintaining global compatibility throughout the ATC system.

1.3.2.7 System of autonomous rovers and their applications

Sahin et al., in Chapter 14, present a typical test bed for system of systems in the realm of mobile robotics. Here a system of autonomous rovers will be presented in the context of system of systems. In addition, a system of homogenous modular micro robots will be presented in the context of system of systems. The chapter starts with the introduction of the components and their roles in the system of autonomous rovers. Then, each system will be presented focusing on electrical, mechanical, and control characteristics and their capabilities in the system of autonomous rovers. Robust data aggregation and mine detection are then examined as applications of the system of autonomous rovers.

1.3.2.8 Space applications of system of systems

Caffall and Michael, in Chapter 15, have started by referring to future applications of SoS via the popular fictional space program on television—"Space . . . the Final Frontier. These are the voyages of the starship Enterprise. Its five-year mission: to explore strange new worlds, to seek out new life and new civilizations, to boldly go where no man has gone before." The authors ask the question: how many of us heard these words and dreamed of standing alongside Star Trek's Captain James T. Kirk and First Officer Mr. Spock to explore new worlds? Interesting enough, Chief Engineer Commander Scott performs countless technological miracles as he saves

his beloved USS *Enterprise* and crew from certain destruction; however, Scotty never mentions the miracle of the *Enterprise* as a wonderful system of systems. Could it be that future engineers solved all the problems of system of systems? Could it be that Scotty never encountered problems with the system of systems comprising the *Enterprise*, freeing him up to devote his time on advancing the interplanetary communications, warp engine, phaser, transporter, deflector shield, and other systems of the *Enterprise*? Apparently not, at least as evidenced by, for instance, the emergent behavior of the *Enterprise* as it interacts with a (fictitious) legacy unmanned scientific probe named *Voyager 6* whose onboard systems have been reconfigured and augmented by an alien race. The authors further note that it is up to us to advance the system-of-systems technology for the development of a space foundation that, indeed, may lead to the Federation and star ships. While many folks almost exclusively think of space exploration as the Shuttle orbiter that has captured worldwide interest for over twenty-five years, space exploration is multifaceted. The National Aeronautics and Space Administration (NASA) has a extraordinary history of space exploration to include the Rovers that continue to explore Mars, space telescopes that can see into deep space, a human-inhabitable space station that orbits the Earth, unmanned spacecraft that explore the Sun and the planets in our solar system, and the unforgettable lunar exploration that began on July 20, 1969, as Commander Neil Armstrong stepped onto the Moon's surface. So, what is the relationship of space exploration with a system-of-systems concept? Well, let us first consider space exploration missions that are planned for the next couple of decades.

1.3.2.9 *Airport operations: a system-of-systems approach*

Nahavandi et al., in Chapter 16, have presented an SoS approach to airport security. Analysis of airport operations is commonly performed in isolation, sharing only simple information such as flight schedules. With the increased concern over security in our airports, a new approach is required whereby all aspects of the airport operations are considered. A system-of-systems methodology is proposed and demonstrated through example. This method provides the decision maker with an improved understanding of the implication of policy decisions, resource allocations, and infrastructure investment strategies, through the capture of emergent behaviors and interdependences. New tools and methodologies are required for model development and analysis. These tools and methods are presented in this chapter.

1.3.2.10 *KASERS in SoS design*

Rubin, in Chapter 17, has used KASER (Knowledge Amplification by Structured Expert Randomization) as a tool for the SoS design. The author indicates that the U.S. Army needs robotic combat vehicles that can autonomously navigate the battlefield and carry out planned missions that necessarily embody unplanned details. On one end of the spectrum lie the simple

insect-like robots that have been popularized by Steel and Brooks [1995]. Their simple behaviors can be evolved, in much the same manner as a simple program can be created through the use of chance alone. Of course, more complex behaviors cannot be tractably evolved because the search space here grows exponentially. What is needed are heuristics to guide the evolutionary process. We can of course program search strategies and have indeed programmed robots to perform a myriad of complex functions—from the robot Manny's (U.S. Army) ability to walk the battlefield to unmanned aerial vehicles (UAVs). What is needed is a means to program more complex and reliable functionality for constant cost. That is, a system of systems (SoS) is needed. For example, one can program a robotic vehicle to sense and avoid an obstacle on the right. But then, what is the cost of programming the same robot to sense and avoid an obstacle on the left? It should be less and is, to some extent, if object-oriented component-based programs are written. The problem here though is that the cost is front loaded. The programmer needs to know *a priori* most of the domain symmetries if he or she is to capture them in the form of objects. A better solution is to do for symbolic programming what fuzzy logic did for Boolean logic [Zadeh, 1996; Rubin, 1999]. That is, we need the programmer to be able to constrain the robot's behavior in the form of generalized actions. Then, instances of these generalizations constitute the desired program. Even search-control heuristics can be acquired through the exercise of this paradigm. Instead of programming the robot to do a specific action, we program the robot to (heuristically) search a space of actions for one or more that is consistent with environmental constraints. The writing of such programs is easier for the human, and the constraints that instantiate them serve to render the search space tractable for the machine.

1.3.2.11 *System-of-systems standards*

Johnson, in Chapter 18, discusses a very key issue in SoSE. Standards are found in every arena of human endeavor. Representative examples are found in technical specifications, methods of measurement, medical diagnostics, judicial procedures, management processes, power distribution, building codes, information production and categorization, food production and processing, and financial transactions. Clear standards usually benefit all the players in a given field or industry; however, there are times when a standard may allow one group to compete effectively against another, even though their product may be inferior (e.g., Matsushita VHS versus Sony Betamax). One current example of the competition for standard supremacy in the digital video realm is between High Definition (HD) and Blue-ray. Currently, there are no standards specifically related to system of systems (SoS) or system of systems engineering (SoSE) products, processes, management, or other aspects or endeavors.

This chapter gives some background on standards, how they are used, and identifies some arenas in which dialogue is pointing to the instantiation of potential standards that could impact and foster future SoS standards

development. The first section provides a discussion of the what, why, and how of contemporary standards and their development, evolution, application, and management. The next section presents SoS standards considerations, pathfinder SoS endeavors and their approaches to standards use and implementation, and potentially emerging standards. Considerations are for SoS standards evolution, development, applicability, and management. The Future Combat Systems SoS projects approach to the use of standards to ensure communications interoperability among families of systems (FoS), and the Global Earth Observation system-of-systems approach to the identification, registration, and accommodation of disparate standards are two examples given of pathfinder SoS endeavors and their approaches to standards use and implementations. Emerging potential SoS standards and tools are given to wrap up the section on system-of-systems standards. Final comments conclude the chapter with presentation of the International Consortium on System of Systems (ICSOS) and discussion of their initial perspectives concerning SoS standards and the inevitable instantiation of a committee to guide and oversee SoS standards evolution, development, usage, application, and management.

Since system-of-systems literature, definitions, and perspectives are marked with great variability in the engineering community, standards are key need here. Viewed as an extension of systems engineering to a means of describing and managing social networks and organizations, the variations of perspectives leads to difficulty in advancing and understanding the discipline. Standards have been used to facilitate a common understanding and approach to align disparities of perspectives to drive a uniform agreement to definitions and approaches. Having the ICSOS—International Consortium on System of Systems [De Laurentis et al., 2007]—represent to the IEEE and INCOSE for support of technical committees to derive standards for system of systems will help unify and advance the discipline for engineering and healthcare.

1.4 Other SoSE issues

In this section, for the benefit of the readers, a wider perspective on system of systems and system of systems engineering from a recent work by the author [Jamshidi, 2008] will be given.

1.4.1 Open systems approach to system of systems engineering

Azani [2008], in Jamshidi [2008], has discussed an open systems approach to SoSE. The author notes that SoS exists within a continuum that contains ad hoc, short-lived, and relatively simple SoS on one end, and long-lasting, continually evolving, and complex SoS on the other end of the continuum. Military operations and less sophisticated biotic systems (e.g., bacteria and ant colonies) are examples of ad hoc, simple, and short-lived SoS, while galactic and more sophisticated biotic systems (e.g., ecosystem, human colonies) are

examples of SoS at the opposite end of the SoS continuum. The engineering approaches utilized by galactic SoS are at best unknown and perhaps forever inconceivable. However, biotic systems of systems seem to follow, relatively speaking, less complicated engineering and development strategies, allowing them to continually learn and adapt, grow and evolve, resolve emerging conflicts, and have more predictable behavior. Based on what the author already knows about biotic SoS, it is apparent that these systems employ robust reconfigurable architectures, enabling them to effectively capitalize on open systems development principles and strategies such as modular design, standardized interfaces, emergence, natural selection, conservation, synergism, symbiosis, homeostasis, and self-organization. Azani [2008] provides further elaboration on open systems development strategies and principles utilized by biotic SoS, discusses their implications for engineering of man-made SoS, and introduces an integrated SoS development methodology for engineering and development of adaptable, sustainable, and interoperable SoS based on open systems principles and strategies.

1.4.2 *SoS integration*

Integration is probably the key viability of any SoS. Integration of SoS implies that each system can communicate and interact (control) with the SoS regardless of their hardware, software characteristics, or nature. This means that they need to have the ability to communicate with the SoS or a part of the SoS without compatibility issues such as operating systems, communication hardware, and so on. For this purpose, an SoS needs a common language the SoS's systems can speak. Without having a common language, the systems of any SoS cannot be fully functional, and the SoS cannot be adaptive in the sense that new components cannot be integrated to it without major effort. Integration also implies the control aspects of the SoS, because systems need to understand each other in order to take commands or signals from other SoS systems. For further insight, see the work by Cloutier et al. [2008] on network centric architecture of SoS.

1.4.3 *Engineering of SoS*

Emerging needs for a comprehensive look at the applications of classical systems engineering issue in SoSE will be discussed in this volume. The thrust of the discussion will concern the reality that the technological, human, and organizational issues are each far different when considering a system of systems or federation of systems and that these needs are very significant when considering system of systems engineering and management. As we have noted, today there is much interest in the engineering of systems that are comprised of other component systems, and where each of the component systems serves organizational and human purposes. These systems have several principal characteristics that make the system family designation

appropriate: operational independence of the individual systems, managerial independence of the systems; often large geographic and temporal distribution of the individual systems; emergent behavior, in which the system family performs functions and carries out purposes that do not reside uniquely in any of the constituent systems, but which evolve over time in an adaptive manner, and where these behaviors arise as a consequence of the formation of the entire system family and are not the behavior of any constituent system. The principal purposes supporting engineering of these individual systems and the composite system family are fulfilled by these emergent behaviors. Thus, a system of systems is never fully formed or complete. Development of these systems is evolutionary and adaptive over time, and structures, functions, and purposes are added, removed, and modified as experience of the community with the individual systems and the composite system grows and evolves. The systems engineering and management of these systems families poses special challenges. This is especially the case with respect to the federated systems management principles that must be utilized to deal successfully with the multiple contractors and interests involved in these efforts. Please refer to the paper by [Sage and Biemer \[2007\]](#) and [De Laurentis et al. \[2007\]](#) for the creation of an SoS Consortium (ICSOS) of concerned individuals and organizations by the author of this chapter. See [Wells and Sage \[2008\]](#) for more information and challenges ahead in SoSE.

1.4.4 SoS management: the governance of paradox

Sausser and Boardman [2008], in Jamshidi [2008], have presented an SoS approach to the management problem. They note that the study of SoS has moved many to support their understanding of these systems through the groundbreaking science of networks. The understanding of networks and how to manage them may give one the fingerprint which is independent of the specific systems that exemplify this complexity. The authors point out that it does not matter whether they are studying the synchronized flashing of fireflies, space stations, structure of the human brain, the Internet, the flocking of birds, a future combat system, or the behavior of red harvester ants. The same emergent principles apply: large is really small; weak is really strong; significance is really obscure; little means a lot; simple is really complex; and complexity hides simplicity. The conceptual foundation of complexity is paradox, which leads us to a paradigm shift in the SE (systems engineering) body of knowledge.

Paradox exists for a reason, and there are reasons for systems engineers to appreciate paradox even though they may be unable to resolve them as they would a problem specification into a system solution. Hitherto paradoxes have confronted current logic only to yield at a later date to more refined thinking. The existence of paradox is always the inspirational source for seeking new wisdom, attempting new thought patterns, and ultimately building systems for the “flat world.” It is our ability to govern, not control,

these paradoxes that will bring new knowledge to our understanding on how to manage the emerging complex systems called system of systems.

Sauser and Boardman [2008] have established a foundation in what has been learned about how one practices project management, established some key concepts and challenges that make the management of SoS different from our fundamental practices, presented an intellectual model for how they classify and manage an SoS, appraised this model with recognized SoS, and concluded with grand challenges for how they may move their understanding of SoS management beyond the foundation.

1.4.5 Deepwater coastguard program

One of the earliest realizations of an SoS in the United States is the so-called Deepwater Coastguard Program shown in Figure 1.1. As seen here, the program takes advantage of all the necessary assets at their disposal, such as helicopters, aircrafts, cutters, satellite (GPS), ground station, human, and computers, with all systems of the SoS integrated together to react to unforeseen circumstances to secure the coastal borders of the Southeastern United States (e.g., Florida coast). The Deepwater program is making progress in the development and delivery of mission effective command, control, communications, computers, intelligence, surveillance, and reconnaissance (C4ISR) equipment [Keeter, 2007]. The SoS approach, the report goes on, has improved the operational capabilities of legacy cutters and aircraft, and will provide even more functionality when the next generation of surface and air platforms arrives in service. The key feature of the system is its ability to interoperate among all Coast Guard mission assets and capabilities with those of appropriate authorities both at local and federal levels.

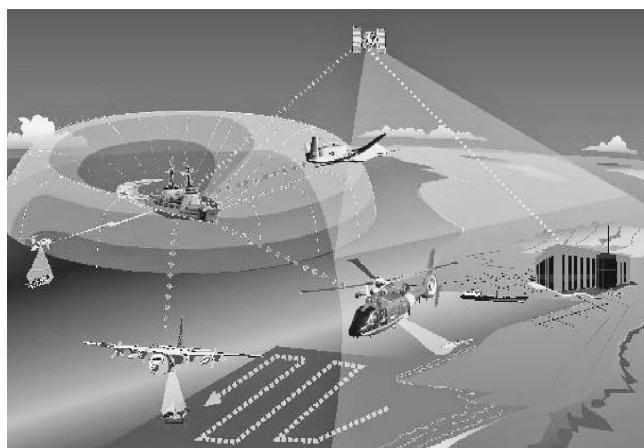


Figure 1.1 A security example of an SoS—deepwater coastguard configuration in United States.

1.4.6 Future combat missions

Another national security or defense application of SoS is the future combat mission (FCM). Figure 1.2 shows one of numerous possible configurations of an FCM system is:

...envisioned to be an ensemble of manned and potentially unmanned combat systems, designed to ensure that the Future Force is strategically responsive and dominant at every point on the spectrum of operations from nonlethal to full scale conflict. FCM will provide a rapidly deployable capability for mounted tactical operations by conducting direct combat, delivering both line-of-sight and beyond-line-of-sight precision munitions, providing variable lethal effect (nonlethal to lethal), performing reconnaissance, and transporting troops. Significant capability enhancements will be achieved by developing multifunctional, multimission and modular features for system and component commonality that will allow for multiple state-of-the-art technology options for mission tailoring and performance enhancements. The FCM force will incorporate and exploit information dominance to develop a common, relevant operating picture and achieve battle space situational understanding [Global Security Organization, 2007].

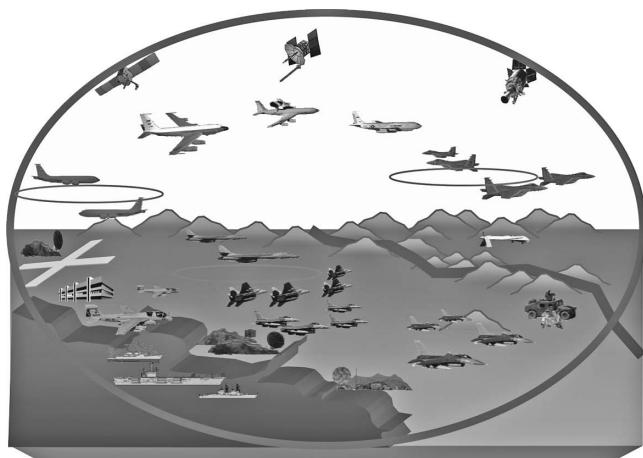


Figure 1.2 A defense example of an SoS. (Courtesy, Don Walker, Aerospace Corporation.)

1.4.7 *Systems engineering for the Department of Defense system of systems*

Dahmann [2008] in Jamshidi [2008], have addressed the national defense aspects of SoS. Military operations are the synchronized efforts of people and systems toward a common objective. In this way from an operational perspective, defense is essentially a systems-of-systems (SoS) enterprise. However, despite the fact that today almost every military system is operated as part of a system of systems, most of these systems were designed and developed without the benefit of systems engineering at the SoS level factoring the role the system will play in the broader system-of-systems context. With changes in operations and technology, the need for systems that work effectively together is increasingly visible. See also Chapter 12 by Dickerson in this book.

1.4.8 *Sensor networks*

The main purpose of sensor networks is to utilize the distributed sensing capability provided by tiny, low-powered and low-cost devices. Multiple sensing devices can be used cooperatively and collaboratively to capture events or monitor space more effectively than a single sensing device [Sridhar et al., 2007]. The realm of applications for sensor networks is quite diverse and includes military, aerospace, industrial, commercial, environmental, health monitoring, to name a few. Applications include traffic monitoring of vehicles, cross-border infiltration detection and assessment, military reconnaissance and surveillance, target tracking, habitat monitoring and structure monitoring, etc.

The communication capability of these small devices, often with heterogeneous attributes, makes them good candidates for system of systems. Numerous issues exist with sensor networks, such as data integrity, data fusion and compression, power consumption, multidecision making, and fault tolerance, all of which make these SoS very challenging just like other SoS. It is thus necessary to devise a fault-tolerant mechanism with a low computation overhead to validate the integrity of the data obtained from the sensors ("systems"). Moreover, a robust diagnostics and decision-making process should aid in monitoring and control of critical parameters to efficiently manage the operational behavior of a deployed sensor network. Specifically, Sridhar et al. [2007] will focus on innovative approaches to deal with multivariable multispace problem domain, as well as other issues, in wireless sensor networks within the framework of an SoS. In this book, see Chapter 11 by Wu et al. for an SoS approach in treating sensor networks.

1.4.9 *Healthcare systems*

Under a 2004 Presidential Order, the U.S. Secretary of Health has initiated the development of a National Healthcare Information Network (NHIN),

with the goal of creating a nationwide information system that can build and maintain Electronic Health Records (EHRs) for all citizens by 2014. The NHIN system architecture currently under development will provide a near-real-time heterogeneous integration of disaggregated hospital, departmental, and physician patient care data, and will assemble and present a complete current EHR to any physician or hospital a patient consults [Sloane, 2006]. The NHIN will rely on a network of independent Regional Healthcare Information Organizations (RHIOs) that are being developed and deployed to transform and communicate data from the hundreds of thousands of legacy medical information systems presently used in hospital departments, physician offices, and telemedicine sites into NHIN-specified meta-formats that can be securely relayed and reliably interpreted anywhere in the country. The NHIN “network of networks” will clearly be a very complex SoS, and the performance of the NHIN and RHIOs will directly affect the safety, efficacy, and efficiency of healthcare in the United States. Simulation, modeling, and other appropriate SoSE tools are under development to help ensure reliable, cost-effective planning, configuration, deployment, and management of the heterogeneous, life-critical NHIN and RHIO systems and subsystems [Sloane et al., 2007]. ICSOS represents an invaluable opportunity to access and leverage SoSE expertise already under development in other industry and academic sectors. ICSOS also represents an opportunity to discuss the positive and negative emergent behaviors that can significantly affect personal and public health status and the costs of healthcare in the United States [De Laurentis et al., 2007].

1.4.10 Global Earth Observation System of Systems

Global Earth Observation System of Systems (GEOSS) is a global project consisting of over 60 nations whose purpose is to address the need for timely, quality, long-term, global information as a basis for sound decision making [Butterfield et al., 2006]. Its objectives are (1) improved coordination of strategies and systems for Earth observations to achieve a comprehensive, coordinated, and sustained Earth observation system or systems, (2) a coordinated effort to involve and assist developing countries in improving and sustaining their contributions to observing systems and their effective utilization of observations and the related technologies, and (3) the exchange of observations recorded from *in situ*, in a full and open manner with minimum time delay and cost. In GEOSS, the

SoSE Process provides a complete, detailed, and systematic development approach for engineering systems of systems. Boeing's new architecture-centric, model-based systems engineering process emphasizes concurrent development of the system architecture model and system specifications. The process is applicable to

all phases of a system's lifecycle. The SoSE Process is a unified approach for system architecture development that integrates the views of each of a program's participating engineering disciplines into a single system architecture model supporting civil and military domain applications [Pearlman, 2006].

ICSoS will be another platform for all concerned around the globe to bring the progress and principles of GEOSS to formal discussions and examination on an annual basis (Figure 1.3).

1.4.11 E-enabling and SoS aircraft design via SoSE

A case of aeronautical application of SoS worth noting is that of e-enabling in aircraft design as a system of an SoS at Boeing Commercial Aircraft Division [Wilber, 2008]. The project focused on developing a strategy and technical architecture to facilitate making the airplane (Boeing 787, see [Figure 1.4](#)) network aware and capable of leveraging computing and network advances in industry. The project grew to include many ground-based architectural components at the airlines and at the Boeing factory, as well as other key locations such as the airports, suppliers and terrestrial Internet service suppliers (ISPs).

Wilber [2008] points out that the e-enabled project took on the task of defining a system of systems engineering solution to the problem of inter-operation and communication with the existing numerous and diverse elements that make up the airlines' operational systems (flight operations

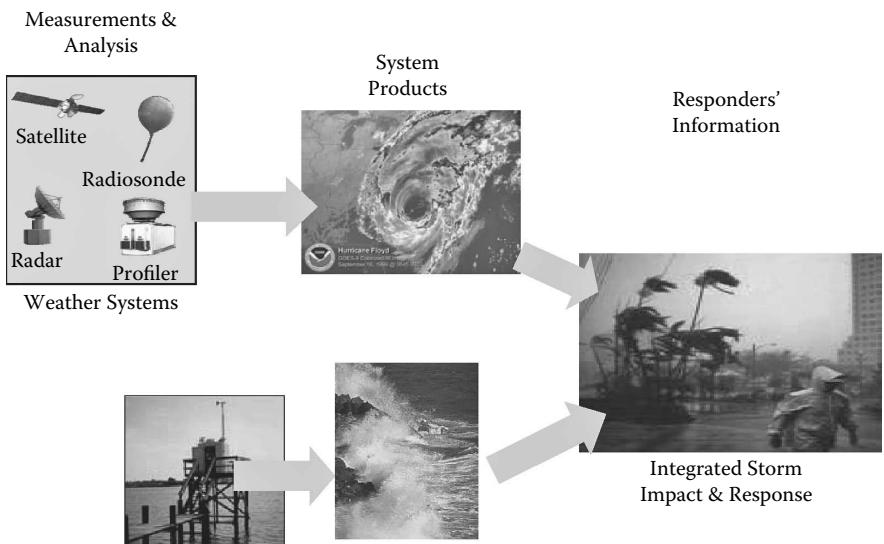
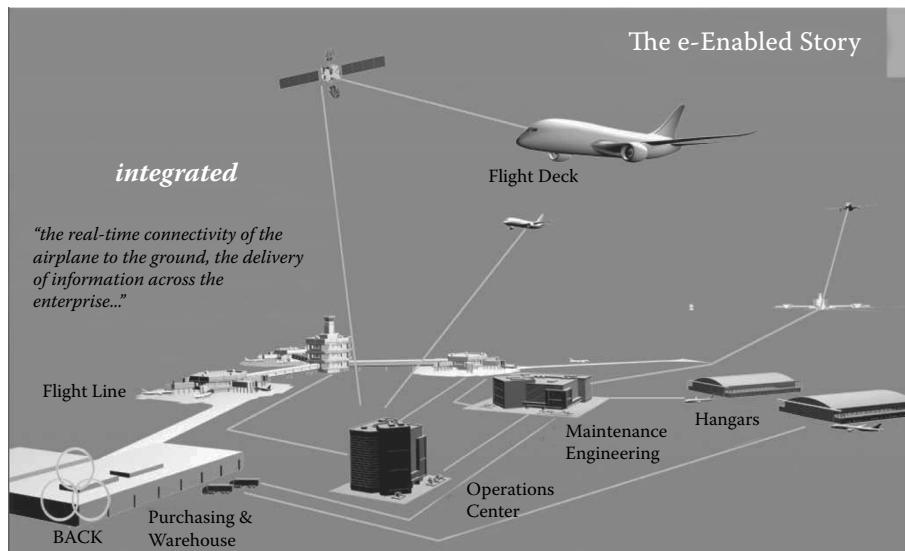


Figure 1.3 SoS of the GEOSS Project. (Courtesy, Jay Pearlman, Boeing Company.)



(a)



(b)

Figure 1.4 Application of an SoS for the Boeing 787. (a) Boeing's 787 Dreamliner. (b) E-enabling SoS for Boeing 787.

and maintenance operations). The objective has been to find ways of leveraging network-centric operations to reduce production, operations and maintenance costs for both Boeing and the airline customers.

One of the key products of this effort is the “e-enabled architecture.” The e-enabling architecture is defined at multiple levels of abstraction. There is a single top-level or “reference architecture” that is necessarily abstract and multiple “implementation architectures.” The implementation architectures

map directly to airplane and airline implementations and provide a family of physical solutions that all exhibit common attributes and are designed to work together and allow reuse of systems components. The implementation architectures allow for effective forward and retrofit installations addressing a wide range of market needs for narrow and widebody aircraft.

The 787 “open data network” is a key element of one implementation of this architecture. It enabled onboard and offboard elements to be networked in a fashion that is efficient, flexible, and secure. The fullest implementations are best depicted in Boeing’s GoldCare architecture and design.

Wilber [2007, 2008] presented an architecture at the reference level and how it has been mapped into the 787 airplane implementation. *GoldCare* environment is described and is used as an example of the full potential of the current e-enabling.

1.4.12 A system-of-systems perspective on infrastructures

Thissen and Herder [2008], in Jamshidi [2008], have touched upon a very important application in the service industry. Infrastructure systems (or infrasystems) providing services such as energy, transport, communications, and clean and safe water are vital to the functioning of modern society. Key societal challenges with respect to our present and future infrastructure systems relate to, among other things, safety and reliability, affordability, and transitions to sustainability. Infrasystem complexity precludes simple answers to these challenges. While each of the infrasystems can be seen as a complex system of systems in itself, increasing interdependency among these systems (both technologically and institutionally) adds a layer of complexity.

One approach to increased understanding of complex infrasystems that has received little attention in the engineering community thus far is to focus on the commonalities of the different sectors, and to develop generic theories and approaches such that lessons from one sector could easily be applied to other sectors. The system-of-systems paradigm offers interesting perspectives in this respect. The authors present, as an initial step in this direction, a fairly simple three-level model distinguishing the physical/technological systems, the organization and management systems, and the systems and organizations providing infrastructure-related products and services. The authors use the model as a conceptual structure to identify a number of key commonalities and differences between the transport, energy, drinking water, and ICT sectors. Using two energy-related examples, the authors further illustrate some of the system of systems-related complexities of analysis and design at a more operational level. The authors finally discuss a number of key research and engineering challenges related to infrastructure systems, with a focus on the potential contributions of systems-of-systems perspectives.

1.4.13 A system-of-systems view of services

Tien [2008], in Jamshidi [2008], has covered a very important application of SoS in today's global village—*service industry*. The services sector employs a large and growing proportion of workers in the industrialized nations, and it is increasingly dependent on information technology. While the interdependences, similarities, and complementarities of manufacturing and services are significant, there are considerable differences between goods and services, including the shift in focus from mass production to mass customization (whereby a service is produced and delivered in response to a customer's stated or imputed needs). In general, a service system can be considered to be a combination or recombination of three essential components—people (characterized by behaviors, attitudes, values, etc.), processes (characterized by collaboration, customization, etc.), and products (characterized by software, hardware, infrastructures, etc.). Furthermore, inasmuch as a service system is an integrated system, it is, in essence, a system of systems, and its objectives are to enhance its efficiency (leading to greater interdependency), effectiveness (leading to greater usefulness), and adaptiveness (leading to greater responsiveness). The integrative methods include a component's design, interface, and interdependency; a decision's strategic, tactical, and operational orientation; and an organization's data, modeling, and cybernetic consideration. A number of insights are also provided, including an alternative system-of-systems view of services; the increasing complexity of systems (especially service systems), with all the attendant life-cycle design, human interface, and system integration issues; the increasing need for real-time, adaptive decision making within such systems of systems; and the fact that modern systems are also becoming increasingly more human-centered, if not human-focused. Thus, products and services are becoming more complex and more personalized or customized.

1.4.14 System of systems engineering in space exploration

Jolly and Muirhead [2008], in Jamshidi [2008], have covered SoSE topics that are largely unique for space exploration with the intent to provide the reader a discussion of the key issues, the major challenges of the twenty-first century in moving from systems engineering to SoSE, potential applications in the future, and the current state of the art. Specific emphasis is placed on how software and electronics are revolutionizing the way space missions are being designed, including both the capabilities and vulnerabilities introduced. The role of margins, risk management, and interface control are all critically important in current space mission design and execution—but in SoSE applications they become paramount. Similarly, SoSE space missions will have extremely large, complex, and intertwined command and control and data distribution ground networks, most of which will involve extensive

parallel processing to produce tera- to petabytes of products per day and distribute them worldwide.

1.4.15 Robotic swarms as an SoS

As another application of SoS, a robotic swarm is considered by Sahin [2008] in Jamshidi [2008]. A robotic swarm based on ant colony optimization and artificial immune systems is considered. In the ant colony optimization, the author has developed a multiagent system model based on the food-gathering behaviors of the ants. Similarly, a multiagent system model is developed based on the human immune system. These multiagent system models, then, were tested on the mine detection problem. A modular micro robot is designed to perform to emulate the mine detection problem in a basketball court. The software and hardware components of the modular robot are designed to be modular so that robots can be assembled using hot-swappable components. An adaptive Time Division Multiple Access (TDMA) communication protocol is developed in order to control connectivity among the swarm robots without the user intervention.

1.4.16 Communication and navigation in space SoS

Bahsin and Hayden [2008], in Jamshidi [2008], have taken upon the challenges in communication and navigation for space SoS. They indicate that communication and navigation networks provide critical services in the operation, system management, information transfer, and situation awareness to the space system of systems. In addition, space systems of systems require system interoperability, enhanced reliability, common interfaces, dynamic operations, and autonomy in system management. New approaches to communications and navigation networks are required to enable the interoperability needed to satisfy the complex goals and dynamic operations and activities of the space system of systems. Historically space systems had direct links to Earth ground communication systems, or they required a space communication satellite infrastructure to achieve higher coverage around the Earth. It is becoming increasingly apparent that many systems of systems may include communication networks that are also systems of systems. These communication and navigation networks must be as nearly ubiquitous as possible and accessible on the demand of the user, much like the cell phone link is available at any time to an Earth user in range of a cell tower. The new demands on communication and navigation networks will be met by space Internet technologies. It is important to bring Internet technologies, Internet protocols (IP), routers, servers, software, and interfaces to space networks to enable as much autonomous operation of those networks as possible. These technologies provide extensive savings in reduced cost of operations. The more these networks can be made to run themselves, the less humans will have to schedule and control them. The Internet technologies also bring with

them a very large repertoire of hardware and software solutions to communication and networking problems that would be very expensive to replicate under a different paradigm. Higher bandwidths are needed to support the expected voice, video, and data transfer traffic for the coordination of activities at each stage of an exploration mission.

Existing communications, navigation, and networking have grown in an independent fashion, with experts in each field solving the problem just for that field. Radio engineers designed the payloads for today's "bent pipe" communication satellites. The Global Positioning satellite (GPS) system design for providing precise Earth location determination is an extrapolation of the LOng RAnge Navigation (LORAN) technique of the 1950s, where precise time is correlated to precise position on the Earth. Other space navigation techniques use artifacts in the RF communication path (Doppler shift of the RF and transponder-reflected ranging signals in the RF) and time transfer techniques to determine the location and velocity of a spacecraft within the solar system. Networking in space today is point to point among ground terminals and spacecraft, requiring most communication paths to/ from space to be scheduled such that communications is available only on an operational plan and is not easily adapted to handle multidirectional communications under dynamic conditions.

Bahsin and Hayden [2008] begin with a brief history of the communications, navigation, and networks of the 1960s and 1970s in use by the first system of systems, the NASA Apollo missions; it is followed by short discussions of the communication and navigation networks and architectures that the DoD and NASA employed from the 1980s onward. Next is a synopsis of the emerging space system of systems that will require complex communication and navigation networks to meet their needs. Architecture approaches and processes being developed for communication and navigation networks in emerging space system and systems are also described. Several examples are given of the products generated in using the architecture development process for space exploration systems. The architecture addresses the capabilities to enable voice, video, and data interoperability needed among the explorers during exploration, while in habitat, and with Earth operations. Advanced technologies are then described that will allow space system of systems to operate autonomously or semiautonomously.

1.4.17 National security

Perhaps one of the most talked-about application areas of SoSE is national security. After many years of discussing the goals, merits, and attributes of SoS, very few tangible results or solutions have appeared in this or other areas of this technology. It is commonly believed that,

Systems Engineering tools, methods, and processes
are becoming inadequate to perform the tasks needed

to realize the systems of systems envisioned for future human endeavors. This is especially becoming evident in evolving national security capabilities realizations for large-scale, complex space and terrestrial military endeavors. Therefore the development of Systems of Systems Engineering tools, methods and processes is imperative to enable the realization of future national security capabilities [Walker, 2007].

In most SoSE applications, heterogeneous systems (or communities) are brought together to cooperate for a common good and enhanced robustness and performance.

These communities range in focus from architectures, to lasers, to complex systems, and will eventually cover each area involved in aerospace related national security endeavors. These communities are not developed in isolation in that cross-community interactions on terminology, methods, and processes are done [Walker, 2007].

The key is to have these communities work together to guarantee the common goal of making our world a safer place for all.

1.4.18 Electric power systems grids as SoS

Korba and Hiskens [2008] in Jamshidi [2008], provide an overview of the systems of systems that are fundamental to the operation and control of electrical power systems. Perspectives are drawn from industry and academia and reflect theoretical and practical challenges that are facing power systems in an era of energy markets and increasing utilization of renewable energy resources (see also [Duffy et al. \[2008\]](#)). Power systems cover extensive geographical regions and are composed of many diverse components. Accordingly, power systems are large-scale, complex, dynamical systems that must operate reliably to supply electrical energy to customers. Stable operation is achieved through extensive monitoring systems and a hierarchy of controls, which together seek to ensure that total generation matches consumption, and that voltages remain at acceptable levels. Safety margins play an important role in ensuring reliability, but tend to incur economic penalties. Significant effort is therefore being devoted to the development of demanding control and supervision strategies that enable reduction of these safety margins, with consequent improvements in transfer limits and profitability.

1.4.19 SoS approach for renewable energy

Duffy et al. [2008] in Jamshidi [2008] have detailed the SoS approach to sustainable supply of energy. They note that over one-half of the petroleum consumed in the United States is imported, and that percentage is expected to rise to 60% by 2025. America's transportation system of systems relies almost exclusively on refined petroleum products, accounting for over two-thirds of the oil used. Each day, over eight million barrels of oil are required to fuel over 225 million vehicles that constitute the U.S. light-duty transportation fleet. The gap between U.S. oil production and transportation oil needs is projected to grow, and the increase in the number of light-duty vehicles will account for most of that growth. On a global scale, petroleum supplies will be in increasingly higher demand as highly populated developing countries expand their economies and become more energy intensive. Clean forms of energy are needed to support sustainable global economic growth while mitigating impacts on air quality and the potential effects of greenhouse gas emissions. The growing dependence of the United States on foreign sources of energy threatens her national security. As a nation, the authors assert that we must work to reduce our dependence on foreign sources of energy in a manner that is affordable and preserves environmental quality.

1.4.20 Sustainable environmental management from a system of systems engineering perspective

Hipel et al. [2008] in Jamshidi [2008] have provided a rich range of decision tools from the field of SE for addressing complex environmental SoS problems in order to obtain sustainable, fair and responsible solutions to satisfy the value systems of stakeholders, including the natural environment and future generations who are not even present at the bargaining table. To better understand the environmental problem being investigated and thereby eventually reach more informed decisions, the insightful paradigm of a system of systems can be readily utilized. For example, when developing solutions to global warming problems, one can envision how societal systems, such as agricultural and industrial systems, interact with the atmospheric system of systems, especially at the tropospheric level. The great import of developing a comprehensive toolbox of decision methodologies and techniques is emphasized by pointing out many current pressing environmental issues, such as global warming and its potential adverse affects, and the widespread pollution of our land, water, and air systems of systems. To tackle these large-scale complex systems of systems problems, systems engineering decision techniques that can take into account multiple stakeholders having multiple objectives are explained according to their design and capabilities. To illustrate how systems decision tools can be employed in practice to assist in reaching better decisions for benefiting society, different decision tools are applied to three real-world systems of systems environmental problems.

Specifically, the graph model for conflict resolution is applied to the international dispute over the utilization of water in the Aral Sea Basin; a large-scale optimization model founded upon concepts from cooperative game theory, economics, and hydrology is utilized for systematically investigating the fair allocation of scarce water resources among multiple users in the South Saskatchewan River Basin in Western Canada; and multiple criteria decision analysis methods are used to evaluate and compare solutions to handling fluctuating water levels in the five Great Lakes located along the border of Canada and the United States [Wang et al., 2007].

1.4.21 Transportation systems

The National Transportation System (NTS) can be viewed as a collection of layered networks composed by heterogeneous systems for which the Air Transportation System (ATS) and its National Airspace System (NAS) is one part. At present, research on each sector of the NTS is generally conducted independently, with infrequent and/or incomplete consideration of scope dimensions (e.g., multimodal impacts and policy, societal, and business enterprise influences) and network interactions (e.g., layered dynamics within a scope category). This isolated treatment does not capture the higher-level interactions seen at the NTS or ATS architecture level; thus, modifying the transportation system based on limited observations and analyses may not necessarily have the intended effect or impact. A systematic method for modeling these interactions with a system-of-systems approach is essential to the formation of a more complete model and understanding of the ATS, which would ultimately lead to better outcomes from high-consequence decisions in technological, socioeconomic, operational, and political policy-making context [De Laurentis, 2005]. This is especially vital as decision makers in both the public and private sector, for example at the interagency Joint Planning and Development Office (JPDO), which is charged with transformation of air transportation, are facing problems of increasing complexity and uncertainty in attempting to encourage the evolution of superior transportation architectures [De Laurentis and Callaway, 2004; De Laurentis, 2008].

Keating [2008], in Jamshidi [2008], has also provided insight into this important aspect of SoS. Emergent behavior of an SoS resembles the slowdown of the traffic going through a tunnel, even in the absence of any lights, obstacles, or accident. A tunnel, automobiles, and the highway, as systems of an SoS, have an emergent behavior or property in slowing down [Morley, 2006]. Fisher [2006] has noted that an SoS cannot achieve its goals due to its emergent behaviors. The author explores interdependencies among systems, emergence, and “interoperation” and develops maxim-like findings such as

these: (1) Because they cannot control one another, autonomous entities can achieve goals that are not local to themselves only by increasing their influence through cooperative interactions with others. (2) Emergent composition is often poorly understood and sometimes misunderstood because it has few analogies in traditional systems engineering. (3) Even in the absence of accidents, tight coupling can ensure that a system of systems is unable to satisfy its objectives. (4) If it is to remain scalable and affordable no matter how large it may become, a system's cost per constituent must grow less linearly with its size. (5) Delay is a critical aspect of systems of systems.

1.4.22 System of land, sea, and air vehicles as SoS

One of the more appropriate system of systems at the research level for both academia and industry is a system of “rovers.” A team of researchers at the University of Texas, San Antonio ACE (Autonomous Control Engineering) Center have embarked on a very challenging program to network centric rovers along all three domains of operations—land, sea and air. Figure 1.5 and [Figure 1.6](#) show some of the work being done at the UTSA-ACE Center. Further information can be secured at ace.utsa.edu as well as references [Azarnoosh et al., 2006; Sahin et al., 2007]. The work is being done on a system of UAVs, and then plans are to network heterogeneous combinations (e.g., underwater vehicles in contact and cooperate with a UAV and that in turn in contact with a land rover). Other published works of the ACE team are given in Jaimes et al., 2008; Joordens et al., 2008; Kuma Ray et al., 2008; Benavidez et al., 2008; Shaneyfelt et al., 2008a,b; Prevost et al., 2008; Nagothu et al., 2008; and Parisi et al., 2008.



Figure 1.5 A system of land rovers performing a fire-detection task. (Courtesy of ACE, University of Texas, San Antonio, ace.utsa.edu.)



Figure 1.6 A system of underwater rovers performing a communication task. (Courtesy of ACE, University of Texas, San Antonio.)

1.5 Conclusions

This chapter is written to serve as an introduction to the book. We also gave an up-to-date state of the art in systems of systems and systems of systems engineering based on other works of the author. The subject matter of this book is an unsettled topic in engineering in general and in systems engineering in particular. An attempt has been made to cover many open questions in both theory and applications of SoS and SoSE. It is our intention that this book would be the beginning of much debate and challenges among and by the readers of this book. The book is equally intended to benefit industry, academia or government. A sister volume, by the author, on the subject is under press at the present time and can give readers further insight into SoS [Jamshidi, 2008].

References

- Abbott, R. 2006, Open at the top; open at the bottom; and continually (but slowly) evolving. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, April 2006.
- Abel, A. and S. Sukkarieh. 2006, The coordination of multiple autonomous systems using information theoretic political science voting models. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, April 2006.
- ANSI/IEEE. 2000. Recommended practice for architecture description of software-intensive systems. ANSI/IEEE 1471-2000. Institute of Electrical and Electronics Engineers.
- Asimow, M. 1962. *Introduction to Design*. Prentice-Hall, Englewood Cliffs, NJ.

- Azani, C. 2008. An open systems approach to system of systems engineering. *System of Systems Engineering—Innovations for the 21st Century*, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Azarnoosh, H., B. Horan, P. Sridhar, A. M. Madni, and M. Jamshidi. 2006. Towards optimization of a real-world robotic-sensor system of systems. *Proceedings of World Automation Congress (WAC) 2006*, July 24–26, Budapest, Hungary.
- Benavidez, P., K. Nagothu, A. Kumar Ray, T. Shaneyfelt, S. Kota, L. Behera, and M. Jamshidi. 2008. Multi-domain robotic swarm communication systems, *Proc. IEEE System of Systems Engineering Conference*, Monterey Bay, CA, June 2–4.
- Bhasin, K. B. and J. L. Hayden. 2008. Communication and navigation networks in space system of systems, in *System of Systems Engineering—Innovations for the 21st Century*, Chapter 15, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Butterfield, M. L., J. Pearlman, and S. C. Vickroy. 2006. System-of-systems engineering in a global environment. *Proceedings of International Conference on Trends in Product Life Cycle, Modeling, Simulation and Synthesis PLMSS*, 2006.
- Carlock, P. G., and R. E. Fenton. 2001. System of systems (SoS) enterprise systems for information-intensive organizations. *Systems Engineering* 4(4):242–261.
- Cloutier, R., M. J. DiMario, and H. W. Polzer. 2008. Net-centricity and system of systems. *System of Systems Engineering—Innovations for the 21st Century*, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Crossley, W. A. 2004. System of systems: an introduction of Purdue University Schools of Engineering's Signature Area. *Engineering Systems Symposium*, March 29–31, 2004, Tang Center–Wong Auditorium, MIT.
- Dagli, C. H. and N. K. Ergin. 2008. System of systems architecting. *System of Systems Engineering—Innovations for the 21st Century*, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Dahmann, J. 2008. Systems engineering for Department of Defense systems of systems, in *System of Systems Engineering—Innovations for the 21st Century*, Chapter 9, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Dahmann, J. and K. Baldwin. 2008. Systems engineering for Department of Defense systems of systems. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 9, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- De Laurentis, D.A. 2005. Understanding transportation as a system-of-systems design problem. *AIAA Aerospace Sciences Meeting and Exhibit*, 10–13 Jan. 2005. AIAA-2005-123.
- De Laurentis, D. 2008. Understanding transportation as a system-of-systems problem. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 20, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- De Laurentis, D.A. and R. K. Callaway. 2004. A system-of-systems perspective for future public policy. *Review of Policy Research* 21(6): 829–837.
- De Laurentis, D., C. Dickerson, M. Di Mario, P. Gartz, M. Jamshidi, S. Nahavandi, A. Sage, E. Sloane, and D. Walker. 2007. A case for an international consortium on system of systems engineering. *IEEE Systems Journal* 1(1):68–73.
- DiMario, M. J. 2006. System of systems interoperability types and characteristics in joint command and control. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, April 2006.
- Duffy, M., B. Garrett, C. Riley, and D. Sandor. 2008. Future transportation fuel system of systems. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 17, (M. Jamshidi, Ed.), John Wiley & Sons, New York.

- Fisher, D. 2006. *An Emergent Perspective on Interoperation in Systems of Systems*, (CMU/SEI-2006-TR- 003). Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.
- Gladwell, M. 2005. *Blink: The Power of Thinking Without Thinking*, Little, Brown and Company, Time Warner Book Group, New York.
- Global Security Organization. 2007. Future combat systems—background. <http://www.globalsecurity.org/military/systems/ground/fcs-back.htm>.
- Hipel, K., A. Obeidi, L. Fang, and D. M. Kilgour. 2008. Sustainable environmental management from a system of systems engineering perspective. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 11, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Jaimes, A., J. Gomez, S. Kota, and M. Jamshidi. 2008. An approach to surveillance in an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles UAV(s), *Proc. IEEE System of Systems Engineering Conference*, Monterey Bay, CA, June 2–4, paper # 1569111991.
- Jamshidi, M. 2005. Theme of the IEEE SMC 2005, Waikoloa, Hawaii, USA. <http://ieeesmc2005.unm.edu/>, October 2005.
- Jamshidi, M. 2008. *System of Systems Engineering—Innovations for the 21st Century*, Wiley & Sons, New York.
- Jamshidi, M. 2008. Introduction to system of systems engineering. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 1, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Jolly, S. D. and B. Muirhead. 2008. Communication and navigation networks in space system of systems. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 15, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Joordens, M., J. Serna, S. Songer, C. Friday, J. Hoy, R. Seiger, W. Madalinski, and M. Jamshidi. 2008. Low cost underwater robot sensor suite, *Proc. IEEE System of Systems Engineering Conference*, Monterey Bay, CA, June 2–4, paper # 1569095252.
- Keating, C. B. 2008. Emergence in system of systems. *System of Systems Engineering—Innovations for the 21st Century*, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Keeter, H. C. 2007. Deepwater command, communication, sensor electronics build enhanced operational capabilities. U.S. Coastguard Deepwater Program site, <http://www.uscg.mil/deepwater/media/feature/july07/c4isr072007.htm>.
- Korba, P. and I. A. Hiskins. 2008. Operation and control of electrical power systems. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 16, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Kotov, V. 1997. Systems of systems as communicating structures. Hewlett Packard Computer Systems Laboratory Paper HPL-97-124, pp. 1–15.
- Kumar Ray, A., M. Gupta, L. Behera, and M. Jamshidi. 2008. Sonar based autonomous automatic guided vehicle (AGV) navigation, *Proc. IEEE System of Systems Engineering Conference*, Monterey Bay, CA, June 2–4.
- Lopez, D. 2006. Lessons learned from the front lines of the aerospace. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, April 2006.
- Lukasik, S. J. 1998. Systems, systems of systems, and the education of engineers. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing* 12(1):55–60.
- Manthorpe, W. H. 1996. The emerging joint system of systems: a systems engineering challenge and opportunity for APL, *John Hopkins APL Technical Digest* 17(3):305–310.
- Meilich, A. 2006. System of systems (SoS) engineering & architecture challenges in a net centric environment. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, April 2006.

- Mittal, S., B. P. Zeigler, J. L. R. Martin, and F. Sahin. 2008, Modeling and simulation for systems of systems engineering, *Systems Engineering—Innovations for the 21st Century*, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Morley, J. 2006. Five maxims about emergent behavior in systems of systems. <http://www.sei.cmu.edu/news-at-sei/features/2006/06/feature-2-2006-06.htm>.
- Nagothu, K., M. Joordens, and M. Jamshidi. 2008. Distributed protocol for communications among underwater vehicles, *Proc. IEEE System of Systems Engineering Conference*, Monterey Bay, CA, June 2–4, paper # 1569108743.
- Parisi, C., F. Sahin, and M. Jamshidi. 2008. A discrete event XML based system of systems simulation for robust threat detection and integration, *Proc. IEEE System of Systems Engineering Conference*, Monterey Bay, CA, June 2–4.
- Pearlman, J. 2006. GEOSS—global earth observation system of systems. Keynote presentation, 2006 IEEE SoSE Conference, Los Angeles, CA, April 24, 2006.
- Pei, R. S. 2000. Systems of systems integration (SoSI)—a smart way of acquiring Army C4I2WS systems. *Proceedings of the Summer Computer Simulation Conference*, pp. 134–139.
- Prevost, J., M. A. Joordens, and M. Jamshidi. 2008. Simulation of underwater robots using Microsoft's Robot Studio®, *Proc. IEEE System of Systems Engineering Conference*, Monterey Bay, CA, June 2–4.
- Rubin, S.H. 1999. Computing with words, *IEEE Trans. Syst. Man, Cybern.*, 29(4): 518–524.
- Sage, A. P. and S. M. Biemer. 2007. Processes for system family architecting, design, and integration. *IEEE Systems Journal*, ISJ1-1, September, pp. 5–16, 2007.
- Sage, A. P. and C. D. Cuppan. 2001. On the systems engineering and management of systems of systems and federations of systems. *Information, Knowledge, Systems Management* 2(4):325–334.
- Sahin, F. 2008. Robotic swarm as a system of systems. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 19, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Sahin, F., M. Jamshidi, and P. Sridhar. 2007. A discrete event XML based simulation framework for system of systems architectures. *Proceedings the IEEE International Conference on System of Systems*, April 2007.
- Sauser, B. and J. Boardman. 2008. System of systems management. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 8, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Shaneyfelt, T., S. Kota, and M. Jamshidi. 2008a. Towards net-centric system of systems robotics in air, sea and land, *Proc. IEEE System of Systems Engineering Conference*, Monterey Bay, CA, June 2–4, paper # 1569101444.
- Shaneyfelt, T., M. A. Joordens, K. Manoj Nagothu, and M. Jamshidi. 2008b. RF communication between surface and underwater robotic swarms, *Proc. World Automation Congress*, Waikoloa, HI, September 28–October 2.
- Sloane, E. 2006. Understanding the emerging national healthcare IT infrastructure. *24x7 Magazine*. December, 2006.
- Sloane, E., T. Way, V. Gehlot, and R. Beck. 2007. Conceptual SoS model and simulation systems for a next generation National Healthcare Information Network (NHIN-2). *Proceedings of the 1st Annual IEEE Systems Conference*, Honolulu, HI, April 9–12, 2007.
- Sridhar, P., A. M. Madni, M. Jamshidi. 2007. Hierarchical aggregation and intelligent monitoring and control in fault-tolerant wireless sensor networks. *IEEE Systems Journal* 1(1):38–54.

- Steels, L. and R. Brooks (Eds.). 1995. *The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents*, Mahwah, NJ: Lawrence Erlbaum Assoc.
- Thissen, W. and P. M. Herder. 2008. System of systems perspectives on infrastructures. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 11, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Tien, J. M. 2008. A system of systems view of services. *System of Systems Engineering—Innovations for the 21st Century*, Chapter 13, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Walker, D. 2007. Realizing a corporate SOSE environment. Keynote presentation, 2007 IEEE SoSE Conference, San Antonio, TX, 18 April 2007.
- Wang, L., L. Fang, and K. W. Hipel. 2007. On achieving fairness in the allocation of scarce resources: measurable principles and multiple objective optimization approaches. *IEEE Systems Journal* 1(1):17–28, 2007.
- Wells, G. D. and A. P. Sage. 2008. Engineering of a system of systems. *System of Systems Engineering—Innovations for the 21st Century*, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Wilber, F. R. 2007. A system of systems approach to e-enabling the commercial airline applications from an airframer's perspective. Keynote presentation, 2007 IEEE SoSE Conference, San Antonio, TX, 18 April 2007.
- Wilber, F. R. 2008. Boeing's SOSE approach to e-enabling commercial airlines, *System of Systems Engineering—Innovations for the 21st Century*, (M. Jamshidi, Ed.), John Wiley & Sons, New York.
- Wojcik, L. A. and K. C. Hoffman. 2006. Systems of systems engineering in the enterprise context: a unifying framework for dynamics. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, April 2006.
- Zadeh, L.A. 1996. Fuzzy logic = computing with words, *IEEE Trans. Fuzzy Syst.*, 4(2): 103–111.

chapter two

SoS architecture

Reggie Cole

Contents

2.1	Introduction.....	37
2.2	Architecture principles and practices	41
2.2.1	The architecture design process.....	41
2.2.1.1	Analysis.....	41
2.2.1.2	Synthesis.....	42
2.2.1.3	Evaluation	43
2.2.2	Architecture design principles	44
2.2.2.1	Needs often compete.....	45
2.2.2.2	Needs change over time	45
2.2.2.3	Resource availability constrains the solution space	46
2.2.2.4	Design compromise is necessary.....	46
2.3	SoS architecture considerations	47
2.3.1	Autonomy	48
2.3.2	Complexity	49
2.3.3	Diversity.....	49
2.3.4	Integration strategy	49
2.3.5	Data architecture	52
2.3.6	System protection	54
2.4	Critical success factors in architecting SoS solutions.....	55
2.4.1	Robust design.....	55
2.4.2	Architecture alignment	56
2.4.3	Architecture governance.....	56
2.4.4	Architecture description	57
2.5	Architecture frameworks	58
	References	68

2.1 Introduction

Along with defining the problem, one of the most important jobs of the systems engineer is to partition the problem into smaller, more manageable problems and make critical decisions about the solution. One of the most

critical decisions is the architecture—the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution [1]. While it is impossible to understand all the characteristics and consequences of the architecture at the time the system is designed, it is possible to produce a system architecture that maximizes the ability of the system to meet user needs, while minimizing the unintentional consequences.

In one sense, architecting a complex system that is comprised of a number of collaborating independent systems is no different than designing a simple system. Both start with definition of a problem and conception of solution. Both warrant consideration of environmental context. Both involve analysis of factors related to effectiveness. And both require design compromises and balancing of competing priorities. The basic process is the same. In fact, it has been well documented for nearly 50 years [2]. But compared to the design of simple systems, the complexity of designing a system-of-systems (SoS) solution is daunting, and the design process must be approached with that complexity in mind.

Figure 2.1 shows the basic SoS model. Notice that the system elements of the SoS are themselves systems. They address their own needs and solve their own specific problems. They have their own emergent properties. In fact, they have their own purpose for existing. But they are also part of a larger system—the SoS—which itself addresses a need and has emergent properties, resulting from the interaction of systems within the SoS. The need to maintain autonomy while at the same time operating within the SoS context greatly increases the complexity of an SoS and is at the heart of the SoS architecture challenge [3].

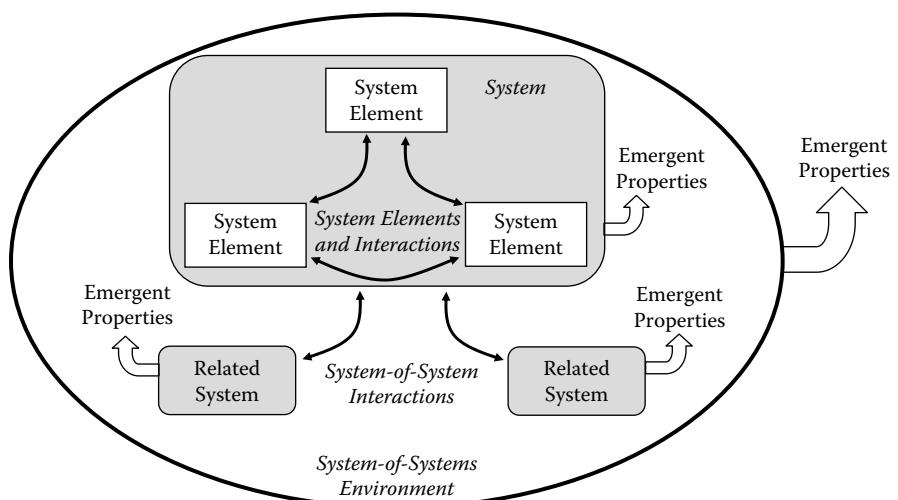


Figure 2.1 The basic SoS model.

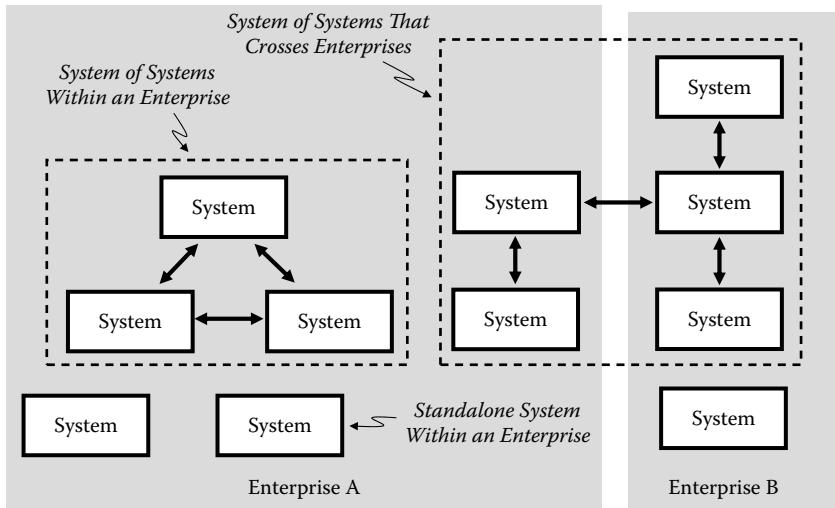


Figure 2.2 The SoS context.

System-of-systems architecture is primarily concerned with the architecture of systems created from other autonomous systems. There are two important architecture disciplines that are closely related to SoS architecture. First, there is *system architecture*, which is principally concerned with the people, activities, and technologies—including structure and behavior—that make up an autonomous system within an enterprise. While it will normally interact with other autonomous systems in the enterprise to maximize the capabilities of the enterprise, its core functions should not be dependent on those systems. Next, there is *enterprise architecture*, which is primarily concerned with organizational resources (people, information, capital, and physical infrastructure) and activities [4]. Figure 2.2 illustrates the SoS context. While the SoS architect must consider characteristics of component systems that comprise the SoS, the design of those systems is not their main focus. The SoS architect must also consider the larger enterprise context of the SoS, but the enterprise architecture is the primary concern of the enterprise architect. In fact it is often necessary for the SoS architect to consider multiple enterprises, since it is not uncommon for an SoS to cross enterprise boundaries.

Consider a practical SoS problem, illustrated in Figure 2.3, which will be used throughout the chapter. In this example a North American company acquires two other companies—one from the Pacific Rim and one from South America—to form a larger multinational corporation that will provide integrated wireless services for customers in North America, South America, and the Pacific Rim. The new integrated transcontinental wireless network (ITWN) will require the architecting of an SoS solution comprised

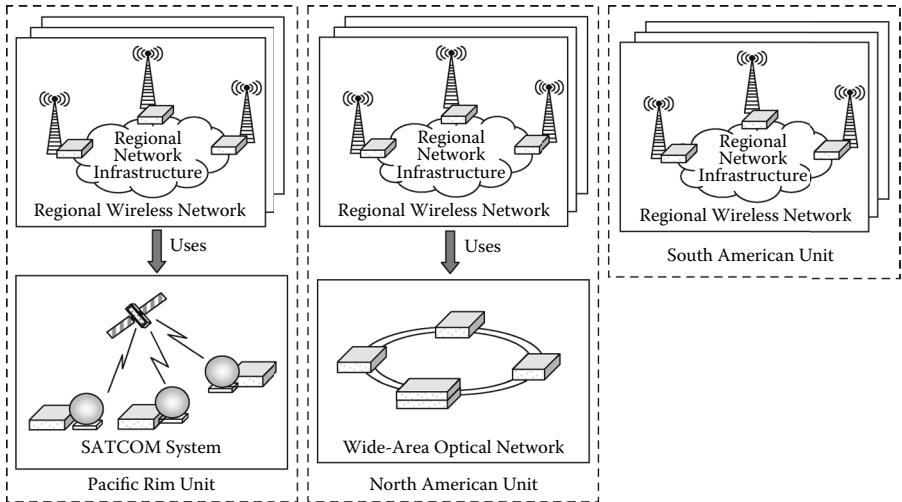


Figure 2.3 Integrated transcontinental wireless network.

of existing networks owned and operated by the newly formed units. Here is the SoS problem in a nutshell:

- Need:
 - Create a transcontinental network that provides integrated wireless services to customers in North America, South America, and the Pacific Rim.
 - Make use of existing resources owned by the newly formed enterprise.
- Resources:
 - Several wireless networks in North America, all connected via the wide-area optical network owned by the North American Unit.
 - Several wireless networks in the Pacific Rim, all connected via a satellite communications (SATCOM) system owned by the Pacific Rim Unit.
 - Several wireless networks in South America, all operated as independent, standalone networks by the South American Unit.
- Constraints:
 - Minimize changes to existing networks.
 - Continue to provide optical transport services in North America.
 - Continue to provide SATCOM services using the Pacific Rim SATCOM system

The various networks in the newly formed network will continue to serve their basic need. The North American optical network will continue to provide general transport services to customers, in addition to connecting the regional wireless networks; and the Pacific Rim SATCOM system

will continue to provide SATCOM services all over the Pacific Rim, in addition to connecting the regional wireless networks. Forming the ITWN will require creating an SoS solution that provides new capabilities created from the integration of existing networks, while continuing to serve the primary needs of the standalone networks.

This SoS example highlights many of the unique challenges of architecting SoS solutions—and there are many—but there are also some basic ideas that apply to all types of architecture design problems. After discussing those basic ideas—the architecture design process and architecture principles that apply equally to the architecting of all types of systems—we will discuss special considerations and critical success factors for architecting SoS solutions.

2.2 *Architecture principles and practices*

Architecture design begins with the recognition of a need, statement of the problem, and the articulation of a solution strategy. It continues with solution synthesis and analysis of alternatives. It ends with an architecture model—a blueprint of the system to be built [5]. While the process is pretty straightforward, it is complicated by the fact that design does not flow logically from needs. Design is a truly human endeavor. While all designs are ultimately driven by needs and tend to build on previous solutions, design involves compromise, and there is no simple recipe for making the necessary compromises [6]. There is, however, a well-defined process for architecting systems and a set of time-honored principles for navigating the solution space.

2.2.1 *The architecture design process*

The basic architecture design process, shown in [Figure 2.4](#), starts with analysis of needs, proceeds with solution synthesis, and completes with evaluation of the solution to meet the stated needs [7]. The architecture design process includes the following:

- Analysis—analysis of needs
- Synthesis—synthesis of solutions
- Evaluation—evaluation of solutions

2.2.1.1 *Analysis*

Sometimes the needs are well understood and even clearly communicated through requirements. But it is rare that all needs are understood, much less documented in the form of concise requirements. Regardless of whether the needs are communicated concisely and thoroughly via requirements or communicated using simple needs statements or operational concept descriptions, it is nonetheless the job of the architect to fully understand the needs.

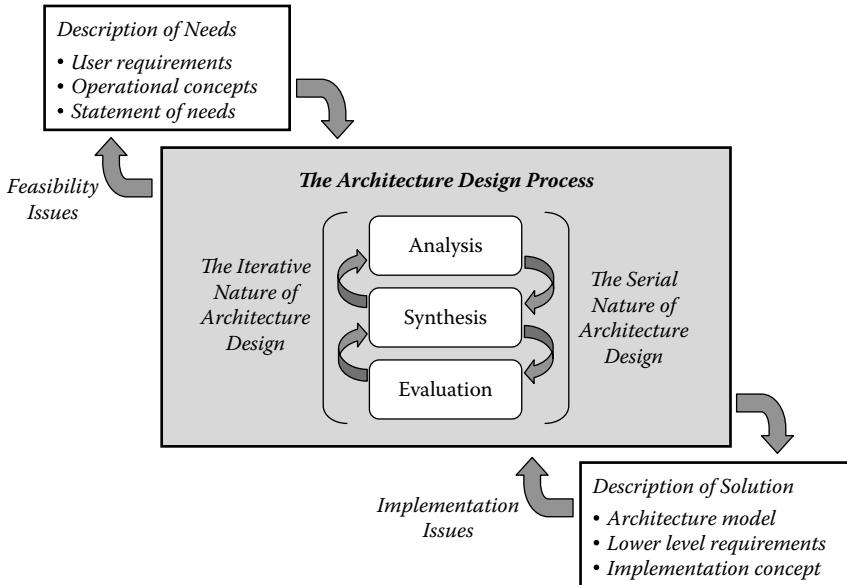


Figure 2.4 The architecture design process.

If the needs have been thoroughly examined, with most contradictory and infeasible requirements removed, and concisely communicated, the job of analyzing needs is certainly easier and less time consuming than might otherwise be the case. Moreover, it is not uncommon for stated needs to be based on perception or popular beliefs. While needs that are merely perceived cannot be ignored, the designer must be able to separate them from real needs. The analysis step is critical, and its omission generally results in risk to the remaining steps.

While understanding needs is critically important, understanding solution constraints cannot be overlooked. It is impossible to produce an effective design without fully considering the constraint space. In some cases constraints can drive the design just as much as needs. This is especially true for SoS, where solutions are based largely on existing systems and infrastructure. While these resources provide a rich base from which to construct new solutions, they also constrain the solution in a significant way.

2.2.1.2 Synthesis

The next step in the architecture design process is design synthesis. This step in the process is largely a transformation step, in which needs and constraints are transformed into solution designs. While the steps before and after are primarily analytical steps, synthesis is more creative. This is where the designer puts pen to paper and the innovative juices start to flow—where invention happens. Synthesis is where all the competing needs and vexing

constraints are merged into a solution. There is no recipe for this part; it is a truly human endeavor.

To get a true perspective on the nature of synthesis, let us imagine the lone designer sitting down to synthesize a solution. Even in this modern age, the tools are not that elaborate—perhaps a sheet of calculation paper, a cocktail napkin, a whiteboard—not really that different from inventors of the past. Their most important tool by far is the human mind and its ability to synthesize. The process of synthesis is a story of conflict. The designer must make tough decisions about which characteristics are more important than others, where to make trade-offs, where to stand firm on a particular aspect of the solution, and where to make compromises.

Now, imagine that the problem is too large for one person to tackle alone. Imagine a room filled with designers, each gathered around a whiteboard, each facing the same dilemma as the lone designer, each making critical decisions in their own mind about the necessary compromises, and not all coming to the same conclusion. It is a much tougher problem than the one faced by the lone designer. Now, imagine the real world. Not only is the problem usually too large for an individual designer to tackle it alone, but the skills needed to solve the problem tend to be geographically distributed. Now, each designer or team of designers is tackling the same dilemma, separated by space and often time. Now, add in the SoS perspective, where design decisions must account for many more factors, including operations and sustainment of existing systems, along with politics, economics, and risks that come with a complex SoS environment.

Synthesis does not end when the designers have come up with a solution. The solution has to be evaluated, which requires that it be modeled in a way that will support evaluation. And in fact, one design is not enough. Due to the subjective nature of synthesis and the natural limitations of the human mind, the goodness of a design can only be truly evaluated by comparing it to other designs [1]. The output of the synthesis process is a set of alternative solutions that are modeled with sufficient detail to allow evaluation to proceed.

2.2.1.3 Evaluation

The final step in the design process is evaluation. First and foremost is the evaluation of the design alternatives that resulted from design synthesis. From the perspective of the architecture design process, the means for evaluating the alternatives is less important than the fact that multiple design alternatives are identified and evaluated with respect to a set of criteria. The importance of evaluating multiple alternatives cannot be overemphasized—it is a cornerstone of the architecture design process.

However, evaluation of alternatives is not the only aspect of design evaluation; there are other concerns as well. The most significant among those concerns, the two that tend to overshadow all others, are cost and capability. It is not surprising that cost and capability tend to drive selection and refinement

of the final design. In fact, it is this fundamental economic problem of maximizing capability while minimizing cost that drives much innovation.

Aside from their ability to segregate the alternatives and allow the down-selection to a single “best” alternative, cost and performance evaluations are also likely to drive much of the design optimization. Evaluation is the stage in architecture design where critical refinements occur. Designs are simplified to eliminate unnecessary complexity, reduce costs, reduce risks, and improve dependability; lower-cost substitutes for component solutions are explored and evaluated; and most significantly, performance is optimized.

The architecture design process fits within a larger context of the systems engineering lifecycle, depending on inputs (description of needs) and producing outputs (solution description). But this larger process is also iterative, with implementation issues potentially resulting in architecture design rework and resolution of feasibility problems. It is not uncommon to discover during design synthesis that the necessary technologies for solutions to meet needs are too immature to be useful in an operational environment. And it is quite common to find that all solutions resulting from design synthesis fall short of meeting performance or cost targets.

The output of the architecture design process is the solution description. There are three critical aspects of the solution that must be described. The first aspect is the architecture model—the blueprint of the solution—which communicates the solution to those who will implement and use it. The second aspect is the lower-level requirements, including interface requirements, which constrain the design and implementation of entities within the architecture. The third aspect is the implementation concept, which describes the strategy for implementing the solution.

One of the most critical decisions in the architecture design process is to determine when design is complete. This decision is made difficult by the fact that, in any engineering problem, analysis, synthesis, and evaluation are never truly done. Design refinement continues and implementation problems continue to emerge until the system is delivered and is put into operation. In fact, they continue throughout the system lifecycle. But at some point the decision must be made to baseline the architecture and move to the next stage of lower-level design. The goal of the architecture design process is to do enough analysis, design, and evaluation to allow the later stages of design refinement and implementation to proceed with an acceptably low risk that the architecture will need to be significantly reworked.

2.2.2 *Architecture design principles*

Beyond the basic steps for performing architecture design discussed in the previous section, a set of guiding principles is needed. Mayhall lays out a comprehensive set of principles that govern all types of engineering design

[8]. Those principles provide the foundation for the four architecture design principles discussed here, principles that must be taken into account in the architecting of all systems. The four principles are:

- Needs often compete.
- Needs change over time.
- Resource availability constrains the solution space.
- Design compromise is necessary.

2.2.2.1 Needs often compete

The problem with needs is that they have a tendency to compete. That is, the full set of needs tend to call for solutions that compete with each other. In a motor vehicle, for instance, the need for fuel efficiency and power (both legitimate needs) tend to compete with each other. The designer has to deal with these competing needs and must do it in a way that strikes the right balance between fuel efficiency and responsiveness. Striking that balance involves sacrificing the optimal satisfaction of one need in order to satisfy some competing need to an acceptable degree.

2.2.2.2 Needs change over time

The viability of a solution cannot be evaluated without consideration of the circumstance that drives the need for it. Consider the problem of people communicating quickly, over long distances. That need was satisfied in the nineteenth century with the telegraph. While the telegraph was a pretty good solution to that problem in the nineteenth century, it would be completely inadequate today—the need has changed. While the basic need for long-distance communication still exists today, it has become much more elaborate, due in large part to the advancement of technology and expectations of users.

It takes time to design and build systems. Analysis, synthesis, and evaluation take time; implementation, testing, and deployment take time; and time is required for solutions to take shape. Moreover, it is the need, not at the time of conception or even at the time the system is placed into operation, that is really important, rather the need that is present in the prime of the system lifecycle. And it is not uncommon for systems to be in operation for decades before adequate resources are available to replace them. When a clear need is present at the same time as the available resources, circumstances are right for a potential solution. And when those circumstances persist for a sufficient length of time, one that is adequate for design and implementation to properly play out, then a solution to a problem can be realized. User need is both a necessary friend and an unwilling adversary when it comes to architecting systems.

2.2.2.3 *Resource availability constrains the solution space*

The design, implementation, operation, and sustainment of all systems depend on the availability of resources. The first, and arguably the most critical resource is money. Building complex systems is not cheap. Without capital investment, it is impossible to bring teams of people together to perform analysis, synthesis, and evaluation activities, procure the necessary technology components, integrate and test the system, put it into operation, and sustain it throughout its lifecycle.

But money alone is not enough. People possess another critical resource—knowledge and skill—with which it is impossible to design, implement, test, deploy, and operate complex systems. And it is not always possible to buy the required knowledge and skill. Consider early development of long-span suspension bridges [9]. While there was certainly a need for these bridges and probably the money to build them long before they were actually built, the required knowledge and skill was lacking, and the absence of that critical resource was an obstacle to their design and construction.

The availability of necessary technologies is also a critical resource. Network-centric systems—distributed systems that are connected via communication networks—would not be possible without high-speed computing platforms, high-bandwidth, low-latency communications networks and ubiquitous capacity in both. A special case of technology resources—availability of existing systems and infrastructure—is especially relevant to SoS architecture.

2.2.2.4 *Design compromise is necessary*

The final principle—the necessity for design compromise—is an inevitable result of the first three principles. Design is driven by needs—which have a tendency to compete and change over time—and is constrained by resource availability. Compromise is necessary to create a solution that strikes a balance among the competing needs; compromise is necessary to create a solution that is robust in the presence of changing needs; and compromise is necessary to deal with resource constraints. It is a dilemma, and there is no easy way out. The necessity for compromise is unavoidable.

A critical decision related to design compromise is the degree to which resource constraints will be allowed to drive the solution before attempting to change the constraint space. Sometimes the reliance on existing technologies overconstrains the solution space to the degree that needs cannot be met. It is often necessary in such cases to invest in a technology development initiative to shift the resource constraint space.

A related problem is the need to balance top-down analysis and synthesis with bottom-up analysis and synthesis. It is important to do top-down analysis to ensure that the most important problems are being adequately addressed and the design is not being driven by point solutions. However, there is a danger with ignoring bottom-up analysis [10]. The more constrained the technology resources space is, the more important it is to do thorough bottom-up analysis and synthesis. And since SoS architecture depends very

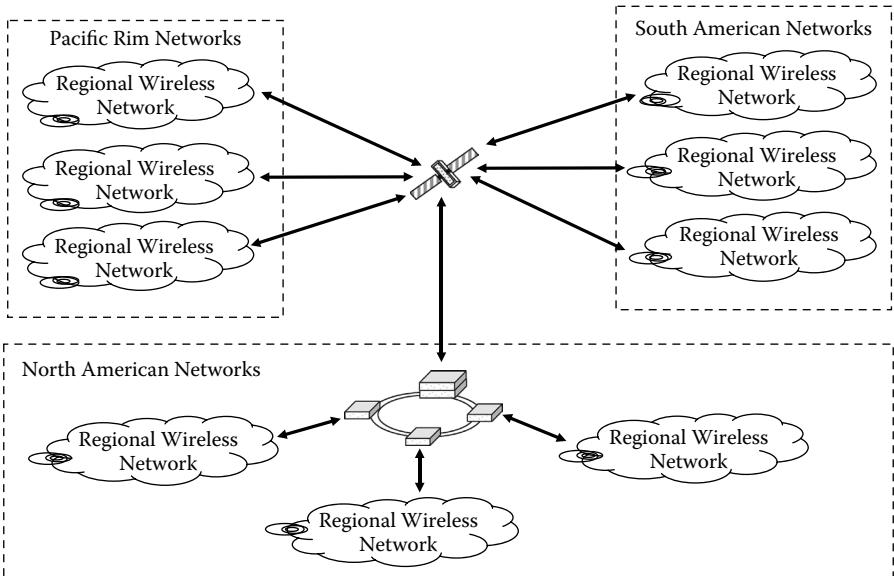


Figure 2.5 Transcontinental wireless network physical connectivity.

heavily on the use of existing systems and infrastructure, it is critical that adequate attention is given to bottom-up analysis.

Consider the ITWN example discussed earlier. The regional networks all need to be connected. An analysis of alternative solutions indicated that the most cost-effective approach would be to lease transport services from a major international carrier. But that would not meet the business need of leveraging the investment in the existing SATCOM system. So, the architecture decision to use excess capacity in the SATCOM system, illustrated in Figure 2.5, was in the end a design compromise to accommodate a business need.

The principle of compromise drives the architect to create a total system solution that integrates all design drivers, both needs and constraints; makes necessary compromises that strike a healthy balance among competing needs; and makes necessary design compromises to deal with constraints. In the end, since all designs are a result of compromise, there is really no perfect design, just those that are less flawed than others [6].

2.3 SoS architecture considerations

While the architecture design process and architecture design principles apply to system architecting at all levels, the architecting of SoS solutions requires some special considerations. Those considerations include the following:

- Autonomy
- Complexity

- Diversity
- Integration strategy
- Data architecture
- System protection

2.3.1 *Autonomy*

The SoS elements are themselves autonomous systems. They have their own stakeholders and their own mission in their respective enterprise; they have their own business processes, management organizations, and operating budgets; they are autonomous systems, and they need to maintain that autonomy even after SoS integration. Simply stated, SoS integration cannot do something that compromises the integrity of the systems that make up the SoS. There are two key aspects of system autonomy that must be preserved: technical autonomy and operational autonomy.

Technical autonomy is related to platforms, interfaces, and infrastructure of the existing systems. The integrity of external interfaces must be preserved, as they could be critical to the primary mission of the system. Those external interfaces may in fact have little to do with the SoS solution. Nonetheless, they have to be preserved. The integrity of system platforms also needs to be preserved. It is not uncommon for an SoS solution to appear sound, only to have later evaluation uncover the need for major unplanned upgrades to existing platforms. Such a situation is likely to place undue burden on sustainment of the existing system and could put its primary mission in jeopardy.

There are also infrastructure considerations. It is quite common for an SoS solution to require major unplanned infrastructure improvements, which will certainly affect capital budgets for operating and sustaining existing systems. It is often necessary to disrupt the technical autonomy of systems when they become part of a larger SoS but it is important that changes are planned and technical autonomy is reestablished.

Operational autonomy is related to organizations and business processes. Organizations within the enterprise are structured to operate and sustain systems. Those organizations have business processes for operating and sustaining their organic systems. There are also organizational relationships with other enterprises, related to the systems in each of those enterprises. Organizations, organizational relationships, and business processes are at the heart of the operational architecture of an enterprise. It is important to preserve the operational autonomy of systems when they become part of a larger SoS. Like technical autonomy, it is often necessary to disrupt the operational autonomy of individual systems in the SoS, but it is critical that the operational autonomy be reestablished.

2.3.2 Complexity

The use of existing systems to create SoS solutions introduces unavoidable complexities, both in terms of constraints and consequences. Existing systems and infrastructure provide a good starting point, since they are already in place, but they also constrain the solution. These constraints are not typically straightforward, and rarely are they well documented. There are often elements in place to support existing systems which have little value from an SoS perspective. Unfortunately, they add complexity to the SoS solution. It is a sort of tax for using existing systems and infrastructure.

Another aspect of complexity in the SoS is the natural specialization and optimization of systems to perform their primary function. The basic laws of economics result in systems that become specialized for solving their specific problem in the most cost-effective way. While specialization and optimization are good for autonomous systems, they result in different solution approaches that are often incompatible. Dealing with the incompatibilities typically requires the introduction of “bridges” into the SoS solution, which again add complexity.

Finally, there is the problem of fuzzy functional architecture partitions [11], the gaps and overlaps in functional responsibilities. The need to preserve technical autonomy often means that multiple systems in an SoS will perform similar or even identical functions, and often do so in very different ways.

2.3.3 Diversity

The systems that comprise an SoS are diverse. While that is often good from a robustness perspective—reducing common-mode weaknesses—it also creates challenges for the SoS architect. The first challenge is diversity of needs. Each system in an SoS was likely motivated by a certain set of needs, which tends to change over time. As the circumstances that drove the primary need changes, so does the business case for the system itself. Changing stakeholder needs for the systems within an SoS will likely lead to different evolutionary paths. Diverging needs of systems within an SoS can literally rip the SoS solution apart.

The second challenge is environmental diversity. Since each system in an SoS is likely to be managed separately, with separate budgetary constraints, political environments, and leaders, they are each subject to a different set of forces that shape their evolution. Like diverging needs, diverging environments can also be a source of great strain on any SoS solution.

2.3.4 Integration strategy

Normally, when a system is architected, the system is partitioned into the various entities, each having a core set of responsibilities within the system, and each designed to be integrated to form the overall system. That is

not typically true for an SoS. The SoS is made up of autonomous systems that were not principally designed as part of a component in larger system. Beyond the known external interfaces, it is unlikely they were designed with external integration in mind.

Three basic integration problems must be addressed in creating the SoS. The first and most obvious is physical integration. It is unlikely that all systems within the SoS use compatible interface protocols. While the rising demand for SoS solutions has (at least in part) led to a decrease in protocol diversity and an increase in standardization, there are still needs for different protocols. Systems evolve independently, with their own set of optimization and specialization drivers. The natural result is the use of different protocols, many of which are nonstandard. This physical integration problem has to be solved.

The second problem is functional integration. Since existing systems serve as building blocks for the SoS, functions performed by the different systems in the SoS must be integrated and deconflicted. Fuzzy functional partitions are a result of multiple systems carrying out similar or identical functions. Since systems within the SoS need to preserve their technical autonomy, functional isolation (or damping) is often required. Functional isolation involves isolating the functions of one system with the SoS from functions being performed by another system in the SoS. Functional damping involves muting certain aspects of system functions to allow systems to play well together.

The third problem is semantic integration. Semantic integration has to do with how signals or data are interpreted by different systems. For example, suppose an SoS consists of three different systems that each provide their *status* to each other. Each of them is likely to define *status* according to their own needs. Table 2.1 shows an example of the semantic integration problem using something as simple as *status*.

Table 2.1 Semantic Integration Example (Status Semantics)

System	Name for "Status"	Meaning of Status
System A	state	binary state (up or down)
System B	mode	operating mode (normal, degraded or offline)
System C	health	<p>detailed state of health:</p> <ul style="list-style-type: none"> • subsystem X status <ul style="list-style-type: none"> – on line or off line • subsystem Y status <ul style="list-style-type: none"> – subsystem state <ul style="list-style-type: none"> • on line or off line – subsystem connection status <ul style="list-style-type: none"> • connected or not connected • subsystem Z status <ul style="list-style-type: none"> – state (normal or initializing) – mode (control or monitor)

There are two basic solution strategies for integrating systems within the SoS, strategies that address physical, functional, and semantic integration aspects. The first strategy—and probably the most common—is *SoS bridging*, which involves introducing a new system that has the responsibility of dealing with physical, functional, and semantic integration aspects. The second strategy is the *SoS refactoring*, which involves modifying existing systems in such a way that makes bridging unnecessary. There are good reasons for adopting each of these strategies, but ultimately the SoS architect must decide which will be used.

Figure 2.6 illustrates *SoS bridging*, and Figure 2.7 illustrates *SoS refactoring*. The bridging approach, based on the combination of two well-documented design patterns (the adapter and the bridge [12]), has the advantage that it minimizes the modification to existing systems, with most physical, functional, and semantic integration requirements being satisfied by the SoS Bridge. But bridging adds complexity to the solution and can be burdensome from operations and sustainment perspectives. The refactoring approach typically minimizes SoS complexity, makes the SoS easier to operate, and increases sustainability but is usually more expensive in the near term and more disruptive to existing systems.

Most SoS architectures—at least initially—employ bridging. It offers a lower-risk approach to integrating systems and tends to be less expensive in the near term. But given that the result is often more complex and less operable/sustainable, it is necessary to evaluate the benefit of employing a refactoring strategy, either initially or at some time after the initial SoS is placed into operation.

Consider the application of bridging in the ITWN example. While North America and South America use compatible wireless protocols, the Pacific

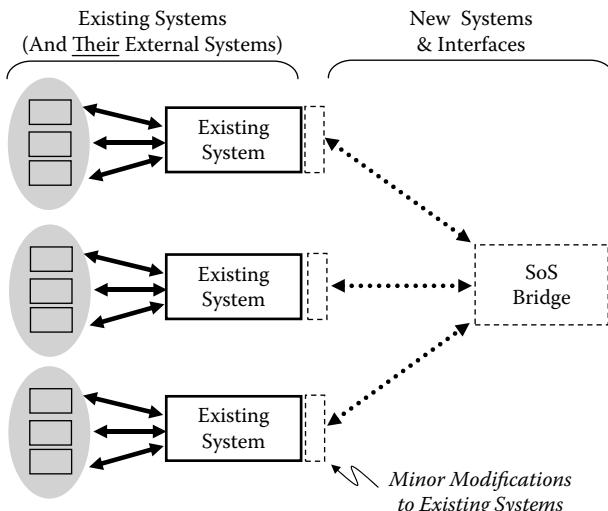


Figure 2.6 SoS bridging.

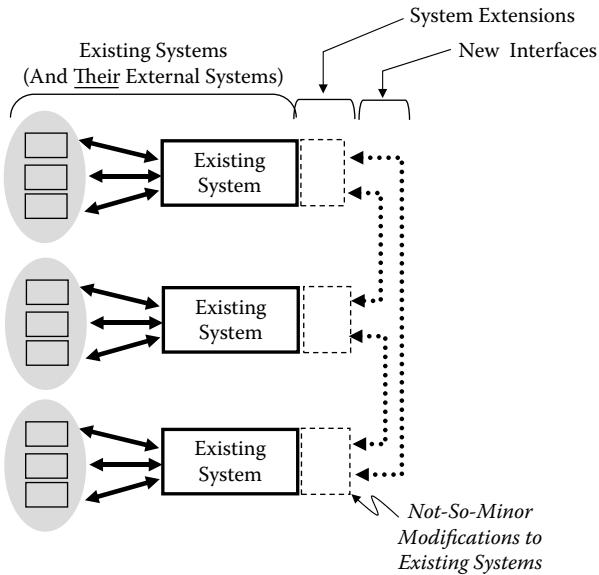


Figure 2.7 SoS refactoring.

Rim uses a different wireless protocol and will require the use of bridges (commonly called gateways in the telecommunications community) to allow integration of the different networks.

2.3.5 Data architecture

Most SoS solutions have two needs in terms of data architecture. First, it is necessary to ensure data consistency and semantics. Next, it is necessary to support the need for persistent storage of shared data—that is, information that is needed by more than one system in the SoS, even though it is owned by a single system within the SoS.

While a single data store for the entire SoS might seem like a good approach—least complex, lowest risk in terms of data integrity, and most economical to create and manage—it has two significant disadvantages, which limits its practical utility for SoS. First, a central data store would not preserve the autonomy of existing systems. Second, meeting the required levels of performance and availability is often difficult in an SoS environment. Three other data-storage strategies have emerged for SoS, models that do preserve system autonomy and make it easier to deal with performance and availability requirements.

The first strategy, which at first glance may not seem like much of a strategy at all, is the *uncoordinated data model* (shown in Figure 2.8). The uncoordinated data model is in fact not a bad approach. Although it requires that shared data be exchanged via traditional system-to-system interfaces and requires that

each system deal with data-structure and semantic problems in its own way, it is a simple and economical strategy. Its key disadvantages are the risk of problems with data structure and semantics and potentially high volumes of duplicate data being exchanged over the interfaces. Nonetheless, it is a practical strategy for SoS problems in which there is anticipated to be a low volume of shared data with a low risk of data structure and semantic problems.

The second strategy—the *coordinated data model*—mitigates one key aspect of the uncoordinated data model, the semantic problem. The coordinated data model (shown in Figure 2.9) involves a coordinated agreement on data format and semantics. This is the pattern used for most SoS problems in which there is anticipated to be a low volume of shared data. It has the simplicity of the uncoordinated data model but adds documented agreements on data structure and semantics. It is not uncommon to start with the uncoordinated data model and move to the coordinated data model as data structure and semantic problems emerge.

The third strategy—the *federated data model*—is the most sophisticated approach for SoS and is best applied when the volume of shared data is high and there is a high risk of data structure and semantic problems. The federated data model (shown in Figure 2.10) is the only strategy of the three that has a separate SoS data store outside of the existing systems. The SoS data

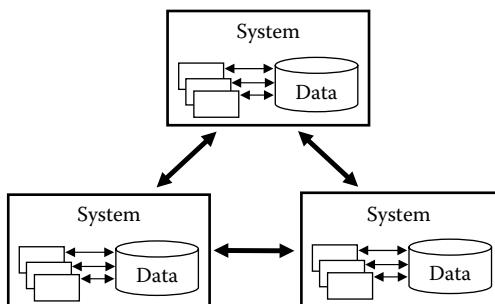


Figure 2.8 Uncoordinated data model.

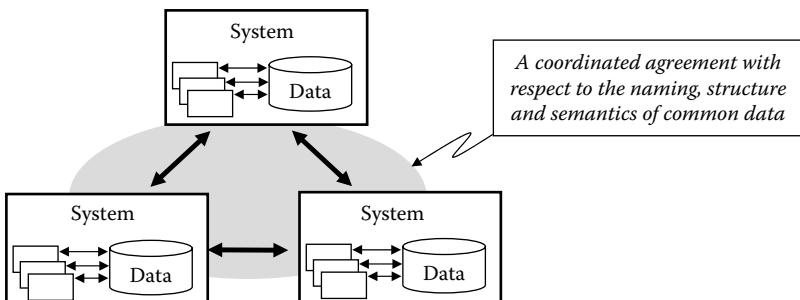


Figure 2.9 Coordinated data model.

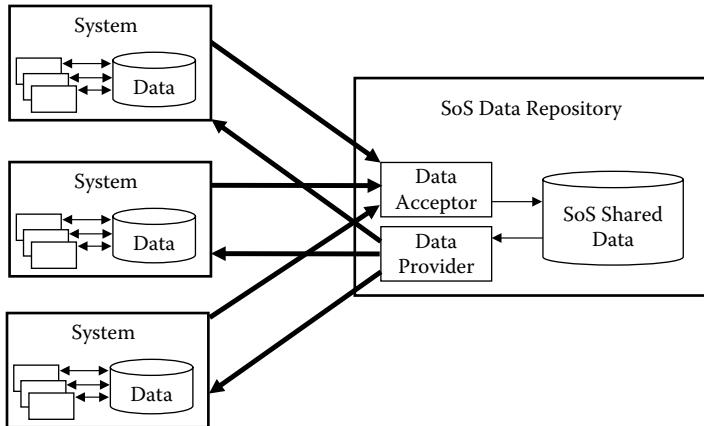


Figure 2.10 Federated data model.

repository contains shared data used by systems within the SoS. The shared data are typically owned by one system in the SoS and posted to the SoS data repository in an agreed-to format. It is not uncommon for the system owning the shared data to have its own internal format and post the agreed-to format in the SoS data store. (As a cautionary note, while it is tempting to post ALL SoS data in the enterprise data repository, it places an unnecessary burden on the SoS data repository and can often lead to an overly expensive and often unmanageable solution.)

2.3.6 System protection

Protecting systems within the SoS is an important consideration. Allowing systems to interact with each other, while continuing to prevent unauthorized access to system data and other resources, is no small task. Securing systems involves meeting four key security objectives [13]:

- Confidentiality—preventing unauthorized access
- Authentication—providing a means for identifying authorized users
- Integrity—restricting the unauthorized modification of resources
- Nonrepudiation—guaranteeing the identities of resource consumers and providers

But security is only one aspect of protection. Another aspect of protection is unintentional disruption by other systems within the SoS. Take the case where one system in the SoS provides a key function within the SoS, a function accessed by other systems within the SoS. It is possible for systems using the function to overload the system that provides the function. It is also possible for a fault in one system in the SoS to ripple throughout other systems within the SoS.

System isolation is a common technique employed for protection against unintentional disruption. Isolation involves introducing a separation layer between the internal subsystems of a system and external systems with which it interacts.

2.4 Critical success factors in architecting SoS solutions

Several factors are critical in successfully architecting SoS solutions. These factors can mean the difference between a successful solution and an unsuccessful one. The factors discussed here apply to traditional as well as SoS architecture, but are especially important for SoS. While by no means a comprehensive list, these factors serve as a list of things that should not be overlooked. They include:

- Robust design
- Architecture alignment
- Architecture governance
- Architecture description

2.4.1 Robust design

A robust system is one in which the system serves its intended purpose under the full range of environmental conditions [14]. There are three important aspects of architecture robustness for SoS, and these are related directly to the fact that systems within the SoS are diverse and need to maintain their autonomy.

The first aspect of robustness—business case robustness—is related to the fact that needs change over time. As needs for individual systems change, their role in the SoS can be affected. It is important, therefore, that the proper functioning of the SoS not be too sensitive to changes in the business case for each system within the SoS. For example, take the ITWN example, in which the SATCOM system is integral to the architecture. As the SATCOM system nears its end of life, it is not unthinkable that the SATCOM system would be abandoned and replaced by leased transport services. A robust design would be insensitive to changes in the transport mechanism.

The second aspect of robustness—schedule robustness—is related to the ability of a system within the SoS to provide a necessary capability on time. It is not uncommon for system improvements to be delayed for technical or financial challenges. If the planned improvement of one of the systems within the SoS is a critical capability for the SoS, the design is not very robust. A more robust design from a schedule-robustness perspective would be to have a contingency approach that meets the critical need of the SoS.

The third aspect of robustness—technological robustness—is related to the technological environment. Take the case where the SoS solution is based on a standardization of communications protocol within the SoS. In this example, an analysis of alternatives might have indicated that a particular communications

protocol was best. However, over time a better protocol emerged, and systems within the SoS started migrating to the new protocol. A robust design would assume that there would exist within the SoS a variety of different protocols that systems within the SoS would need to accommodate.

2.4.2 Architecture alignment

While it is a necessity that systems within the SoS maintain their autonomy, it is nearly impossible to create or improve an SoS solution without some disruption. The SoS architect must assume that some disruption will occur and plan for realignment to reestablish architectural resonance. There are three important aspects of architecture alignment that need to be considered. The first aspect—organizational alignment—involves realigning organizations to function within the SoS context. The second aspect—business process alignment—involves updating business processes and procedures to function within the SoS context. The third aspect—technological alignment—involves aligning technological aspects.

Consider the ITWN example in which all three are required. Each of these systems has organizations for doing service provisioning and network management, but none were designed to interoperate with other organizations. Each of those organizations will have to make internal changes to accommodate the SoS architecture. Moreover, since each system has business processes and procedures for provisioning a service—and none of those processes or procedures accounts for interaction with other organizations—processes and procedures will need to be realigned to function effectively within an SoS context. Finally, each system might have a fundamentally different meaning for the term communication service—not just a semantic difference but also a substantial difference in terms of technical implementation. These technical differences will need to be addressed.

2.4.3 Architecture governance

While it is important for systems within an SoS to maintain their autonomy, it is not practical to allow uncoordinated changes to occur. When a system becomes part of an SoS, it becomes part of a larger federation of systems, and that federation must have rules all members of the federation agree to honor. Those rules form the basis of architecture governance. Without governance, maintaining the proper functioning of an SoS is nearly impossible.

There are two aspects of architecture governance relevant to SoS. The first aspect—governance of roles and responsibilities—is related to dealing with the fuzzy-partition problem. As individual system needs change over time, their role within the SoS might need to be revisited. Having mechanisms in place for managing roles and responsibilities allows for change to occur, but not in an uncoordinated way.

The second aspect of architecture governance—interface governance—is related to the details of system-to-system interfaces. In the case of SoS, these can also become fuzzy. Consider the example in which one system in the SoS is posting shared data. Any system within the SoS can retrieve the shared data. When a decision is made to change the structure or semantics of the shared data (in even the smallest way), it is important that the change is coordinated with all users of the data. Those affected include not only those currently using the data but also those that are in the process of making changes to their internal systems to use the data in the future.

2.4.4 *Architecture description*

As systems become more and more complex, it becomes increasingly important to represent the architecture using a well-defined model. The architecture model provides a means for performing analysis of the system structure and behavior and also provides a roadmap for those implementing the architecture. Moreover, as the number of people involved with the analysis, synthesis, evaluation, implementation, and operation of the system increases, describing the architecture using a well-defined set of models and semantics becomes very important.

Architecture descriptions are necessarily done from multiple viewpoints—no single viewpoint of the architecture adequately captures even the most salient aspects of the architecture of a system. (Architecture description of complex systems has become very similar to the multiview drawings used in building architecture.) The multi-view architecture approach is acknowledged to have started with the Zachman framework [15] in the late 1980s. This multiview approach has since been formalized by the Institute of Electrical and Electronics Engineers [1]. Although the Zachman framework was the first real architecture framework, it has spawned several other frameworks.

There are clearly benefits to using an architecture framework. Architecture frameworks provide a roadmap for describing the architecture of a system. They serve as something of a checklist for all the concerns that may need to be addressed during architecture design. They also provide a rich set of semantics for representing the architecture, as well as providing metadata for the architecture description itself.

But there are also some dangers. Architecture frameworks are designed to be comprehensive. As a result, they specify a comprehensive set of views. In reality, few architecture descriptions need to capture everything specified in a framework—tailoring is almost always necessary. Failing to adequately tailor the framework can result in overspecification and too much time being spent describing the architecture, with too little time spent on the most important aspects of the design. Another danger of architecture frameworks is that they can lead to a false sense of adequacy; the result is a well-described but poorly designed solution.

2.5 Architecture frameworks

We end this chapter with a discussion of some specific architecture frameworks. Each of these frameworks was designed to meet a specific need. While some of these are more suited to SoS architecture than others, all of them can be used. The frameworks are listed below:

- The Zachman Framework
- The Department of Defense Architecture Framework (DoDAF)
- The Ministry of Defence Architecture Framework (MoDAF)
- The Federal Enterprise Architecture (FEA) Framework
- The Rational Unified Process (RUP)
- The Open Group Architecture Framework (TOGAF)

The structures of the three most significant frameworks—Zachman, DoDAF and MoDAF—are discussed in more detail.

The Zachman Framework, introduced in 1987 [16] was the first real architecture framework that is applicable to SoS architecture. Aside from its continued use today, the Zachman Framework (shown in [Table 2.2](#)) served as the foundation for every other architecture framework discussed in this chapter.

The Office of the Secretary of Defense (OSD) maintains DoDAF, which is used across the United States Department of Defense for describing architectures. Its structure (shown in [Table 2.3](#)) [17] is very good for describing SoS architectures. It has three core views—operational view, system view and technical standards view—and several view products within each view.

The United Kingdom (U.K.) Ministry of Defence modeled MoDAF after the DoDAF model. Notice that its structure (shown in [Table 2.4](#)) [18] is very similar to DoDAF. In addition to the core views contained in DoDAF, MoDAF adds two additional core views: the strategic view and the acquisition view. MoDAF is another excellent framework for representing SoS architectures.

The Federal Enterprise Architecture (FEA) Framework, maintained by the United States Office of Management and Budget, is used to represent enterprise architectures in the various nondefense agencies within the United States government [19]. While the FEA framework can be used to represent SoS architectures, it is most useful for representing enterprise architectures.

While DoDAF, MoDAF, and FEA were all created for government use in the United States and the United Kingdom, they are all freely available and well-documented frameworks and should be considered for use in representing SoS architectures in all types of enterprises, private enterprises as well as government enterprises.

The Rational Unified Process (RUP), maintained and licensed by IBM, is widely used to represent enterprise software architectures [20]. While RUP does specify some specific architecture views, it is more of a software development process framework than an architecture framework.

Table 2.2 The Zachman Framework

Addresses ⇒		What	How	Where	Who	When	Why
Describes ⇒		Data	Function	Network	People	Time	Motivation
Layer 1	Scope	Business entities	Business functions	Business locations	Organizations & enterprises	Business events	Business goals
Layer 2	Business model	Semantic model	Business process model	Business logistics	Organization chart	Master schedule	Business plan
Layer 3	System model	Logical data models	Application architecture	Distribution architecture	Human interface architecture	Processing structure	Business rules
Layer 4	Technical model	Physical data models	System design	Technology architecture	Presentation architecture	Control structure	Rule design
Layer 5	Detailed representation	Database schemata	Source code	Network architecture	Security architecture	Timing definition	Rule specification

Table 2.3 The DoD Architecture Framework

View	View Description	View Product	View Product Name	View Product Description
All views	Aspects that crosscut operational, systems and technical standards views	AV-1 AV-2	Overview & summary information Integrated dictionary	General scope, purpose and intended users Data dictionary that contains terms and definitions used in all view products
Operational view	Describes operational nodes, operational activities and operational information	OV-1	High-level operational concept graphic	High-level description of the operational concept
		OV-2	Operational node connectivity description	Operational nodes, connectivity and information needs between nodes
		OV-3	Operational information exchange matrix	Identifies the information exchanged between nodes along with relevant attributes of the exchange
		OV-4	Organizational relationships chart	Organizational, role or other relationships among organizations
		OV-5	Operational activity model	Capabilities and operational activities along with activity interrelationships
		OV-6a	Operational rules model	Describes business rules that constrain operation
		OV-6b	Operational state transition description	Identifies business process response to events or conditions
		OV-6c	Operational event-trace description	Traces actions in a scenario or sequence of events
		OV-7	Logical data model	Describes the system data

Systems view	Describes system, service and interconnection functionality and relationships	SV-1	System/service interface description	Identifies system nodes (or services) and interconnections
		SV-2	System/service communications description	Defines the communications infrastructure for supporting system/service interconnections
		SV-3	System-system matrix System-service matrix Service-service matrix	Relationships among systems and services in the architecture
		SV-4a	Systems functionality description	Describes functions performed by the system and defines data flows among functions
		SV-4b	Services functionality description	Describes functions performed by the services and defines data flows among functions
		SV-5a	Operational activity to systems function traceability matrix	Mapping of systems functions in the system view to operational activities in the operational view
		SV-5b	Operational activity to systems traceability matrix	Mapping of systems in the system view back to operational activities in the operational view
		SV-5c	Operational activity to services traceability matrix	Mapping of services in the system view back to operational activities in the operational view
		SV-6	Systems/services data exchange matrix	Describes details of data being exchanged between system/services along with the attributes of the exchange

(continued)

Table 2.3 The DoD Architecture Framework (continued)

View	View Description	View Product	View Product Name	View Product Description
Technical standards view	Identifies the policies and standards that govern the system architecture	SV-7	Systems/services performance parameters matrix	Describes the performance characteristics of systems/services
		SV-8	Systems/services evolution description	Planned evolution of systems/services
		SV-9	Systems/services technology forecast	Emerging technologies that are expected to affect the architecture along with their anticipated timeframes
		SV-10a	Systems rules model	Describes the constraints placed on systems/services
		SV-10b	Systems state transition description	Identifies systems/services response to events or conditions
		SV-10c	Systems event-trace description	Traces actions in a scenario or sequence of events
		SV-11	Physical [data] schema	Physical implementation of the logical data model—message formats, file structures, physical schema, etc.
		TV-1	Technical standards profile	Listing of applicable standards or policies that constrain the architecture
		TV-2	Technical standards forecast	Description of emerging standards (or changes to existing standards) that are expected to impact the architecture along with their timeframes

Table 2.4 Ministry of Defence Architecture Framework

View	View Description	View Product	View Product Name	View Product Description
All views	All View products provide information pertinent to the entire Architecture	AV-1	Overview & summary information	Includes assumptions, constraints, and limitations that may affect high-level decisions relating to an architecture
		AV-2	Integrated dictionary	Presents all the elements used in an architecture as a stand alone structure
Strategic views	The Strategic Views support the capability management process	StV-1	Enterprise vision	Addresses the enterprise concerns associated with the overall vision and defines the strategic context for a group of enterprise capabilities
		StV-2	Capability taxonomy	Describes the taxonomy of capabilities
		StV-3	Capability phasing	Describes the planned evolution of capabilities over time
		StV-4	Capability dependencies	Describes dependencies between planned capabilities
		StV-5	Capability to organizational deployment mapping	Describes the planned deployment of capabilities in an organizational context
		StV-6	Operational activity to capability mapping	Provides a mapping between the capabilities in the strategic view to operational activities in the operational view

(continued)

Table 2.4 Ministry of Defence Architecture Framework (continued)

View	View Description	View Product	View Product Name	View Product Description
Operational view	Describes operational nodes, operational activities and operational information	OV-1a	High-level operational concept graphic	Graphic representation of the high-level operational concept
		OV-1b	Operational concept description	Provides a supplementary textural description that explains and details the scenario contained within the associated high level operational concept graphic
		OV-1c	Operational performance attributes	Provides detail of the operational performance attributes associated with the scenario represented in the high level operational concept graphic
		OV-2	Operational node relationships description	Operational nodes, connectivity and information needs between nodes
		OV-3	Operational information exchange matrix	Identifies the information exchanged between nodes along with relevant attributes of the exchange
		OV-4	Organizational relationships chart	Organizational, role or other relationships among organizations
		OV-5	Operational activity model	Capabilities and operational activities along with activity interrelationships
		OV-6a	Operational rules model	Describes business rules that constrain operation

Systems view	Describes system, service and interconnection functionality and relationships	OV-6b	Operational state transition description	Identifies business process response to events or conditions
		OV-6c	Operational event-trace description	Traces actions in a scenario or sequence of events
		OV-7	Information model	Address the information perspective of an operational architecture
		SV-1	Resource interaction specification	Address the composition and interaction of resources
		SV-2a	System port specification	Specifies the ports on a system, and the protocols used by those ports when communicating with other systems
		SV-2b	System port connectivity description	Specifies the communications links between systems and may also list the protocol stacks used in connections
		SV-2c	System connectivity clusters	Defines how individual connections between systems are grouped into logical connections between parent resources
		SV-3	Resource interaction matrix	Provides a tabular summary of the resource interactions specified in the SV-1 for the architecture
		SV-4	Functionality description	Address human and system functionality
		SV-5	Function to operational activity traceability matrix	Addresses the linkage between functions described in SV-4 and operational activities specified in OV-5

(continued)

Table 2.4 Ministry of Defence Architecture Framework (continued)

View	View Description	View Product	View Product Name	View Product Description
		SV-6	Systems data exchange matrix	Specifies the characteristics of the system data exchanged between systems
		SV-7	Resource performance parameters matrix	Depicts the performance characteristics of a resource (system, role or capability configuration)
		SV-8	Capability configuration management	Presents a whole lifecycle view of a resource, describing how its configuration changes over time
		SV-9	Technology & skills forecast	Provides a summary of emerging technologies and skills that impact the resources that constitute the architecture
		SV-10a	Resource constraints specification	Specifies functional and nonfunctional constraints on the implementation aspects of the architecture (i.e. the structural and behavioral elements of the SV viewpoint).

		SV-10b	Resource state transition description	A graphical depiction of a resource (or function) response to various events by changing its state
		SV-10c	Resource event-trace description	Provides a time-ordered examination of the interactions between functional resources
		SV-11	Physical [data] schema	Defines the structure of the various kinds of system data that are utilized by the systems in the architecture
Technical standards view	Identifies the policies and standards that govern the system architecture	TV-1	Standards profile	Listing of applicable standards or policies that constrain the architecture
		TV-2	Standards forecast	Description of emerging standards (or changes to existing standards) that are expected to impact the architecture along with their timeframes
Acquisition view	Provides programmatic details, including dependencies between projects and capability integration	AcV-1	Acquisition clusters	Provides an organizational perspective on programs
		AcV-2	Program timelines	Provides a timeline perspective on programs

The Open Group Architecture Framework (TOGAF) [21] is maintained and licensed by The Open Group. While TOGAF does call out some high-level viewpoints, it does not provide a specification for describing the architecture, stressing instead the methodology and tools for doing architecture design.

A topic closely related to architecture frameworks is modeling languages. The Unified Modeling Language (UML™) is a general modeling language created to represent software-intensive systems [22]. While it has been used to represent general systems, it is best suited for modeling software architectures. The extension of UML to better represent system problems led to the creation of SysML™ [23]. The creation of SysML is much more than a mere tailoring of UML; it is a much more comprehensive modeling language for describing systems. While neither UML nor SysML are architecture frameworks, they both specify a modeling language that can be used in architecture frameworks to create specific view products or models.

References

1. ANSI/IEEE 2000. Recommended Practice for Architecture Description of Software-Intensive Systems. Institute of Electrical and Electronics Engineers, 1471-2000.
2. Asimow, M. 1962. *Introduction to Design*. Prentice-Hall, Englewood Cliffs, NJ.
3. Cole, R. 2006. The changing role of requirements and architecture in systems engineering. *Proceedings of the First Annual IEEE Conference on System of Systems Engineering*.
4. Rouse, W. 2005. Enterprises as systems: essential challenges and approaches to transformation. *Systems Engineering Journal* 8(2):138–150.
5. Haskins, C. et al. 2006. *INCOSE Systems Engineering Handbook*. International Council on Systems Engineering.
6. Petroski, H. 2003. *Small Things Considered—Why There Is No Perfect Design*, Vintage Books, New York.
7. Faulconbridge, R. et al. 2003. *Managing Complex Technical Projects—A Systems Engineering Approach*. Artech House, Boston, MA.
8. Mayhall, W. 1979. *Principles in Design*. Van Nostrand Reinhold Company, New York.
9. Petroski, H. 1992. *To Engineer is Human: The Role of Failure in Successful Design*. Vintage Books, New York.
10. Char, G. 2006. Balancing abstraction and implementation constraints when specifying problems. *Proceedings of the First Annual IEEE Conference on System of Systems Engineering*.
11. Meilich, A. 2006. System of systems (SoS) engineering and architecture in a net centric environment. *Proceedings of the First Annual IEEE Conference on System of Systems Engineering*.
12. Johnson, R. et al. 1995. *Design Patterns*. Addison-Wesley, Reading, MA.
13. Warkentin, M. and R. Vaughn. 2006. *Enterprise Information Systems Assurance and Systems Security: Managerial and Technical Issues*, IGI Publishing, Hershey, PA.
14. Clausing, D. and D. Fry. 2005. Improving system reliability by failure-mode avoidance including four concept design strategies. *Systems Engineering Journal* 8(3):245–261.

15. Khoshafian, S. 2000. *Service Oriented Enterprises*. Auerbach Publications, Boston, MA.
16. Zachman, J. 1987. A framework for information systems architecture. *IBM Systems Journal* 26(3):276–292.
17. United States Department of Defense. 2007. DoD Architecture Framework Version 1.5, United States Department of Defense.
18. U.K. Ministry of Defence. 2007. The MOD Architecture Framework Version 1.1, U.K. Ministry of Defence.
19. U.S. Office of Management and Budget. 2006. FEA Consolidated Reference Model Document Version 2.1, Office of Management and Budget.
20. McGovern, J. et al. 2004. *A Practical Guide to Enterprise Architecture*. Prentice-Hall, Englewood Cliffs, NJ.
21. Dargan, P. 2005. *Open Systems and Standards for Software Product Development*. Artech House, Boston, MA.
22. Holt, J. 2004. *UML™ for Systems Engineering: Watching the Wheels*. 2nd Edition, Institute of Electrical Engineers.
23. Object Management Group. 2006. OMG SysML™ Specification, The Object Management Group.

chapter three

*Emergence of SoS, sociocognitive aspects**

Beverly Gay McCarter and Brian E. White

Contents

3.1	Introduction.....	72
3.2	Scale or view	75
3.2.1	Definition of view.....	77
3.2.2	Mindset range: finite or infinite?.....	81
3.2.3	Multiview analysis	83
3.3	Emergence	85
3.3.1	Definitions of emergence.....	85
3.3.2	Examples of emergence	88
3.3.3	Beneficial emergence.....	89
3.3.4	Emergence and prediction	90
3.3.5	Emergence and entropy.....	91
3.3.6	Emergence and surprise.....	91
3.3.7	Emergence in system of systems.....	93
3.4	Conclusion	102
	References	102

This chapter offers a human-centric treatment of the concepts of multi-scale analysis and emergence in system of systems (SoS)[†] engineering,

* Portions reprinted, with permission, from B. E. White, "On Interpreting Scale (or View) and Emergence in Complex Systems Engineering," 1st Annual IEEE Systems Conference, Honolulu, HI, 9–12 April 2007. © 2007 IEEE.

† The authors define a system as an interacting mix of elements forming an intended whole greater than the sum of its parts. *Features*: These elements may include people, cultures, organizations, policies, services, techniques, technologies, information/data, facilities, products, procedures, processes, and other human-made or natural entities. The whole is sufficiently cohesive to have an identity distinct from its environment. An SoS is defined as a collection of systems that functions to achieve a purpose not generally achievable by the individual systems acting independently. *Features*: Each system can operate independently and is managed primarily to accomplish its own separate purpose. An SoS can be geographically distributed, and can exhibit evolutionary development and/or emergent behavior [1].

or more generally, complex systems engineering (CSE) [2]. This includes a characterization of what an individual might do in conceptualizing a given systems engineering situation s/he is facing. The authors suggest fresh interpretations of the terms *scale* and *emergence* that will contribute to a more collaborative approach to improving the CSE practice. Because other authors use “scale” in several different ways, potentially causing confusion, the present authors propose “view” instead. Here a given view is defined as a combination of “scope,” “granularity,” “mindset,” and “time-frame.” Although “emergence” has a rich spectrum of definitions in the literature, the authors prefer to emphasize the unexpected, especially “surprising,” flavor or emergence. In this endeavor sociocognitive aspects are paramount, and in an age of increasing organizational complexity, small group dynamics are becoming more significant. Human psychological and social behaviors that impact information sharing, productivity, and system interoperability need closer examination as organizations increasingly rely on decentralization to achieve their goals. Therefore, this chapter also highlights important dynamic social issues, although the authors do not focus on trying to solve them. Rather, areas of further research are suggested which can lead to better CSE, when people are considered part of any (complex) system in the SoS.

3.1 *Introduction*

It is important to bring a degree of humility when approaching complex systems engineering (CSE) problems to be solved or at least mitigated. This is not a sign of weakness,* but a quality that can serve individuals very well. More specifically, one needs to be aware and accepting of the wide variety of views of reality taken by different individuals and cultures around the world. One important dynamic issue that groups face is the difficulty of members to fully understand that other group members may not share their view of how the world is, or how they think it should be. Lack of such awareness and understanding causes problems for both intra- and intergroup interactions. Among the many factors that contribute to

* Even Albert Einstein was known as humble, not only in contemplating the universe but also toward himself and his view of humanity: “Adding to his aura was his simple humanity. His inner security was tempered by the humility that comes from being awed by nature.... ‘It is rather naïve and imperfect, as might be expected from such a young fellow like myself’ Einstein confessed with a pretense of humility.... His religious feelings of awe and humility also informed his sense of social justice.... ‘What separates me from most so-called atheists is a feeling of utter humility toward the unattainable secrets of the harmony of the cosmos,’ he explained.” [3, pp. 5, 25, 385, and 389, respectively]

one's unique world view are psychological, sociological, and even biological factors.*

This has implications concerning why some people, in general, cannot understand or are less willing to accept others' perceptions of reality. This is about the difficulty people—almost all people—have in fully embracing, not just tolerating, the fact that no two people think or view the world in precisely the same way. If something is outside one's life experience, it is very difficult for that person to envision it. It may be too "foreign" or contrary to those things they do perceive. This is a bit like trying to envision infinity; people live life in a linear fashion with a start and a finish. Therefore, typically it is very hard (or impossible) to imagine infinity. Again, consider "instinct" as being patterns of behavior established over time through repetition. Thus, it is hard to break such ingrained response patterns. Couple that with the way the brain is hardwired, and one can begin to see the difficult underlying dynamics involved with group interactions, perhaps putting in question many management theories proposed over the course of the past several decades. In Section 3.3, the authors will explain how leaders need to bring this awareness to their own complex systems arena, i.e., their organization.

To help facilitate progress, particularly in addressing the influences of human nature, one must recognize that each person, as a consequence of finite human brains and the limitations of human minds, sees a different perception of any underlying (infinite) reality.[†] No one person can see the

* "How does the brain, with its diverse and distributed functions, come to arrive at a unified sense of identity?... Could we speak of a person's brain without, ultimately, speaking of the person?... Belief in an inner essence, or central core, of personhood, was called 'ego theory.' The alternative, 'bundle theory,' made more neurological sense but offended our deepest intuitions.... [i.e.] An embodied brain acts, thinks, has certain experiences, and that's all.... The idea that certain forms of insanity were 'disorders of the self' had been around for two centuries and more, but now the concept was being refined. The core deficits of autism and schizophrenia, for example, were revealed as faults in the brain circuits underlying personal awareness.... In the process, it gave definition to that fundamental unit of social intercourse: the person. Just as the brain had evolved systems for guiding interaction with the physical world so, we rather belatedly realised, it had also evolved specialised mechanisms for enabling the interaction of 'self' and 'other.'" [4]

† There seems to be little doubt that reality exists; it is just that one can debate whether anyone can claim to know reality directly. Again, some quotes about Einstein are of note: "For all practical purposes, the physical reality of atoms and molecules was now conclusively proven. 'At the time atoms and molecules were still far from being regarded as real,' the theoretical physicist Max Born later recalled. 'I think that these investigations of Einstein have done more than any other work to convince physicists of the reality of atoms and molecules.'... Einstein's fundamental dispute with the Bohr–Heisenberg crowd... was about reality. Does it exist? More specifically, is it meaningful to speak about a physical reality that exists independently of whatever observations we can make? 'At the heart of the problem,' Einstein said of quantum mechanics, 'is not so much the question of causality but the question of realism.'" [3, pp. 106 and 460–461, respectively]

Gestalt, or complete/full view of complexity. People can only see one slice or narrow perspective of it at once. However, how narrow or broad that slice is depends on the inherent abilities of the perceiver. Different people see different degrees of reality. One goal for easing interpersonal conflict is to not only establish a tolerance for the fact that different people adamantly subscribe to different views of reality, but to nurture abilities to actually “see” and accept that all views of reality make up the totality of an infinite and complex reality. This “realization” is not only relevant to individuals and leaders, but to organizations and groups, as the authors will discuss in Section 3.3.

People should deal with ambiguous barriers of terminology before real progress can be made in exchanging, correctly interpreting, and acting on each other’s ideas. This process can do much to establish a feeling of trust, a complex condition so necessary for true collaboration between individuals and among groups. Again, notions of trust will be elaborated in Section 3.3.

Ryan [5] explains that ontology* is about “the study of being or reality or existence,” and that epistemology is about “the study of knowledge.” Thus, an epistemic “property depends on how reality is conceptualized, and that is always relative to an observer and subjective.” Because of its psychological theme, this chapter focuses on epistemic aspects of CSE and deemphasizes the more scientific ontological counterparts.[†] That is, the authors emphasize observation and subjectivity as opposed to reality and objectivity.[‡]

* “As a young empiricist, excited by his readings of Ernst Mach, Einstein had been willing to reject any concepts that could not be observed, such as the ether and absolute time and space and simultaneity.... In his maturity, Einstein more firmly believed that there was an objective ‘reality’ that existed whether or not we could observe it. The belief in an external world independent of the person observing it, he repeatedly said, was the basis of all science.” [3, pp. 333–334]

† “Bohr and his adherents scoffed at the idea that it made sense to talk about what might be beneath the veil of what we can observe. All we can know are the results of our experiments and observations, not some ultimate reality that lies beyond our perceptions.’... Einstein told Pauli that he still objected to the fundamental tenet in quantum mechanics that a system can be defined only by specifying the experimental method of observing it. There was a reality, he insisted, that was independent of how we observed it. ‘Einstein has the philosophical prejudice that a state, termed “real,” can be defined objectively under any circumstances, that is, without specification of the experimental arrangement used to examine the system,’ Pauli marveled in a letter to Max Born.” [3, pp. 460–461 and 538, respectively]

‡ Later in life Einstein gave more emphasis to ontology, especially in the context of quantum mechanics, which he believed was incomplete. “A new fashion has arisen in physics,” Einstein complained, which declares that certain things cannot be observed and therefore should not be ascribed reality....there is no single underlying reality that is independent of our observations. ‘It is wrong to think that the task of physics is to find out how nature is,’ Bohr declared. ‘Physics concerns what we can say about nature.’ This inability to know a so-called ‘underlying reality’ meant that there was no strict determinism in the classical sense.... Einstein never fully came around, even as experiments repeatedly showed quantum mechanics to be valid. He remained a realist, one who made it his creed to believe in an objective reality, rooted in certainty, that existed whether or not we could observe it.... When the talk turned to quantum mechanics, he

3.2 Scale or view

Kuras and White [6,7] asserted that multiscale (multiview) analysis is crucial to more effective CSE. It should not be unexpected that a number of carefully chosen perspectives can reveal, albeit sometimes surprisingly, patterns that help one better understand SoS, or more generally, complex systems and enterprises.* These different views, together, can elicit ideas for influencing or shaping the environment of a complex system to help guide or shape it toward more useful capabilities.

The increasing speed of change in many complex systems dictates that one cannot “troubleshoot” problems[†] and engineer solutions as in the past. One must be able to harness multiple views to see the *Gestalt*, or big picture, including a system’s (or SoS’s) environment, and suggest ways to try to influence that environment or at least shape the system’s further evolution toward better mission capabilities. A “realized” enterprise reinvents itself through a process of continual innovation and integration that moves toward higher levels of evolutionary progress, which also can be thought of as increased complexity (see [Figure 3.1](#)). [10]

once again tried to poke holes in the idea that our observations can affect and determine realities. ‘When a mouse observes,’ Einstein asked them, ‘does that change the state of the universe?’ [3, pp. 332–333 and 515, respectively] [Author’s comment on the latter quote: Perhaps it does! Perhaps the multiplicity of observations is what makes such a complex system. Perhaps that is why no one is able to have the full view of an infinite reality . . . it is always in flux from a variety of sources.]

* The authors define a complex system as an open *system* with continually cooperating and competing elements. *Features:* This type system continually evolves and changes its behavior (often in unexpected ways) according to its own condition and its external environment. Changes between states of order and chaotic flux are possible. Relationships among its elements are imperfectly known and are difficult to describe, understand, predict, manage, control, design, and/or change. *Notes:* Here “open” means free, unobstructed by artificial means, and with unlimited participation by autonomous agents and interactions with the system’s environment. A complex system is not necessarily an enterprise. An enterprise is defined as a *complex system* in a shared human endeavor that can exhibit relatively stable equilibria or behaviors (homeostasis) among many interdependent component *systems*. *Features:* An enterprise may be embedded in a more inclusive complex system. External dependencies may impose environmental, political, legal, operational, economic, legacy, technical, and other constraints. *Notes:* An enterprise usually includes an agreed-to or defined scope/mision and/or set of goals/objectives. [1]

+ “The traditional view of management is that the manager sees something that has not gone according to plan and tries to fix it by ‘troubleshooting,’ trying to find the piece that ‘broke’ and fix it. ‘The study of complex adaptive systems suggests that we might be better off maintaining the anxiety, sustaining the diversity, letting the thing simmer for a while longer to see what will happen on its own’ [8, p. 3]. For the manager, this creates a dilemma, even in the self-organizing environment. If one does not place sufficient constraints and ‘control’ workers, they may reach a point where the organization goes unstable. Yet too much control limits adaptability. Those running an organization, if they want maximum learning and growth, have a very fine line to tread to maintain this.” [9]

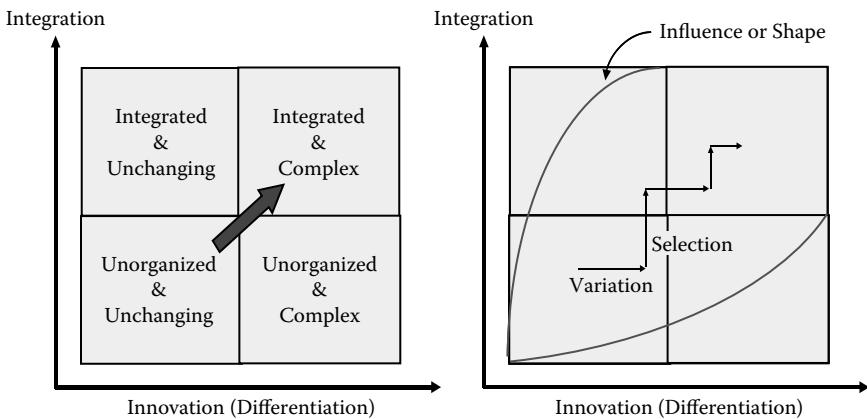


Figure 3.1 Ever increasing complexity through iterative combinations of innovation and integration using continual variation and selection techniques [11] (after Gharajedaghi, 1999; [12]).

Complexity requires that leaders, in particular, look at the desired big picture, and “tweak” selected variables they can control to try to steer their organization in that general direction. Sometimes, it is more important to postpone making decisions instead of trying to appear to be “in control,” e.g., behaving as an “alpha male” [13]. Could some erstwhile leaders be reacting so quickly out of fear (a root cause of “rigidity of thought”* that hampers them from embracing new ideas)? Do they need to “stop” the problem in its tracks because of uncertainty in what may happen? A research question to consider: Is this tied to the prevalent emphasis on risk management instead of the pursuit of opportunity that seems so much more appropriate in complex system environments [14]?

One needs to consider an alternative mindset: short-, mid-, and long-term plans must continually change to better adapt to uncontrollable and unpredictable environments. Rather than making early decisions, to appear assertive and in control, for instance, a more informed policy would argue for encouraging variety to accelerate processes of natural evolution and waiting until it becomes clearer what to shape or select. A good research question: What are some 20% to 80% rule heuristics for telling a leader when one needs to stop learning and intervene by making a decision? Perhaps this is where the recent neuroanatomy/evolutionary biology research about how these “instincts” evolve will be helpful. As mentioned earlier, behavior patterns learned over time can become “instinctive” due to the neural pathways in the brain that are created. One no longer needs to think about what should be done, one just reacts. Can this research be used in conjunction with research

* This is a phrase from psychology labeling a tendency to stick to one’s current mindset, outlook, or belief system, leading to inflexibility in thinking that can hinder changes or adaptability in behavior or responses to stimuli.

about “mirror neurons” [15,16] to develop new training programs that get people to “break” rigid constructs of reality and their “instinctive” reactions that may no longer be appropriate? Standard counseling psychology techniques, incorporating many from group therapy, such as are implied in coaching alpha males [17] can be used as well.

Running an organization has a strong “art” component, because leaders need to understand themselves (first) as well as other people and their individual psychologies. Interpersonal dynamics do not always just involve the group members; sometimes it is the behavior of the leader that really matters. These understandings are critical to their success as leaders in shaping how their organization should evolve.*,+^{*}

The authors now discuss, a little more precisely, what they mean by view.

3.2.1 *Definition of view*

A specific instance of *view* is defined as any combination of *scope*, *granularity*, *mindset*, and *timeframe*. Each of these latter terms is defined as follows.

Scope: What is included in an individual’s conceptualization.

Notes: Conceptualization is akin to perception (e.g., visualization). An individual’s conceptualization is affected by a host of variables, both physical and nonphysical. For example, everyone experiences the same color in different, albeit mostly similar, ways, as a function of how a given subband of wavelengths affects the optical system of the eye and brain, and what psychological factors of the mind may be associated with certain colors. Therefore, it should be readily easy to accept that no two people “see” reality in the same way. Specific analogies of scope are the field of view (FoV) of a camera, or more appropriately here, the “mind’s eye.” When one sets

* “If there is too much change and freedom, then their system can tip over into chaos—witness what happens in a revolution, for example. Too little innovation, and systems become rigid—totally predictable but able to respond only through tried and established methods. Governing an organization is therefore an art, and there needs to be constant monitoring of the system to check which way it is heading. If it is becoming too stable, then change and a degree of freedom, perhaps through decentralization, need to be introduced to push the system back to complexity. Conversely, if there is too much change and the system is threatening to melt down, restraints and disciplines must be quickly reinforced.” [18, p. 16;9]

+ “...view your organization as a complex system rather than using the machine metaphor; provide minimum specifications, and don’t script plans in minute detail; don’t always ‘go with the data’ when ‘going with your gut’ feelings may be more appropriate; take your organization to the ‘edge of chaos’ by fostering a level of anxiety, diversity, discomfort, and contentiousness to promote creativity; take advantage of paradox and tension rather than fighting them; let solutions to knotty problems emerge; promote informal communication networks in the organization rather than banishing them; and use the ‘tit-for-tat’ strategy of competition-cooperation to forge positive, symbiotic relationships as a first strategy and abandon that as a strategy only when your ‘partner’ does not reciprocate.” [9]

or determines scope, by definition, this means that everything else, *not* in scope, is “abstracted out,” e.g., not “seen” by that individual, at least in that view, because those things are not relevant to the person’s intended present state of being, e.g., purpose.

Granularity: The ability of a person to discern and discriminate individual items of a conceptualization.

Notes: Granularity is akin to a capability to observe details, e.g., it is like resolution. Subsets of detailed items will likely include arrangements or patterns, some of which may not be discernable in other views.

Mindset: What currently captures an individual’s attention in a conceptualization.

Note: Mindset is akin to one’s cognitive focus that may observe or contemplate, e.g., within his/her scope and with the associated granularity, a single object, pattern, notion, or idea, or collection of such elements. Rigidity of thought and its inherent influences undoubtedly comes into play here.

Everyone’s view of reality is bounded, often rigid. Many factors influence mindset, e.g., the way the brain is “hardwired,” brain chemistry, how one was raised, relationships with siblings, culture, life experiences, and physical embodiment. Like how one sees a color, life events are seen differently, e.g., witnesses’ varying descriptions of criminal events—often not seeing what in actuality was there when they did not expect it to be there.

One’s mindset is also influenced by beliefs. Fundamentalism is a “safe haven” in a world of uncertainty and confusion.* One does not have to think or understand; one just has to believe. This allows people to avoid dealing with uncertainty when faced with the fact that they cannot “know” the full complexity that surrounds them. There is a strong analogy here (again

* “Not everyone, however, sees fundamentalism as inherently damaging. Some scholars believe that, by offering psychological security and social identity to people otherwise adrift, it offers the best hope for a stable future. ‘A case can be made that someone with a strong, confident religious identity is better qualified to survive in a globalising world of shifting and collapsing identities,’ says historian Philip Jenkins of Pennsylvania State University.... Some scholars even argue that fundamentalism is religion’s last hurrah. Jenkins says the history of movements such as Calvinism suggests that fundamentalist movements eventually become secular. Fundamentalism, he suggests, may be a ‘necessary way station on the road to enlightenment.’” [19] Indeed, often, a complex system will swing to the extreme (edge of chaos?) as it gradually defines or shapes the median, or middle, path, somewhat akin to statistical regression to a mean. Joe DeRosa gave an example of this at a 12 September 2007 New England Chapter meeting of the International Council on Systems Engineering (INCOSE). In a “minority game” a group of individuals in a room is told up-front that each person is to choose, independently, without collusion with others, one of two walls, and to walk over there. The group then observes the distribution of people along both walls. This process is to be repeated a number of times. After the last iteration those by the wall with the fewest people will win a prize. Once everyone understands the rules of this game and is motivated to win a prize, the game begins. Remarkably (and in other games similar to this), there is a rather uneven distribution during the first several iterations but the end result yields roughly the same number of individuals again each wall.

involving rigidity of thought) that informs us about why “heroes” in small working groups may be ostracized. They may perceive solutions or problems in ways not often seen or understood by the rest of the group and threaten the common view of the way things should proceed in the workplace. They see a more comprehensive slice of reality than those around them can see. Again, this points to the need to embrace uncertainty, understanding and accepting that an infinite number of views of reality make up the complex system that surrounds us. Each “slice” or view of reality is a partial “truth” making up the whole. There is no “wrong” view—it all is part of the gestalt, all of it partially correct. Many of the ideas espoused in [17] about coaching alpha male executives apply here, as well.

The aforementioned recent research on mirror neurons* is exciting because it gives some physiological basis and understanding for what may underlie a person’s mindset as well as some destructive group dynamics. Mirror neurons are a physical clue to embracing and understanding another’s perspective—putting oneself in someone else’s shoes, as it were.

Timeline: The time interval of an individual’s conceptualization.

Note: Timeframe is akin to temporal component of one’s conceptualization, e.g., the timescale over which it occurs.

These four dimensions of view are illustrated in [Figure 3.2](#). Scope goes from small to large, and granularity goes from coarse to fine. The mindset axis is more general, in that it cannot be characterized by such a qualitative descriptor, as indicated by the “...’s at each end of the double-sided arrows. Timeframe is envisioned as the three dimensions {scope, granularity, mindset} moving within the fourth dimension of time. Each of these four axes can represent an infinite (unbounded) number of possibilities from which an individual might select in forming a conceptualization.

Figure 3.2 attempts to represent, abstractly, one’s potential views of a complete complex system; one’s “slices” or views of reality fall in the gray area. No one is able to see *all* (an infinite number) of the variables making up a complex system. Our brains are limited, and complexity is unlimited. Diversity of views is helpful, despite the conflict it also causes, because it pushes

* “The discovery of ‘mirror neurons’ in the 1990s... Mirror neurons were activated not only in response to self-generated behaviour (reaching for an object, say) but also in response to actions performed by other individuals. Pain and emotional behaviour were similarly mirrored.... We were a composite of two phantoms.... The so-called ‘minimal’ or ‘core’ self—was,... a transient entity, recreated for each and every object with which the brain interacts.’... The other phantom was the ‘extended’ self: a unified, continuous being journeying from a remembered past to an anticipated future, with a repertoire of skills, stores of knowledge and dispositions to act in certain ways. This ‘autobiographical’ self emerged from language and long-term memory networks.... We are all just a stumble or burst blood vessel away from being someone else. Selfhood is malleable.... The neurological diseases... tended to [show that] one occasionally saw what appeared to be clear dissociations of the two ‘selves.’... [Also,] Now it is not so clear where one person ends and another begins.”[4]

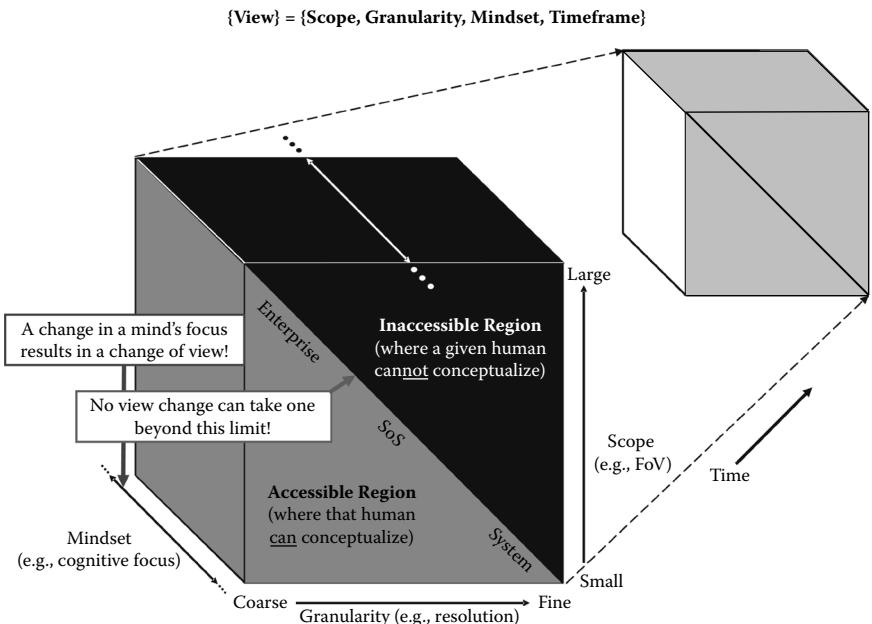


Figure 3.2 Conceptual definition of human perceptions of view.

the boundaries of rigidity of thought. It helps to reveal more aspects of the complex system, more “slices” of the whole complex “pie.”

In terms of quantum mechanics one sees a parallel to complex systems. Similar to the Heisenberg uncertainty principle, there is a limit to how much of a complex system an individual is able to “see” at one time due to our limited views of reality. One cannot see it all, just as a physicist cannot measure both the time and place of a quantum particle accurately. Each individual’s view of reality is different. Not only that, but some are able to see or comprehend more than others. The more perspectives or views of reality one is able to embrace, the greater the understanding of the complex system one holds. The ability to embrace multiple perspectives or views of reality can be taught. Although still subject to careful interpretation and even somewhat controversial, research is showing that mammal brains can change and expand neural pathways. Eventually, this may lead to a greater understanding of different (and hopefully, improved) ways of thinking and behaving. [20–22]

This is consistent with Kuras [6] saying that *no one* has a true perception of the underlying reality of any situation.* Kuras said the brain has a

* Considering the footnotes of the Introduction, Einstein might have asserted something similar.

finite number of neurons,* and therefore, is ultimately of limited capacity to observe everything accurately. For example, imagine actually trying to locate the proverbial needle in a haystack, or to list *all* outcomes of an organization's current endeavors. This implies, for example, that everyone has limits to the *ranges* of scope, granularity, and timeframe that can be perceived. This region of space within which one (a given person) cannot conceptualize is depicted by the inaccessible *black* region of Figure 3.2. The *gray* region is the space in which that same person can perceive or visualize. (In this abstract space, the *black* and *gray* regions of any two individuals are deemed to be different.) Along the mindset axis (see Section 3.3.2, Mindset range: finite or infinite?) the *range* of these regions is depicted as infinite.

Furthermore, each person has a different, or at least distinct, perception of that reality. As Ryan [25] said (see Section 3.3.1, Definitions of emergence), "When practical limitations are the cause, [a] property [of a macrostate] may appear to be emergent to one observer, but is not emergent to an observer with a deeper understanding of the microstate." Sheard [26] used the analogy of the Grand Canyon, where the subcanyons correspond to distinct realities of people. Sheard suggested that achieving a better understanding of something with just one's own viewpoint is limited, and the existence of other competing points of view is necessary (in the present authors' terms) for accelerating the evolution of CSE. This is a fundamental point (see the Introduction) in being able to see and handle the confounding properties of complex adaptive systems.

3.2.2 Mindset range: finite or infinite?

Consider whether a person's mindset has a finite or infinite *range*. The authors believe that the range of the mindset axis is infinite because of analog properties of the brain's physiology and because of the emergent properties of the *mind* (see Section 3.3.6, Emergence and surprise). Various references in the literature [27–29] are quite convincing in this regard. For example:

C. W. Johnson [30] "...it makes little sense to talk of human cognition in terms of individual neurons. Consciousness is intrinsically a systems level property quite distinct from the underlying physiology of lower level components."

* It is better to say one has a finite number of neurons at any given time. Current research [23,24] suggests one is able to generate vital *new* neurons, and that old neurons atrophy. In addition, the brain's "circuits" can adapt. [20, p.112] But, even so, we still have a finite ability to view the whole of any complex system.

Ryan [25] states: "Even though formal and social systems must both ultimately have physical instantiations, they do not have obvious bounds...on possibilities. For instance, although the number of distinct thoughts a human mind will have in its lifetime is finite, we apparently cannot specify in advance any finite set containing every possible thought, nor determine the finest possible distinction between two thoughts the mind is capable of making."

So, although any individual can have only a finite number of distinct thoughts in a lifetime, the *set*, and by implication the *range*, from which those thoughts can be selected is infinite!

The authors feel that the mind's potential is unlimited. The mind is the emergent phenomenon of the brain. Consciousness is a prime example; the individual neurons and parts of the brain by themselves cannot account for consciousness, and the actions of consciousness are much more than the biology of the parts of the brain. The very definition of emergence is one that implies infinity, and just as the universe is infinite* on a macro scale, the mind is infinite on the micro scale, or perhaps, on a different plane in reality.

Clearly, this is a philosophical question that goes beyond an analytical approach considering only the number of neurons (or synapses) involved; counting the number of brain parts and their interconnections in upper-bounding the number of conceptualizations possible is too simplistic. Much larger questions have been raised in terms of what is computable in the universe [31,32]. People seem to view the world with an infinite range of realities. Many are sure, adamantly, of their own view's "truth." But the true elusive reality encompasses all of the "subrealities" making up the complete complex system. Accepting and embracing this notion requires one to embrace uncertainty, that no one person can truly know the whole picture. But uncertainty is one of the most difficult states for most individuals to embrace. It means one has a lack of complete control in their life (where the balance of control and no control is itself uncertain), and one cannot predict with any certainty what course life will take.

* For all practical purposes the authors (and perhaps most other people) cannot conceptualize something that does not end. As human beings we need to think in terms of finiteness. Because of this, hearing that the universe may be finite, "curving" in upon itself as suggested by Einstein's general theory of relativity, one naturally asks, well what's outside it then?! If nothing is outside, not even a pure vacuum, i.e., if the (finite) universe is all there is, then mustn't the universe be infinite?! Then there is the "side information," adding to the confusion, that the universe presently is expanding, and at an accelerating rate. Einstein used the analogy of an unending traversal of the surface of the earth's globe, a finite object. Returning to the same spot would occur at a different (future) time, so in this sense, one can traverse "forever" without end. All this is truly mind boggling.

3.2.3 Multiview analysis

Kuras has argued that scale (the authors' view) is tied to mindset mainly (i.e., $\{\text{view}\} \approx \{\text{mindset}\}$), although Kuras admits that scope and/or granularity often change with one's change in mindset [33]. In the authors' opinion, this interpretation with its emphasis on mindset is too restrictive. However, as indicated by the note in *blue* font in Figure 3.2, a change in mindset does lead to a change of view. See Figure 3.3 for two rather famous examples of this [34]. Again, according to the authors' definition, one's view can change if any of the four components (scope, granularity, mindset, or timeframe), or any combination of these components changes.

Referring to Figure 3.2, the note in *red* font indicates that no view change can take one from the *gray* region into the *black* region. For example, suppose one is working on an SoS at some combination of scope, granularity, mindset, and timeframe within the *gray* region and in the vicinity of the *red* arrowhead. Assuming for now that the mindset and timeframe remains the same, one cannot simultaneously increase both the granularity and the scope; one must be content with increasing either component of view and decreasing the other component accordingly to move along the *gray* side of the diagonal boundary between the two regions. Increasing scope and decreasing granularity by some amounts can be akin to viewing a larger enterprise of which the SoS may be a part. Conversely, increasing granularity and decreasing scope by similar amounts corresponds to viewing a particular system that may help compose the SoS. By analyzing both these new views, as well as the original view, one should be able to learn more about the underlying reality concerning the work, thereby enhancing one's understanding of what to do. By taking advantage of changes to mindset and timeframe, as well, there is an increased richness to be had in solving particular problems of pressing importance. It should be better to consider different perspectives rather than continually trying to "beat a problem into submission" using only one

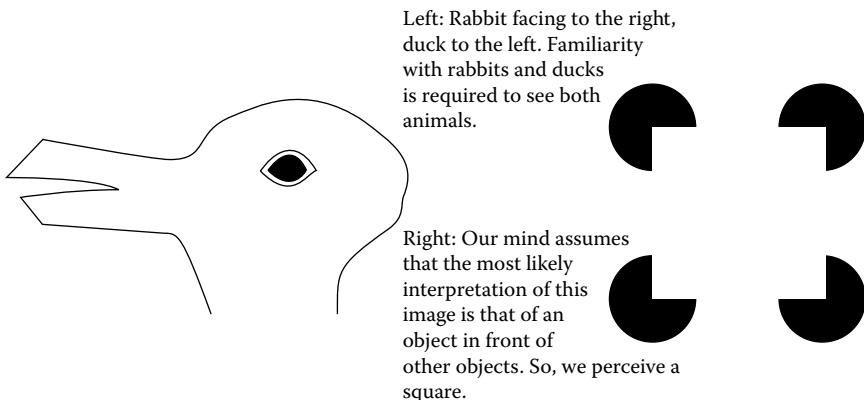


Figure 3.3 Changes in mindset result in two distinct views.

particular view. Using only one point of view and making the problem “fit” that view is limiting and inherently self-destructive in problem solving. One must embrace multiple perspectives on a problem in order to perceive possible working solutions to it. Similar to a wall, some limited views of reality can block our ability to see the operating dynamics underlying a particular problem.

Thus, the usual vertical view of enterprises, SoS, and systems can be tilted to one side as in [Figure 3.2](#) and interpreted in the context of conceptual multiview analysis of a single complex system. Typically fine-grained granularity and narrow scope will be more appropriate for individual systems, particularly automata, while coarse-grained granularity and broad scope will be more appropriate for aggregations of autonomous agents (e.g., individual human beings) mixed with large numbers of autonomic units in social systems such as enterprises.

Human intuition processes information at higher brain levels than one is consciously aware, laterally “connecting the dots” of minute information subconsciously and consciously gleaned through observations. [35,36] Some people do not react in predictable and rational ways as perceived by others’ expectations. (Someone with a similar view may find them eminently predictable and rational.) For them, using rational (only) methods (from these others’ viewpoints) will not work as intended, so how do leaders deal with them in the workplace? “...

1. Increase the flow of information in the system.
2. Use fundamental questioning to keep organizational members thinking about solutions to organizational problems.
3. Keep the size of work teams to 12 or less, and organizational units to 150 or less, to be consistent with the number of relationships people can handle.
4. See the manager as a participant and part of the work system, rather than as an outsider.
5. Promote redundancy, and fight against the traditional practice of guarding information and skills to ensure longevity or security.
6. Don’t be afraid of letting unstable situations simmer until a solution emerges rather than forcing a solution.
7. Consider doing large group interventions where the people put in a room to solve a problem are not just organizational leaders, but a cross-section of organizational members.” [9]

Here are the authors’ selected comments: 1—As alluded to above, what is the critical point that causes too much friction and a collapse of the system? 3—What is the critical number of relationships people can handle? 5—How does a leader get people to trust and to share information? How does one get them to step outside their basic human nature? It takes time to build trust in a traditional way. Is the time available in a complex structure where things happen very rapidly? How can a leader “shortcut” the time needed to develop

a relationship and build trust, as well as group cohesion? Group cohesion and trust among members often can be built through “in-sync” physical activities, through enduring hardships in the group, as well as through problem solving as a team. But these methods take time to accomplish. 6—Again, move away from the macho, decision-making hierarchical culture of the alpha male. 7—This may cut down on “group think,” but what about intimidation and fear of those who are not organizational leaders? How does a leader minimize subtle and overt intimidation tactics?

Now that some of the sociocognitive aspects of human perception, individual viewpoints, multiple perspectives, and multiview analysis of complex systems have been covered to some degree, the authors are ready to discuss the concept of emergence in an SoS.

3.3 *Emergence*

Emergence in an SoS is a result of what happens within and among the systems comprising the SoS, including human actions and relationships. The focus here is not on this objective (ontological) aspect of emergence but rather on the observed (epistemic) aspect involving human cognition, perception, and conceptualization, often involving multiple viewpoints. Intuitively, the authors feel that emergence is a word that suggests the unexpected, something that one is not looking for but that suddenly appears. However, others feel differently. Thus, it is useful to first explore various definitions of emergence. Then some examples are given, and several properties of emergence are examined. Finally, the authors more specifically address the role of emergence in an SoS.

It is assumed that most SoSs of current interest imply the existence of programmatic organizations that are attempting to assemble, manage, or at least oversee those SoSs, where the component systems of each SoS are, in turn, managed by a distributed set of essentially independent organizations. In such cases sociotechnical factors concerning organizations, decentralization, and trust are quite relevant. Therefore, the authors also intersperse relevant thoughts regarding these topics.

3.3.1 *Definitions of emergence*

[30] and [37] both make a basic point: There is no concise, precise, and generally accepted definition of emergence. True, but this can stimulate interest and productive dialog.

The authors prefer the following definition.

Emergence: Something *unexpected* in the collective behavior of an entity within its environment, not attributable to any subset of its parts, that is present (and observed) in a given view and not present (or observed) in any other view.

Notes: Some people employ a broader definition of emergence where things that emerge can be expected as well as unexpected [38]. The authors prefer to

consider expected things to be intentional, designed in, known in advance, explainable with hindsight or deconstructed analyses, or at least not very surprising (e.g., although human evolution or cognition may not be easily explained, these general states of being are no longer surprising), and not warranting special recognition of having an emergent property. The authors' emphasis on the unexpected in the above definition is intended primarily to motivate the development of very adaptable and robust management processes in CSE which can deal more effectively with surprises that inevitably occur in complex systems, in general, and SoSs in particular.*

Groisman's paper on complexity and organizational change [9] provides many insights about complex systems and how the science of complexity can inform the topic of organizational dynamics.[†] This seems especially apropos when applied to an Organization managing an SoS. [The authors are designating such an Organization with a capital "oh" to distinguish it from organizations managing the individual systems of the SoS.] More empirical data certainly is necessary to help bolster the complexity theory concepts in systems engineering environments.[‡] This is a "chicken-and-egg" problem: In situations typical of the existing military acquisition environment, for example, organizations may be reticent to apply complex systems engineering techniques until they have been "validated." However, as can be appreciated, it is more difficult to apply the scientific method strictly to investigations

* *The Black Swan* [39] is highly recommended. The author of this book, Nassim Taleb, treats the importance of being aware of the possibility of highly unlikely but high-impact events, and advocates sound logic in considering complete sets in the event domain when trying to attribute causes.

† "In an environment that seems to be changing, organizations want to be more adaptable and better able to learn from experience in order to reconfigure themselves in the face of new demands [40, p. 373]. Cohen goes on to give other reasons as well: the acceleration of information technology revolution and its taxing of our ability to process information and data, and the degree to which organizations are being created, dismembered, and dismantled. The boundaries of organizations have been permeated through the use of virtual organizations, consultants, outsourcing, temporary hires, and ad hoc teams. Some of these developments can be described easily using the concepts of complexity theory, which, along with chaos theory, are useful in describing the rapid transformations undergone by nonlinear dynamic systems such as organizations." [9]

‡ "Complexity theory is a new way of looking at how complex structures form, adapt, and change....it remains to be seen whether it can be successfully applied to organizations....there is a paucity of empirical data confirming that organizations designed on this new model are more effective and efficient. There is a view that...self-design is more in line with emerging philosophical and ethical views about the workplace. In a normative sense, a complexity theory view can be considered more humanitarian and ethical. Writers in the field, in addition, suggest that organizational designs based on this new paradigm are likely to be more efficient and effective in turbulent environments. As more and more large organizations change their organizational design model to follow the prescriptions of complexity theorists, there will be more opportunities to judge whether these new designs work. This suggests a research agenda that uses the scientific method to determine whether the principles advocated by complexity theorists in the organizational arena can truly improve organizational performance." [9]

of human behavior—there are inherent limitations in relying solely on this approach. Again much research and experimentation remains to investigate how organizations can benefit from complexity theory. On the other hand, rather than wait for validation, those in authority should encourage the development of more agile management techniques. Then they can try out with greater confidence some complex systems engineering principles experimentally within their domain, with a better chance of not only pursuing promising opportunities but also limiting the damage if unexpected negative results begin to appear.

A rather comprehensive treatment of various definitions of emergence is found in Fromm [41]. S. Johnson [28, pp. 18, 21] discusses “the movement from low-level rules to higher-level sophistication...” and “...self-organization, of disparate agents that unwittingly create a higher-level order.” [37] says “...emergence has to do with qualitatively different kinds of description being appropriate to different levels of abstraction....”

Bar-Yam [42] states “Emergence is ...

1. ... what parts of a system do together that they would not do by themselves: collective behavior.
2. ... what a system does by virtue of its relationship to its environment that it would not do by itself: e.g. its function.”

Bar-Yam also gives good explanations of how—in terms of the two examples [see [Section 3.3.2](#), Examples of emergence]: the trees and their forest; and a key and its lock—these two pieces of his definition relate to each other.

Ryan [25]: “Definition 1 (Emergent property). A property is emergent iff [if and only if] it is present in a macrostate and it is not present in the microstate.” “Definition 4 (Novel Emergent Property). A property is a novel emergent property iff it is present in a macrostate but it is not present in any microstate, where the microstates differ from the macrostate only in scope.” “Definition 5 (Emergence). Emergence is the process whereby the assembly, breakdown or restructuring of a system results in one or more novel emergent properties....”

It seems that Ryan’s revelation that emergence only has to do with scope [with granularity held fixed] comes from these definitions! Why can the reverse not also be true?! I.e., why can one not see something “emerge” when observing a view with more granularity, even though the scope has not changed?!

3.3.2 Examples of emergence

As evidenced just above, the authors think that unexpected emergence should include what might happen to an observer experiencing an increase in granularity, perhaps accompanied by a decrease in scope. A hypothetical but likely actual example of this corresponds to a biological researcher's observation of previously unknown patterns in a microscope view of a cell. One sees a "pattern" not otherwise observed because of an increase in granularity (resolution), not an increase in scope. Nevertheless, the pattern that emerges can still be thought of as a macrostate effect from a microstate cause. As opposed to situations where the granularity becomes fine enough, the authors suspect that behavior that only starts to occur when the scope becomes large enough is more germane to the issues SoS engineers face. Again, the underlying behavior (that would lead to emergence) is ongoing whether it is seen or not.

Other examples, this time where the scope does not change, also can illustrate emergence with an increase in granularity. Suppose a vu-graph projector is out of focus enough so that at first the projected image on the screen or wall suggests to the observer just a blob of blurry colors or shadings. Then suppose the projectionist improves the focus so that an observer sees what s/he thinks is a mosaic art form. The observer could term this mosaic emergent but perhaps not surprising. Finally, when the focus is sharpened further, the observer sees that the mosaic is actually comprised of a collage of detailed but unrelated photographs of objects/subjects/scenes; this emergence could be deemed surprising but explainable. One might argue that neither of these steps constitute true emergence, because subsets of the image could reveal the same quality.* However, in another instance in this vein, the blurry image or mosaic might turn into a well-known, 8×8 pattern of squares, and voilà, a chessboard emerges. In this case, no subset of these 64 squares constitutes an official chessboard, so the definition of emergence is satisfied in the sense that the emergent property cannot be ascribed to any subset of the whole.

* In art one finds examples of this in both Pointillism and Impressionism. As one maintains a certain distance from a painting, for example, the images on the canvas appear realistic. But while still taking in the whole painting, as one moves closer to the work, the individual marks appear, and the painting takes on an abstract quality—the realism seems to diminish greatly. The scope has not changed, unless one changes the area of cognitive focus (recall the definition of scope and mindset) away from the whole painting. On the other hand, one might say that the definition of emergence is not satisfied because one can see this abstract quality from only a subset of the painting. This elicits a related thought. With averted vision one may be able to see movement, phase transition, or patterns of emergence within the scope of the contained system as granularity is increased [43]. Research in modeling has shown that one can see patterns in bird flocks by just noticing, in a vision-averted sense, three or four individual birds scattered throughout the flock. A similar phenomenon is at work in picking out four-leaf clovers in a bed of normal clovers.

Ryan has two good examples of emergence that are neither surprising nor unexpected, a Möbius strip [25] and;

The orchestra produces an emergent property, call it a symphony. It is emergent because none of the components . . . could produce it in isolation. [44]

Norman [44] commented:

[The Möbius strip] exists outside of the conceptualization of any human. . . . So, there exists a phenomenon which shows itself only upon closure of a specific relationship [twisting and attaching the ends of the rectangular strip of paper]. What should we call this? In the case of the Möbius strip it [is] the “emergence” of one-sidedness.

3.3.3 *Beneficial emergence*

Emergence can have benefits, consequences, or irrelevant or as yet undetermined effects.

Mogul [45]: “Emergent behavior can be beneficial. . . . But it is not always beneficial. . . . I will use the term ‘emergent misbehavior’ to focus on problematic behavior. . . .”

The authors are more interested in pursuing opportunities associated with beneficial emergent behavior, although one certainly needs good management heuristics to help determine when emergent behavior is “bad,” as indicated previously. Also of interest is how to recognize correctly that not all group dynamics traditionally viewed as “bad” (e.g., of the “conflict” variety) really are so. Such behavior is bad only when people become mean spirited and aggressive. Conflict has tremendous creative potential when its power is harnessed and allows the passion of ideas to be heard without it becoming personal and vindictive.

Boardman and Sauer [46] include “foreseen or deliberately designed in” emergence for an SoS. However, they acknowledge that some emergent behavior (especially the undesired kind) can be unexpected. Again the authors acknowledge that some (nonemergent) behavior can be “designed in,” but prefer to reserve emergent behavior for the *unexpected*. It seems the most one can do is design in “wiggle room” to allow the possibility of adapting future directions to accommodate (what is likely the inevitable) unexpected emergence.

C. W. Johnson [30]: “. . . there are many systems level properties that are not directly related to system sub-components but which might be predicted. . . . For

example, ‘risky shift’ occurs when greater risks are accepted by groups than would have been taken by individual team members. . . . Designers can, therefore, take steps to guard against these behaviors that might otherwise compromise [CSE].”

This is interesting in its own right regarding group dynamics [10].

3.3.4 *Emergence and prediction*

Mogul [45]: “Emergent behavior is that which cannot be predicted through analysis at any level simpler than that of the system as a whole. Explanations of emergence, like simplifications of complexity, are inherently illusory and can only be achieved by sleight of hand. This does not mean that emergence is not real. Emergent behavior, by definition, is what’s left after everything else has been explained.”

The last part of the definition resonates with the emphasis the present authors are placing on unexpected and unexplainable emergence. Note that emergence often is explainable after it is recognized, studied, and analyzed. To prepare better for the possibilities of high-impact (albeit perhaps very unlikely) future unknown unknowns (i.e., Black Swans [39]) one should endeavor to exert more effort and resources into explaining those heretofore unexplained emergent events.

Abbott [47]: “. . . we have no idea how to build simulations that can identify emergent phenomena—or even more difficult, how to identify the possibility of emergent phenomena. . . . I called this the difficulty of looking upward.”

This is an important point to make about emergence; the school of thought the authors are subscribing to has said all along that particular outcomes that emerge (i.e., specific emergent properties) are not predictable.

S. Johnson [28]: “But it is both the promise and the peril . . . that the higher-level behavior is almost impossible to predict in advance. . . .” “. . . understanding emergence has always been about giving up control, letting the system govern itself as much as possible, letting it learn from the footprints. . . .”

One can predict there will be emergence if the system has characteristics of a complex system but one cannot prespecify what behaviors will emerge. Although outcomes may be surprising, the fact that they occurred is *not* (see Section 3.3.6, Emergence and surprise).

C. W. Johnson [30]: "...complexity [is] one of the most significant 'macroethical' questions... 'the key point is that we are increasingly building engineered systems that, because of their inherent complexity, have the potential for behaviors that are impossible to predict in advance.' A recurring theme... will be the 'surprise' that engineers often express following adverse events....incidents revealed traces of interaction between operators and their systems that arguably could not have been anticipated using current engineering techniques."

3.3.5 Emergence and entropy

Weeks et al. [48] illustrate the concept of increasing the mutual information between the system specification, S , and the implementation ("language"), L , as development proceeds. In itself, this makes sense to the present authors. However, they define the amount of emergence as the mutual information, or correlation, between S and L , i.e., the entropy $H(S)$ minus the conditional entropy, $H(S|L)$. The present authors do not use this definition of emergence. Further, Weeks et al. characterize $H(S|L)$ as "surprise"; agreed—that interpretation, i.e., the (unexpected) uncertainty that remains after implementing the system, seems to be of greatest importance. They also say "...a system exhibits minimal emergence when everything is a surprise (zero mutual information)." The present authors feel, to the contrary, when everything is a surprise, there is *maximal*, not minimal, emergence! It is useful to equate "uncertainty" with entropy here. Further, characterizing $H(S|L)$ as surprise (see [Figure 3.4](#)) supports the present authors' kind of emergence.

3.3.6 Emergence and surprise

Damper [37]: "Many authors have attempted to explain emergence in terms of surprise... The problem here is that surprise is in the eye of the beholder." "John Holland... pointed out that von Neumann's demonstration of a self-reproducing machine 'nicely refutes the use of surprise as part of the definition of emergence.' 'It's still a great example of emergence,'... And rather than feeling 'cheated' once we know the details, there should be '...no diminution in wonder....'"

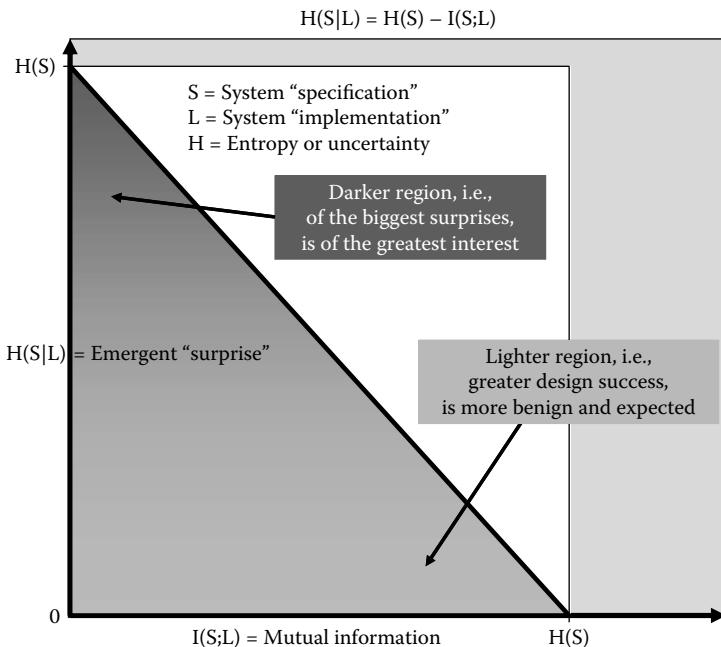


Figure 3.4 Emergent “surprise” as the conditional entropy of an implemented system.

Turning surprise into wonder is not a bad thing! Surprise is what is interesting, especially that different people will differ on the degree to which they are surprised. In the authors’ definition emergence can relate to “non-biological” machines, but is mainly intended to apply to those systems demonstrating biological or adaptive behaviors. Most traditional machines do not demonstrate emergent properties; while a certain result may be a “surprise” to the observer, the sequence of events leading up to the unexpected occurrence can be deconstructed and reproduced. With “biological” or adaptive systems, emergent behavior cannot be predicted, fully deconstructed, or necessarily reproduced.

Ronald et al. [49]: “The description of a phenomenon as emergent is contingent, then, on the existence of an observer; being a visualization constructed in the mind of the observer, emergence can be described as a concept, like beauty or intelligence. Such concepts are slippery.” “Clearly, the existence of an observer is a sine qua non for the issue of emergence to arise at all.” “Our emergence test [see Ronald’s paper] centers on an observer’s avowed incapacity (amazement) to reconcile his perception of an experiment in terms of

a global world view with his awareness of the atomic nature of the elementary interactions.”

The authors agree! These concepts surrounding the inclusion of an observer are fundamental in highlighting the surprise aspect of emergence on which the present authors want to focus. So the degree of surprise is important: As the observer tries to learn more about the emergence phenomena, his/her surprise may diminish. The more important emergent properties, in the authors' opinion, would be those for which surprise persists despite the best efforts of the observer.* Again, these latter properties are more important because, with further study, they might provide clues for Black Swan conditions [39].

3.3.7 *Emergence in system of systems*

Again, consider the problem facing an Organization attempting to manage an SoS. Some things can be learned about how such Organizations can become more successful, or at least remain viable, by examining factors that may have led to the collapse of such complex systems as human civilizations. Perhaps there are several parallels to organizations, cf., the (present authors') italicized passages in the footnote below.[†] Thus, some centralized management functions appear to be necessary for continued health of

* An example related to the Möbius strip may be illuminating. One can try this at home. Construct a Möbius strip with tape and an elongated rectangular-shaped sheet of paper. Then cut the strip down the middle lengthwise. The result is another Möbius strip but one that is twice as long and half as wide. Then, cut this new Möbius strip again, lengthwise. Now the result is two disjoint but interlocking Möbius strips, each of the same length as what one started with at the second iteration. How can this be, considering one started with a Möbius strip in the second case, as in the first, but got a fundamentally different result in the second case compared to the first?! Perhaps those more versed in topology have a mathematical or logical explanation but the present authors do not see it, and remain surprised even after observing the event! Similar things happen when starting with a paper with two twists instead of one. After the first cutting one obtains a strip, with two twists that is twice as long and half as wide. After the second cutting one has two disjoint interlocking strips of the same length with two twists each. Again, this is mind-blowing!

† “[Societal] Collapse is manifest in such things as: a lower degree of stratification and social differentiation; less economic and occupational specialization, of individuals, groups, and territories; *less centralized control*; that is, less regulation and integration of diverse economic and political groups by elites; less behavioral control and regimentation; less investment in the epiphenomena of complexity, those elements that define the concept of ‘civilization’: monumental architecture, artistic and literary achievements, and the like; *less flow of information between individuals*, between political and economic groups, and between a center and its periphery; *less sharing, trading, and redistribution of resources*; *less overall coordination and organization of individuals and groups*; a smaller territory integrated within a single political unit. Not all collapsing societies, to be sure, will be equally characterized by each item on this list, and the list is by no means complete.” [50, p. 4]

the Organization. The next quote* also suggests a traditional hierarchical structure to implement intentions of the central controlling body. In the last two quotes, note the implied desirability of more complexity if one deems a vibrant civilization as being "good." Thus, it would seem that those striving for more advanced SoS should embrace *increased* complexity. For those afraid of nuanced change this is something that may be viewed as counterintuitive. As a result, there may be a quasi life-cycle phenomenon at play here with a continual ebb and flow of an Organization's health between chaos and stability.[†] Because the organizations managing the systems of the SoS are assumed to be independent, the challenge of the Organization is to create policies, incentives and reward structures, or other conditions that influence the environment in which the component systems are being developed and operated in positive ways. This would help to achieve good things for the SoS, even when the separate system organizations and their managers pursue their own self-interests. Hence, one could say that the main job of the Organization is to "engineer the environment" for the benefit of all involved.

Clearly, too much information flow among the organizations of the SoS can cause friction and collapse of an SoS [51], particularly information from uninformed or disruptive sources that can cause confusion among recipients.[‡] An effective Organization needs to have the appropriate amounts and types of interorganizational communication.

Organizations can be both centralized and decentralized; complexity theory can help explain this paradoxical situation. The consideration of multiple "scales" or views may be helpful here. For example, the organization and its suborganizations and their intercommunications can be viewed at one (hierarchical) scale of abstraction. On the other hand much of the communications among individuals can be viewed at another, more ad hoc

* "In a complex society that has collapsed, it would thus appear, the overarching structure that provides support services to the population loses capability or disappears entirely. No longer can the populace rely upon external defense and internal order, maintenance of public works, or delivery of food and material goods... Remaining population must become locally self-sufficient..." "Collapse is a process of decline in complexity." [50, pp. 21, 31]

† "Four concepts lead to understanding collapse, the first three of which are the underpinnings of the fourth. These are: 1. human societies are problem-solving organizations; 2. sociopolitical systems require energy for their maintenance; 3. increased complexity carries with it increased cost per capita; and 4. investment in sociopolitical complexity as a problem-solving response often reaches a point of declining marginal returns." [50, p. 194]

‡ "Lewis [18] uses communication policy to give an organizational example of how complexity theory can be applied. He notes that some have suggested that the purpose of an organization is to reduce and block communication so things can get done rather than talked about. An organization that has no rules concerning communication will have 'total chaos and breakdown,' because every worker will be overwhelmed with memos, meetings, and telephone calls. Yet placing restrictions on communications (such as between departments) provides control and stability, yet 'little learning or change.'" [9]

(distributed), scale. The authors pose the (preferred) idea of a hybrid organization with both hierarchical (derived from a central control mentality) and distributed* or decentralized functions that are not limited to just distinct types of communication.

Malone states that decentralization is necessary in order to make an organization more agile and flexible in an increasingly competitive and dynamic world.[†] For example, the Department of Defense (DoD), a principal enterprise concerned with SoSs, with a tradition of hierarchical command and control but distributed execution (and adaptation) responsibility,[‡] is no exception.

Malone further expounds on the power of decentralization and individual freedom within organizations. Because of the relatively lower costs of communication compared to the past, it is possible to retain the advantages of a centralized but loose hierarchy while empowering staff. Then staff are free to innovate and achieve a variety of goals, many of which can be based on personal values as well as a corporate or directorate vision. However, consider the hypothesis: Complete decentralization will not work; it will cause disintegration. Recall the discussion of collapses of civilizations above. A hybrid organization with a management hierarchy to maintain “loose-coupled” communication among organizational subelements is seen as a minimum deterrent. The parallels to the Organizations of interest here are rather obvious.

* “It is no longer considered strange to read in the literature about organizational effectiveness as being described by variables that are either contradictory, or even paradoxical. Cameron writes about this explicitly: ‘To be effective, an organization must possess attributes that are simultaneously contradictory, even mutually exclusive’ [52, p. 545]. Using this framework, organizations can be both centralized and decentralized, both general and specialist, have both stability and adaptability, and diversified while “sticking to their knitting.” Complexity theory, particularly when used to explain the survivability and adaptability of biological systems, makes use of paradoxical explanations, many of which appear to have applicability to describing all complex adaptive systems, such as...organizations....Governing an organization is therefore an art,...” [9]

† “...by relentlessly reducing the costs of communications, new information technologies are taking us across a threshold into a place where dramatically more decentralized ways of organizing work become at once possible and desirable....We are, it seems, on the verge of a new world of work in which many organizations will no longer have a center at all—or, more precisely, in which they’ll have almost as many ‘centers’ as they have people.” “Now, we are in the early stages of another revolution—a revolution in business—that may ultimately be as profound as the democratic revolution in government.” “For the first time in history, technologies allow us to gain the economic benefits of large organizations, like economies of scale and knowledge, without giving up the human benefits of small ones, like freedom, creativity, motivation, and flexibility.” “The most extreme kind of business freedom occurs in markets because, in this kind of organization, no one is bound by a decision to which he or she doesn’t agree.” [53, pp. ix, 4, 7]

‡ This is embedded in U.S. air combat tactics and the U.S. military’s reliance on the initiative of noncommissioned officers, both of which differentiated U.S. Military and Soviet styles.

Perhaps one need not fear complete decentralization, especially in the context of SoSs. This may be a rare event, because it could be that the hierarchical management structure is self-perpetuating. Many of those in positions of power within an SoS hierarchy got there by adopting the command-and-control (alpha male) mentality of their bosses. Being so successful, why should they be motivated to change this culture? They may have very little personal vested interest in nurturing, viz., coordinating and cultivating, significant degrees of decentralization! Such managers have experienced success with certain types of behavior, thus, creating "instinctive" patterns of response. It is very hard to break behavior thus engrained in physically created neuronal pathways in the brain. One has to consciously, consistently create new pathways and cause the previous ones to disintegrate through atrophy to actually change the behavior and, thus, one's mental model or view of reality.*

Caveat: with such decentralization comes increased conflict—"turf battles." Current modeling research demonstrates that this can cause less integration (people preferring to be with others like themselves) and more "bumping" of group boundaries, thus creating more conflict e.g., factional fighting within Iraq, and initially after the fall of the Soviet Union, among former member republics. Prior to that strong centralized authority "forced" the various groups to cooperate. One should recognize that with an SoS, however, this decentralization can be viewed in reverse. More typically, the systems comprising the SoS and their respective organizations exist prior to the Organization itself. Thus, as opposed to taking an Organization and decentralizing it, the SoS is more accurately the result of a process of trying to aggregate the benefits of the various systems into an SoS. The Organization is then decentralized by default as a *fait accompli*. But as a decentralized operation there can still be problems.

In some ways the decentralized system organizations of an SoS can be thought of as a rhizome.[†] This rhizome paradigm is powerful, but we still

* "When people are making their own decisions, for instance, rather than just following orders, they often work harder and show more dedication and more creativity." "We need to shift our thinking from command-and-control to coordinate-and-cultivate.... rigid standards in one part of a business system can sometimes—paradoxically—allow much more flexibility and freedom in other parts of the same system." "...workers are being rewarded not for efficiently carrying out orders but for figuring out what needs to be done and then doing it." "When they have more freedom, their work becomes more interesting and enjoyable, and they're better able to juggle the various demands that life places on them." "Why can't power, ownership, and initiative be distributed throughout a whole market, rather than being imposed from the top of a hierarchy?" "Decentralize when the motivation and creativity of many people is critical." "What percentage of the intelligence and creativity of the people in your own organization do you think your organization actually uses?" [53, pp. 10–12, 35, 76, 121, 153]

† "In order to resolve the deficiencies fundamental to the structure of hierarchy, we must, by definition, abandon hierarchy as an organizing principle. We must confront hierarchy with its opposite: ... Rhizome acts as a web-like structure of connected but

need *some* organizational functions to be handled in a hierarchical fashion (i.e., a hybrid organization that is a mix of hierarchy and decentralization might work well). One needs to ask continually what combinations of the respective roles of hierarchy and decentralization might work the best from a systems engineering point of view within the context of an Organization and the comprised system organizations.

One function of the hierarchical portion of an organization (i.e., mainly the Organization in this context) would seem to be to continually try to track and shape the decentralized portion (i.e., the systems organizations)* while ensuring that the overall vision, mission, and policies of the Organization are being followed. This is probably much easier said than done.

The potential power of the systems organizations in functioning as a rhizome, to help the Organization, is quite promising.[†] This suggests that the communications functions of the hierarchical and decentralized portions of a hybrid organization should be distinct. A good question to ponder: How

independent nodes, borrowing its name from the structures of plants such as bamboo and other grasses. . . . Each node . . . stands autonomous from the larger structure, but the nodes work together in a larger network that extends benefits to the node without creating dependence. The critical element of a world that focuses power at the level of the individual, . . . while providing the flexibility and potential to achieve greater goals, remains the small, connected and relatively self-sufficient node of this rhizome structure. In human terms, such a node represents an economic and a cultural unit at the size preferred by [us]: the household and the tribe. Functionally self-sufficient but not isolated, cooperating but not controlled, the rhizome economy, combined with a self-awareness of control structures, provides the real-world foundation of stability and freedom.” [54]

* “Centralization and stratification produce ever-greater losses in efficiency due to the increased cost of distribution, coordination and communication. Hierarchy has incredible strength, but the accompanying inflexibility and top-heaviness can make it brittle and unstable. The networked, rhizome structure not only facilitates greater individual freedom, it also creates a more flexible and resilient structure for human ecology. The resiliency of rhizome may prove the deciding factor in our long-term survival as humanity encounters a host of potential threats. In the face of super-viruses, climate-change and overpopulation, the richer, more complex, more rhizomatic ecosystem has historically demonstrated greater survivability.” [54]

† “With a foundation of self-sufficiency established, a node can take advantage of a second strength of the rhizome pattern: network. Loose network connections, such as those in rhizome structures, actually demonstrate far more efficiency at information transfer and processing than the close, authoritarian connections of hierarchies, according to complexity theorist Mark Buchanan [*Nexus: Small Worlds and the Groundbreaking Theory of Networks*, Mark Buchanan]. The more intense, closely held connections within hierarchy prevent information from quickly spreading among large or diverse groups. The weaker, more distributed connections of a network can more quickly disseminate information to a much broader audience....” [54] There are interesting parallels here with point-to-point networks. Fixed structure can be more efficient/effective within the constraints of the specific conditions for which it was designed. But it is not resilient/adaptable.

would these functions be partitioned between the Organization and the systems organizations?

Unfortunately, "... most people are still more comfortable thinking about organizations in fixed, mechanical terms rather than in adaptive, decentralized terms." [55, p. 29] This is a holdover from the Industrial Revolution of over a century ago. We are now coming full circle in understanding the centrality of biological systems in understanding how systems work. The classical hierarchical management structure probably is well ingrained as the correct construct in the thinking of most managers. Do we have to wait for a new generation of leaders for this mindset to change? Hopefully, no!

Progress in an SoS can be related to a process of "variation and selection"*(see [Figure 3.1](#)) to continually try to find the right balance as one progresses toward more capability (a.k.a. complexity), e.g., a shaped and enhanced combination of more innovation and integration. Emergence, e.g., "surprises" coming from the interaction of parts of the whole, and the interaction of the whole with its environment, is responsible for this. One cannot predict what these interactions will produce.

Decentralization of SoS organizations implies that some groupings of individuals will form and disband dynamically in attempts to, among other things, manage an SoS. This presents a host of sociological challenges.[†] One implication of the human nature variables with regards to the more creative thinking members of a team is likely to be possible conflict within the team. "Outside-the-box" thinking often can engender fear and antagonism from other team members, resulting in alienation of the more creative members of the team and general disruption of the group effort, as mentioned earlier regarding rigidity of thought.

* "Harnessing complexity requires taking advantage of variety rather than trying to ignore or eliminate it." "In complex systems, it is difficult to determine what should be rewarded or which choice is appropriate.... In the short run we are not likely to have a direct approach that 'gets it completely right.'" [55, pp. 130, 137, 138]

† "Abstract: The five management concepts under the new decentralization paradigm present several organizational problems. These concepts are founded on the idea of the inherent goodness of human nature, which is unrealistic and impractical. Despite this basic flaw, the new decentralization paradigm is expected to govern the restructuring of many businesses." "The five concepts are: 1) 'radical' restructuring; 2) customer and process orientation; fractal and modular factories; 4) virtual corporation; and 5) atomized organization." "...their design is vague, contradictory and nonoperational....lacking a theoretical foundation....untested hypotheses....[from HRM perspective] barriers crop up which are hard or impossible to surmount and which are, above all, a result of the limited knowledge and abilities of the necessary personnel available and an all too idealistic notion of human nature underlying these new concepts." [56]

Clearly, decentralization presents administrative challenges to SoS organizations, as well.* Even in nonprofit organizations, certain elements are appropriate for the hierarchical portion of the management structure (e.g., formulation and communication of organizational vision, overall mission, and general strategies for attaining same; funding/salary administration and capital budget allocations).

Less central control by the Organization comes with a greater dependency on organizational individuals to work well within small groups, which, in turn, are expected to make independent decisions that move the systems organizations toward collectively stated goals. It is hoped that the individuals will be able to set aside some personal motivations and compensate for limiting personality characteristics, and act cooperatively with others within their own group and with other groups with whom they interact. This requires that they be able to embrace uncertainty and ambiguity. Also, this presents a challenge within the DoD and other federal agencies. For example, there is a strong traditional tendency to treat the control of information as a source of power and institutional or political advantage. Thus, despite high-managerial-level mandates to share information across institutional boundaries to achieve enterprise goals,[†] the reality of residual cultural bias at the working level prevents this from happening in meaningful ways.

However, getting people to actually set aside their personal view of their reality and take on alternate perspectives of others has been shown often to be a very difficult task.[‡] How well one is able to understand or process varying types of information can determine the success of accepting or taking on multiple perspectives.

* "...it is likely that a growing need for coordination and an increase in transaction costs will result from a decentralization of decision making and enterprise functions." A "plausible structural element" used to characterize the paradigm: "...(12) complementary central supervision of the decentralized units—at least at the strategic level; central, profit-oriented supervision...by means of financial control." [56]

† Wolfowitz, Paul, 2 December 2004, "Data Sharing in a Net-Centric Department of Defense," Department of Defense, ASD(NII)/DoD CIO, Number 8320.2

‡ "Breaking down our associative barriers is the first challenge we face... But how do we do it?... In essence,...people succeeded at breaking down...barriers because they did one or more of the following things: 1) Exposed themselves to a range of cultures; 2) Learned differently; 3) Reversed their assumptions; and 4) Took on multiple perspectives." [57, p. 45]

But while it will help innovation to add diversity,* this will also increase conflict within the group. How does one compensate for this? This highlights the particular challenge for small groups to truly embrace those that are different, especially those who offer relatively radical ideas.[†]

In complex (enterprise) environments like that of an SoS, it is better to have an opportunity exploration mindset as opposed to a risk mitigation mindset. Usually, money is not the best, most effective, long-term reward or motivation for individuals, as stated in other sources. But monetary compensation should be at a comfortable level for the individual, and outstanding performance should be recognized in a way that is personally meaningful for the individual concerned. Whether it is through external validation or through the challenge of the work itself, understanding individually tailored motivations is essential. Another problem can be jealousy within the group. Thus, one also needs to consider how jealousy can be mitigated to an acceptable level so that it does not jeopardize the group working as a team.

Thus, it would seem that group performance can be improved, particularly in utilizing the more creative thinkers in the organization or group, by training everyone in the three tenets of a learning organization culture, viz., integration, collaboration, and proactiveness.[‡] Such training might be

* "Working with a diverse group of people, then, is a great way to increase creativity." "What is surprising...is not that people are attracted to people who are similar; this is something we know from personal experience. What is surprising is how predictable this effect is...." so much so "that it can be expressed through a regression equation. The similar-attraction effect can have a devastating impact on our efforts to create diverse teams." "If you are thinking about recruiting a candidate because 'I like her' or 'He's just like one of us,' these might actually be reasons not to hire the person, assuming the job or team requires creativity." [57, pp. 80-82]

† "So how do you reward failure?... 1) Make sure people are aware that failure to execute ideas is the greatest failure, and that it will be punished. 2) Make sure everyone learns from past failures; do not reward the same mistakes over and over again. 3) If people show low failure rates, be suspicious. Maybe they are not taking enough risks, or maybe they are hiding their mistakes, rather than allowing others in the organization to learn from them. 4) Hire people who have had intelligent failures and let others in the organization know that's one reason they were hired." "Explicit rewards, then, can be an effective way to kill off our creativity.... If intrinsic motivation is high, if we are passionate about what we are doing, creativity will flow. External expectations and rewards can kill intrinsic motivation and this kill creativity.... Stephen King puts it this way: 'Money is great stuff to have, but when it comes to the act of creation, the best thing is not to think of money too much. It constipates the whole process.'" [57, pp. 129, 138]

‡ "Building learning organizations requires personal transformations or basic shifts in how we think and interact. As W. Edwards Deming says, nothing happens without 'personal transformation.' And the only safe space to allow for this transformation is a learning community. But at the heart of any serious effort to alter how we operate lies a concern with three dysfunctions of our culture: fragmentation, competition, and reactivity.... [Fragmentation] The walls that exist in the physical world are reflections of our mental walls.... [Competition] Fascinated with competition, we often find ourselves competing with the very people with whom we need to collaborate.... Our overemphasis on competition makes looking good more important than being good. The fear of not

instigated by the Organization but include key representatives of the system organizations. Initial success would depend on some level of up-front trust, but if properly implemented, interorganizational trust would tend to increase, raising the prospects of ultimate success in creating a cadre of high performers within an SoS team.

Trust needs bonding, particularly among individuals from different organizational groups.* Trust needs personal contact outside of meetings to develop.[†] Trust is a huge topic that cannot be treated adequately here; more can be found in [10] and many papers in the trust literature.

In regards to the topic of SoS emergence, any characteristic or property of an SoS that is not observed in any proper subset of the systems comprising the SoS, and *only* such characteristics, can be called emergent. Everyone cited above would seem to agree on this definition of SoS emergence.

Such an emergent (contrary to the authors' preferred definition of emergence, of course) property is probably either there "by design" or "implied" by the intentional aggregation of some subset of systems in an SoS. Recall that, by definition, an SoS achieves some purpose that none of the subsets of the SoS can achieve; that is why the SoS was assembled in the first place. This is fine and good but it is rather boring! The fascinating thing about complex systems in general and an SoS in particular is the unexpected, even surprising, emergent properties. One had better pay particular attention to those or risk paying dearly for the promises offered by the SoS opportunities if something goes very wrong. This is why the authors define emergence to be unexpected and focus on the surprises. In proofing the final version of this chapter, the authors learned of a paper on emergence offering a similar message [60].

looking good is one of the greatest enemies of learning. To learn, we need to acknowledge that there is something we don't know, and to perform activities that we're not good at. [Reactivity] We have grown accustomed to changing only in reaction to outside forces, yet the well-spring of real learning is aspiration, imagination, and experimentation." [58]

* "Trust needs bonding. Self-contained units responsible for delivering specified results are the necessary building blocks of an organization based on trust, but long-lasting groups of trusties can create their own problems, those of organizations within the organization. For the whole to work, the goals of the smaller units have to gel with the goals of the whole. Paradoxically; the more virtual the organization, the more its people need to meet in person.... Trust is not and never can be an impersonal commodity." [59, p. 46]

+ "Trust needs touch.... A shared commitment still requires personal contact to make it real.... The meetings, however, are different. They are more about process than task, more concerned that people get to know each other than that they deliver. Videoconferences are more task focused, but they are easier and more productive if the individuals know each other as people, not just as images on the screen." [59, p. 46] Interestingly, some research suggests this has changed for the next generation(s)—the "digital natives" who are growing up with web-based technologies and mindsets, e.g., akin to using Second Life avatars (http://en.wikipedia.org/wiki/Second_Life).

3.4 Conclusion

Each of us sees a distinct perception of reality. New perceptions can arise from changes in the components of a new interpretation of scale: {view} = {scope, granularity, mindset, timeframe}. To improve the practice of systems engineering, continual surprise is the most important aspect of emergence that results from multiview analysis, especially in SoS environments.

Many problems concerning organization constructs, leadership roles, decentralization, group dynamics, individual behavior, and trust are raised. Although not all these issues are new, the authors feel they are not receiving enough attention in the context of engineering of an SoS. It seems clear that complexity theory and/or systems science is not only applicable but perhaps critical in solving some of these problems. Several areas of research that could greatly improve progress toward the goal of the effective/efficient development and sharing of information in such environments are suggested in [10].* The results of such research also could positively impact knowledge management and the return on investment in systems engineering in general.

References

1. White, B. E. 2006. Fostering intra-organizational communication of enterprise systems engineering practices. National Defense Industrial Association, 9th Annual Systems Engineering Conference, San Diego CA, 23–26 October 2006.
2. White, B. E. 2007. On Interpreting scale (or view) and emergence in complex systems engineering. 1st Annual IEEE Systems Conference, Honolulu, HI, 9–12 April 2007.
3. Isaacson, W. 2007. *Einstein—His Life and Universe*. Simon and Schuster, New York.
4. Broks, P. 2006. The big questions: what is consciousness? NewScientist.com news service, from issue 2578 of *New Scientist* 18, November 2006, pp. 56–61.
5. Ryan, A. personal communication, 15 September 2006.
6. Kuras, M. L. 2007. Complex-System Engineering, Symposium on Complex Systems Engineering. RAND Corporation, Santa Monica, CA.

* Some of these are repeated here:

(1) What are the most effective/efficient combinations of the respective roles of hierarchy and decentralization in a hybrid organization? (2) Assuming the communications functions of the hierarchical and decentralized portions of a hybrid organization are distinct, how should these functions be partitioned? (8) How can neuroanatomy/evolutionary biology research be used in conjunction with research about mirror neurons to develop new training programs that get people to “break” rigid constructs of reality and their “instinctive” reactions that may no longer be appropriate? (10) What are some good techniques for balancing the strength of diversity with the potential for conflict within small groups? (11) How can jealousy within a group be mitigated to an acceptable level so that it does not jeopardize the group working as a team? (17) To what extent might individual behavior and small group dynamics scale to the larger organization and/or an encompassing enterprise? (18) How might one motivate individuals or shape their behavior, leveraging common good traits, to approach organizational or institutional goals?

7. Kuras M. L. and B. E. White. 2005. Engineering enterprises using complex-system engineering. INCOSE Symposium, Rochester, NY, 10–14 July 2005.
8. Pilek, P. E., C. Lindberg, and B. Zimmerman. 1997. Some Emerging Principles for Managing Complex Adaptive Systems. 23 September, 1997. <http://www.directed-creativity.com/pages/ComplexityWP.html>.
9. Grobman, G. M. 2005. Complexity theory: a new way to look at organizational change. *Public Administration Quarterly* 29:350–382.
10. McCarter, B. G. and B. E. White. 2007. Collaboration/cooperation in sharing and utilizing net-centric information, conference on systems engineering research (CSER). Stevens Institute of Technology, Hoboken, NJ, 14–16 March 2007.
11. DeRosa, J. K., G. Rebovich, and R. S. Swartz. 2006. An enterprise systems engineering model. INCOSE Symposium, Orlando, FL, 9–16 July 2006.
12. Gharajedaghi, J. 2006. *Systems Thinking, Second Edition: Managing Chaos and Complexity: A Platform for Designing Business Architecture*, 2nd Edition. Elsevier, London. <http://www.elsevier.com/wps/find/bookdescription.authors/706822/description#description>.
13. Ludeman, K. and E. Erlandson, 2004. Coaching the alpha male. *Harvard Business Review*, May, 2004, p. 58.
14. White, B. E. 2006. Enterprise opportunity and risk. INCOSE Symposium, Orlando, FL, 9–16 July 2006.
15. Ramachandran, V. S. and L. M. Oberman. 2006. Broken mirrors: a theory of autism—studies of the mirror neuron system may reveal clues to the causes of autism and help researchers develop new ways to diagnose and treat the disorder. *Scientific American*, Special Section: Neuroscience, November 2006, pp. 63–69, www.sciam.com.
16. Rizzolatti, G., L. Fogassi, and V. Gallese. 2006. Mirrors in the mind: A special class of brain cells reflects the outside world, revealing a new avenue for human understanding, connecting and learning. *Scientific American*, Special Section: Neuroscience, November 2006, pp. 54–61, www.sciam.com.
17. Ludeman, K. and E. Erlandson. 2004. Coaching the alpha male, *Harvard Business Review*, May, 2004, p. 58.
18. Lewis, R. 1994. From chaos to complexity: implications for organizations. *Executive Development* 7:16–17.
19. Mackenzie, D. 2005. End of the enlightenment. *NewScientist.com* news service, October 2005.
20. Damasio, A. 1994. *Descarte's Error—Emotion, Reasons, and The Human Brain*. Penguin Books, Penguin Group, New York.
21. Society for Neuroscience. 2007. Adult Neurogenesis, brain briefings. Society for Neuroscience, June 2007 http://www.sfn.org/index.cfm?pagename=brainBriefings_adult_neurogenesis&print=on.
22. Gould, E. 2007. How widespread is adult neurogenesis in mammals? *Nature Reviews Neuroscience* 8:481–488.
23. Geddes, L. 2007. Adult-formed brain cells important for memory. *NewScientist.com* news service, 23 May 2007.
24. Geddes, L. 2007. Magnets may make the brain grow stronger. *NewScientist.com* news service, 24 May 2007.
25. Ryan, A. J. 2006. Emergence is coupled to scope, not level. *Nonlinear Sciences*, abstract, <http://arxiv.org/abs/nlin.AO/0609011>.
26. Sheard, S. A. personal communication, 2006.
27. Koch, C. and G. Laurent. 1999. Complexity and the nervous system. *Science* 284:96–98, <http://www.sciencemag.org>.

28. Johnson, S. 2001. *Emergence—The Connected Lives of Ants, Brains, Cities, and Software*, Scribner, New York, p. 127.
29. Phillips, H. 2006. Instant expert: the human brain. *NewScientist.com* news service, 4 September, 2006, <http://www.newscientist.com/article.ns?id=dn9969&print=true>.
30. Johnson, C. W. 2005. What are Emergent Properties and How Do They Affect the Engineering of Complex Systems? <http://www.dcs.gla.ac.uk/~johnson/papers/emergence.pdf>.
31. Davies, P. 2005. Higher laws and the mind-boggling complexity of life. *New Scientist* Print Edition, <http://www.newscientist.com/article.ns?id=mg1852489.1.000&print=true>.
32. Abbott, R. J. 2006. If a tree casts a shadow is it telling the time? Department of Computer Science, California State University, Los Angeles, CA, 14 July 2006.
33. Kuras, M. L. personal communication, 11 August 2006.
34. Snyder, A. 1999. Game, mindset and match. Number 10,947, *The Weekend Australian*, 4–5 December 1999.
35. Clark, B. 2001. *Growing Up Gifted: Developing the Potential of Children at Home and at School*, 6th Edition. Prentice Hall, New York.
36. Gladwell, M. 2005. *Blink: The Power of Thinking Without Thinking*. Little, Brown and Company, Time Warner Book Group, New York.
37. Damper, R. I. 2000. Emergence and levels of abstraction. Editorial for the Special Issue on 'Emergent Properties of Complex Systems,' *International Journal of Systems Science* 31(7) 2000:811–818.
38. Abbott, R. J. 2005. Emergence explained: getting epiphenomena to do real work. Department of Computer Science, California State University, Los Angeles, CA, 12 October 2005.
39. Taleb, N. N. 2007. *The Black Swan—Impact of the HIGHLY IMPROBABLE*, Random House, New York.
40. Cohen, M. 1999. Commentary on the organizational science special issue on complexity. *Organization Science* 10:373–376.
41. Fromm, J. 2005. Types and forms of emergence. *Nonlinear Sciences*, abstract, nlin.AO/0506028, 13 June 2005.
42. Bar-Yam, Y. 2003. Emergence, Concepts in Complex Systems. <http://necsi.org/guide/concepts/emergence.html>.
43. Buchanan, M. 2007. Predicting change, not a moment too soon, *New Scientist*, 11–17 August 2007, p. 9.
44. Ryan, A. and D. O. Norman, personal communications, 21 September 2006.
45. Mogul, J. C. 2005. Emergent (mis)behavior vs. complex software systems, HPL-2006-2, HP Laboratories, Palo Alto, CA, 22 December 2005.
46. Boardman, J. and B. Sauser. 2007. System of Systems—the meaning of of. Symposium on Complex Systems Engineering, RAND Corporation, Santa Monica, CA, 11–12 January 2007.
47. Abbott, R. J. 2007. Emergence and systems engineering: putting complex systems to work. Symposium on Complex Systems Engineering, RAND Corporation, Santa Monica, CA, 11–12 January 2007.
48. Weeks, A., S. Stepney, and F. Polack. 2007. Neutral emergence: a proposal. Symposium on Complex Systems Engineering, RAND Corporation, Santa Monica, CA, 11–12 January 2007.
49. Ronald, E. M. A. 1999. Design, observation, surprise!—a test of emergence. *Artificial Life* 5:225–239.
50. Tainter, J. A. 1988. *The Collapse of Complex Societies*. Cambridge University Press, Cambridge.

51. De Landa, M. 1991. *War in the Age of Intelligent Machines*. Swerve Editions, New York.
52. Cameron, K. 1986. Effectiveness as paradox: consensus and conflict in conceptions of organizational effectiveness. *Management Science* 32:539–553.
53. Malone, T. W. 2004. *The Future of Work—How the New Order of Business Will Shape Your Organization, Your Management Style, and Your Life*. Harvard Business School Press, Boston, MA.
54. Vail, J. 1964. *A Theory of Power*, Chapter 9: Forward, to Rhizome. iUniverse, Inc., New York, <http://www.jeffvail.net/2004/10/theory-of-power-chapter-9.html>.
55. Axelrod, R. and M. D. Cohen. 2000. *Harnessing Complexity: Organizational Implications of a Scientific Frontier*. Basic Books, New York.
56. Drumm, H. J. 1995. The paradigm of a new decentralization: its implications for organization and HRM. *Employee Relations* 17(8):29–46.
57. Johansson, F. 2004. *The Medici Effect – Breakthrough Insights at the Intersection of Ideas, Concepts, and Cultures*. Harvard Business School Press, Boston, MA, 2004.
58. Senge, P. M. 1999. Personal transformation. Society for Organizational Learning, <http://www.solonline.org/res/kr/transform.html>.
59. Handy, C. 1995. Trust and the virtual organization: how do you manage people whom you do not see? *Harvard Business Review*, May–June 1995.
60. McConnell, G. R. 2000. Emergence: A challenge for the systematic. INCOSE Symposium, Minneapolis, 16–20 July 2000.

chapter four

A system-of-systems simulation framework and its applications

Ferat Sahin, Mo Jamshidi, and Prasanna Sridhar

Contents

4.1	Introduction.....	107
4.2	System of systems.....	108
4.3	System-of-systems simulation framework.....	115
4.3.1	DEVS modeling and simulation.....	115
4.3.2	XML and DEVS.....	118
4.4	SoS simulation framework examples	119
4.4.1	Case study 1: Data aggregation simulation	119
4.4.1.1	DEVS-XML format.....	119
4.4.1.2	Programming environment	120
4.4.1.3	Simulation results	121
4.4.2	Case study 2: a robust threat detection system simulation.....	123
4.4.2.1	XML format	123
4.4.2.2	Simulation setup	124
4.4.2.3	Robust threat detection simulation	126
4.5	Conclusion	130
	References	130

4.1 Introduction

There has been a growing recognition that significant changes need to be made in government agencies and industries, especially in the aerospace and defense areas [1,2,3]. Today, major aerospace and defense manufacturers such as Boeing, Lockheed-Martin, Northrop-Grumman, Raytheon, and BAE Systems include some version of “*large-scale systems integration*” as a key part of their business strategies [4]. In some cases, these companies have even established entire business units dedicated to systems integration activities [1].

In parallel, there is a growing interest in new system-of-systems (SoS) concepts and strategies. The performance and functioning of groups of heterogeneous systems have become the focus of various applications including military, security, aerospace, and disaster management systems [5,6]. There is an increasing interest in achieving synergy between these independent systems to achieve the desired overall system performance [7]. In the literature, researchers have addressed the issues of coordination and interoperability in an SoS [4,8,9]. In order to study SoS characteristics and parameters, one needs to have realistic simulation frameworks properly designed for system-of-systems architecture. There are some attempts to develop simulation frameworks for multiagent systems using discrete event simulation tools [10–16]. In these research efforts, the major focus is given to discrete event simulation (DEVS) architecture with JAVA. In [10,11], DEVS modeling is presented and discussed in detail. In [14,16], DEVS is combined with service-oriented architecture (SOA) together to create the DEVS/SOA architecture. In [16], DEVS state machine approach is introduced. Finally, DEVS Modeling Language (DEVSML) is developed by using XML-based JAVAL in order to simulate systems in a net-centric way with relative ease [14].

Based on DEVS and JAVA, a discrete event XML-based SoS simulation framework was recently introduced by the authors in [4,17]. In this chapter, we will extend this SoS simulation framework by designing an SoS problem with heterogeneous autonomous systems for threat detection and data aggregation.

4.2 System of systems

The concept of SoS is essential to more effectively implement and analyze large, complex, independent, and *heterogeneous* systems working (or made to work) cooperatively [4,7,18]. The main thrust behind the desire to view the systems as an SoS is to obtain higher capabilities and performance than that would be possible with a traditional system view. The SoS concept presents a high-level viewpoint and explains the interactions between each of the independent systems. However, the SoS concept is still at its developing stages [19–22].

Systems of systems are super systems comprised of other elements which themselves are independent complex operational systems and interact among themselves to achieve a common goal [4]. Each element of an SoS achieves well-substantiated goals even if they are detached from the rest of the SoS. For example a Boeing 747 airplane, as an element of an SoS, is not an SoS, but an airport is an SoS, or a rover on Mars is not an SoS, but a robotic colony (or a robotic swarm) exploring the red planet is an SoS [4]. Associated with SoS, there are numerous problems and open-ended issues which need a great deal of fundamental advances in theory and verifications. In fact, there is not even a universal definition among system engineering community even though there is a serious attempt to define SoS and create its standards.

Based on the literature survey on system of systems, there are several system of systems definitions [4,23–29]. All of the definitions of SoS have their own merits, depending on their application and domain. However, we will list the more common definitions (with primary focus and application domain) as reported in the literature:

Definition 1: In [24], systems of systems exist when there is a presence of a majority of the following five characteristics: operational and managerial independence, geographic distribution, emergent behavior, and evolutionary development. Primary focus: Evolutionary acquisition of complex adaptive systems. Application: Military.

Definition 2: In [23,25], systems of systems are large-scale concurrent and distributed systems that are comprised of complex systems. Primary focus: Information systems. Application: Private Enterprise.

Definition 3: In [26], enterprise systems of systems engineering (SoSE) is focused on coupling traditional systems engineering activities with enterprise activities of strategic planning and investment analysis. Primary focus: Information intensive systems. Application: Private Enterprise.

Definition 4: In [27], system-of-systems integration is a method to pursue development, integration, interoperability, and optimization of systems to enhance performance in future battlefield scenarios. Primary focus: Information intensive systems integration. Application: Military.

Definition 5: In [28], SoSE involves the integration of systems into systems of systems that ultimately contribute to evolution of the social infrastructure. Primary focus: Education of engineers to appreciate systems and interaction of systems. Application: Education.

Definition 6: In [29], in relation to joint war-fighting, system of systems is concerned with interoperability and synergism of Command, Control, Computers, Communications, and Information (C4I) and Intelligence, Surveillance, and Reconnaissance (ISR) Systems. Primary focus: Information superiority. Application: Military.

During the course of defining SoS, one may want to ask the following questions: How does an SoS differ from large-scale systems (LSS), multiagent systems (MAS) like system of robotic swarms, unmanned aerial vehicle (UAV) swarm, etc.? LSS is defined [30,31] as a system which can be decomposed into subsystems (leading to hierarchical control) or its output information can be distributed (leading to decentralized control). Within these definitions a LSS is not an SoS, since the “systems” of a LSS as an SoS cannot operate independently like a robotic colony or an airport. In other words, a LSS does not create capability beyond the sum of individual capabilities of each system [32].

The MAS, on the other hand, are special cases of SoS that have a further unique property aside from having an emergent behavior; they have homogeneous system members like some robotic architecture, UAV models, etc. This observation is true even when agents in MAS do not communicate

with each other. More on system of systems and systems engineering can be found in [29,33–44].

Our favorite definition is “Systems of systems are large-scale concurrent and distributed systems that are comprised of complex systems.” The primary focus of this definition is information systems, which emphasizes the interoperability and integration properties of an SoS [4].

The *interoperability* in complex systems (i.e., multiagent systems) is very important as the agents operate autonomously and interoperate with other agents (or nonagent entities) to take better actions [17]. Interoperability requires successful communication among the systems. Thus, the systems should carry out their tasks *autonomously* as well as communicate with other systems in the SoS in order to take better actions for the over all goodness of the SoS, not just for themselves. The *integration* implies that each system can communicate and interact (control) with the SoS regardless of their hardware and software characteristics. This means that they need to have the ability to communicate with the SoS or a part of the SoS without compatibility issues such as operating systems, communication hardware, and so on [17]. For this purpose, an SoS needs a common language its components can speak. Without having a common language, the SoS components cannot be fully functional, and the SoS cannot be adaptive in the sense that new components cannot be integrated to the SoS without major effort. Integration also implies the control aspects of the SoS, because systems need to understand each other in order to take commands or signals from other SoS components [17].

One might argue that is why we cannot have a global state-space model of the SoS by combining the state-space models of the systems and subsystems in an SoS. Designing a state-space mathematical model for complex large-scale systems (for example, multiagent systems) is often difficult due to uncertainties in the operation and complex interactions among the systems (agents). In addition, the state-space representation of some systems may not be available. Such decentralized systems also require an efficient data handling (information processing) mechanism to capture the operational behavior of the system. The problem of characterizing the behavior and representation of such systems becomes even more difficult when each of these systems is heterogeneous and independently operational in nature. A naïve description of a system of system (SoS) is multiple instances of such complex heterogeneous operational independent systems working in synergy to solve a given problem as described in Definition 2.

In real-world systems, the problem is addressed in a higher level where the systems send and receive data from other systems in the SoS and make a decision that leads the SoS to its global goals. Let us take the military surveillance example, where different units of the army collect data through their sensors, trying to locate a threat or determine the identity of the target. In this type of situation, army command center receives data from heterogeneous sensor systems such as AWACS, ground RADARS, submarines, and so on. These systems are the parts of the SoS that makes the decision; say the command and control

station. Yet they may be developed with different technology. Thus, they will be different in hardware and/or software. This will create a huge barrier in data aggregation and data fusion using the data received from these systems, because they would not be able to interact successfully without hardware and/or software compatibility. In addition, the data coming from these systems are not unified, which will add to the barrier in data aggregation.

One solution of the problem using system of systems is to modify the communication medium among the SoS components. Two possible ways of accomplishing this task are:

- *Create a software model of each system* using the same software tool. In this approach, each component in the SoS talks to a software module embedded in itself. The software module collects data from the system and through the software model generates outputs and sends to the other SoS components. If these software modules are written with a common architecture and a common language, then the SoS components can communicate effectively regardless of their internal hardware and/or software architectures.
- *Create a common language to describe data*, where each system can express its data in this common language so that other SoS components can parse the data successfully.

The overhead that needs to be generated to have software models of each system on an SoS is enormous and must be redone for new members of the SoS. In addition, this requires the complete knowledge of the state-space model of each SoS component, which is often not possible. Thus, a data-driven approach would have better success on integrating new members to the SoS and also applying the concept to other SoS application domains.

In this chapter, we present an SoS simulation architecture based on Extensible Markup Language (XML) in order to wrap data coming from different sources in a common way. The XML language can be used to describe each component of the SoS and their data in a unifying way. If XML-based data architecture is used in an SoS, the only requirement is for the SoS components to understand/parse XML files received from the components of the SoS. Most complex systems in use by the military and government agencies have the processing and computational power to run an XML parser to process the data received from the components of the SoS.

In XML, data can be represented in addition to the properties of the data such as source name, data type, importance of the data, and so on. Thus, it does not only represent data but also gives useful information about the SoS to take better actions and to understand the situation better. The XML language has a hierarchical structure where an environment can be described with a standard and without a huge overhead. Each entity can be defined by the user in the XML in terms of its visualization and functionality. For example, a hierarchical XML architecture like in Listing 4.1 can be designed for an SoS so

Listing 4.1 An XML-Based System-of-Systems Architecture [4]

```
<!--Created 11/8/2006 Author @ Ferat Sahin -->
<?xml-stylesheet type="text/css" href="genericxml.css"?>

<systemofsystem>
    <id> Id of the System of Systems </id>
    <name> The name of the System of System</name>
    <system>
        <id>Id of the first system</id>
        <name> The name of the first system </name>
        <description> The description of the first system
            </description>
        <dataset>
            <Output>
                <id>Id of the first output</id>
                <data>Data of the first output</data>
            </Output>
            <Output>
                <id>Id of the second output</id>
                <data>Data of the second output</data>
            </Output>
        </dataset>
        <subsystem>
            <id>Id of the subsystem of the first System</id>
            <name>The name of the subsystem</name>
            <description>This is a subsystem of the system in
                an SoS</description>
            <dataset>
                <Output>
                    <data> Data of the subsystem </data>
                </Output>
            </dataset>
        </subsystem>
    </system>
</systemofsystem>
```

that it can be used in the components of the SoS and also be applied to other SoS domains easily. In Listing 4.1, the first line defines the name of the file that describes the functionality of the user-defined keywords used to define the SoS architecture. This file is mainly used for visualization purposes, so that any of the SoS components can display the current data or the current status of the SoS to a human/expert to make sure the proper decision is taken.

The first keyword of the XML architecture is “systemofsystem,” representing an SoS. Everything described after this keyword belongs to this SoS based on the XML architecture. The following keywords, “id” and “name”, are used to describe the system of systems. Then, the keyword “system” is used to declare and describe the first system of the SoS. In addition to “id” and “name”, two

more keywords, “description” and “dataset”, are used to describe the properties of the system and to represent the data coming out of this system. Data sources are denoted by “output” keyword, and data is provided with the keyword “data”. After representing data from two sources, a subsystem is described by the keyword “subsystem”. The subsystem and its data are presented in a similar manner. Other subsystems in this subsystem or in parallel to this subsystem, as well as additional systems, can be described in the system of systems.

Next, we present a case study concerning an experimental setup in which groups of heterogeneous sensors and a swarm of robots are working independently. When these systems work in conjunction, a centralized coordinating system (with a data store) should help in scalability as well as efficient data processing. These systems communicate with standard XML messages (such as using SOAP protocol) to the central coordinator that in turn delivers certain decision-making commands. Such data-driven communication and decision making process does not have to depend on the underlying architecture of individual systems. An example XML file is presented in Listing 4.2 for an SoS that has two systems and a subsystem in one of the systems.

Listing 4.2 XML Architecture for an SoS Consisting of Two Systems and a Subsystem

```
<!--Created 11/8/2006 Author @ Prasanna Sridhar -->
<?xml-stylesheet type="text/css" href="xmcss.css"?>

<systemofsystem>
    <id> 1 </id>
    <name> SoS-one</name>
    <system id="1">
        <name> System-one </name>
        <description> This is a system within an SoS
        </description>
        <dataset>
            <node id="100">
                <data>25, 50%, 600Lux</data>
            </node>
            <node id="102">
                <data>40, 90%, 800Lux</data>
            </node>
        </dataset>
        <subsystem id="1">
            <name> Sub-system</name>
            <description> This is a sub-system within a
            system in an SoS</description>
            <dataset>
                <data> 100 </data>
            </dataset>
        </subsystem>
    </system>
```

(continued)

Listing 4.2 XML Architecture for an SoS Consisting of Two Systems and a Subsystem (continued)

```
<system id="2">
    <name> System-two </name>
    <description> This is a second system within the
        same SoS </description>
    <dataset>
        <data>10</data>
    </dataset>
</system>
</systemofsystem>
```

The representation in Listing 4.2 is important for not only representing data but also visualizing the data and the communication messages in the SoS. This is especially important when a human expert oversees and/or controls the SoS, because effective visualization of the data helps the human expert tremendously in understanding the internal communications of the SoS. Figure 4.1 illustrates how Listing 4.2 will be visualized using Internet Explorer.

The SoS architecture for the integration among the systems should be simulated and tested before they are put in use in the field. The next section

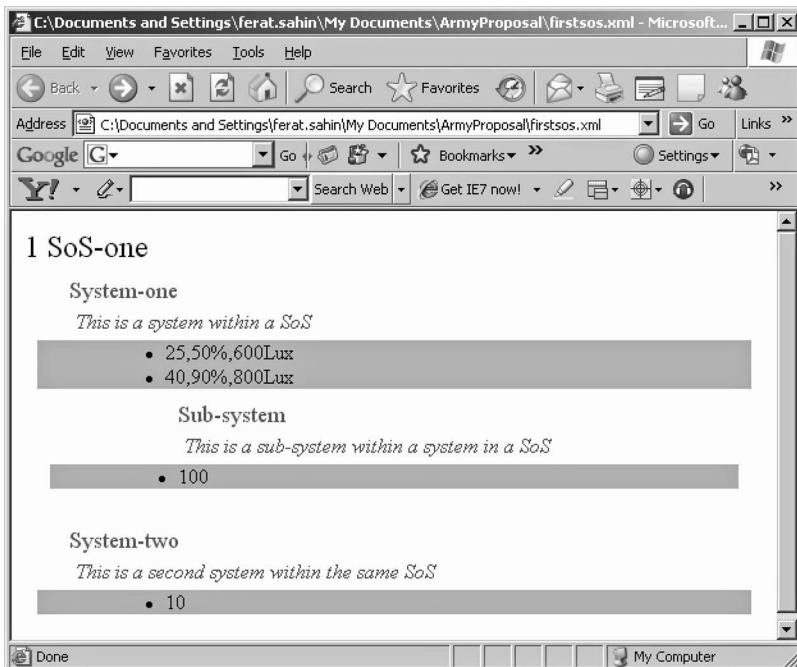


Figure 4.1 Visualization of Listing 4.2 in Internet Explorer.

presents a suitable simulation environment that can accommodate asynchronous interactions and data exchange using XML.

4.3 System-of-systems simulation framework

In a system of systems, systems and/or subsystems often interact with each other because of interoperability and overall integration of the SoS. These interactions can be achieved by efficient communication among the systems either by using peer-to-peer communication or through a central coordinator in a given SoS. Since the systems within SoS are operationally independent, interactions between systems are generally asynchronous in nature. A simple yet robust solution to handle such asynchronous interactions (specifically, receiving messages) is to throw an event at the receiving end to capture the messages from single or multiple systems. Such system interactions can be represented effectively as discrete-event models [4,17]. In discrete-event modeling, events are generated at variable time intervals as opposed to some predetermined time interval seen commonly in discrete-time systems. More specifically, the state change of a discrete-event system happens only upon arrival (or generation) of an event, not necessarily at equally spaced time intervals. To this end, a discrete-event model is a feasible approach in simulating the SoS framework and its interaction [4,17,45]. Several discrete-event simulation engines [46–48] are available that can be used in simulating interaction among heterogeneous mixture of independent systems. We consider one such simulation framework—*Discrete Event System Specification* (DEVS) because of the available effective mathematical representation and its support to distributed simulation using the Department of Defense (DoD) High Level Architecture (HLA) [49].

4.3.1 DEVS modeling and simulation

Discrete Event System Specification (DEVS) [10] is a formalism, which provides a means of specifying the components of a system in a discrete-event simulation. In DEVS formalism, one must specify *basic models* and how these models are connected together. These basic models are called *atomic* models, and larger models which are obtained by connecting these atomic blocks in meaningful fashion are called *coupled* models (shown Figure 4.2). Each of these atomic models has *imports* (to receive external events), *outports* (to send events), set of *state variables*, *internal transition*, *external transition*, and *time advance functions*. Mathematically it is represented as 7-tuple system: $M = \langle X, S, Y, G_{in}, G_{ext}, h, t_a \rangle$, where X is an input set, S is set of states, Y is set of outputs, G_{in} is internal transition function, G_{ext} is external transition function, h is the output function, and t_a is the time advance function. The model's description (implementation) uses (or discards) the message in the event to do the computation and delivers an output message on the outport and

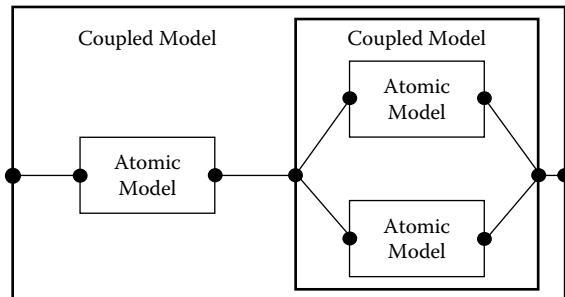


Figure 4.2 DEVS model representing system and subsystems [4,17].

makes a state transition. A JAVA-based implementation of DEVS formalism, DEVSJAVA [50], can be used to implement these atomic or coupled models. In addition, DEVS-HLA [51], based on HLA, will be helpful in distributed simulation for simulating multiple heterogeneous systems in the system-of-systems framework.

The DEVS has been used extensively by the authors, as a part of a funded research at the University of New Mexico (in collaboration with University of Arizona and Jet Propulsion Laboratory) producing several research papers [51,52].

Next we present a simulation example, shown in [Figure 4.3](#), using the DEVS simulation environment [50]. This is a scenario where a target is being tracked by a group of sensors deployed in the region of interest. The sensors are grouped to form clusters, with each cluster having a cluster-head that can send and receive information to a command center (base-station). The entire simulation can be performed by “coupling” all the DEVS atomic models together with meaningful interconnections as shown in Figure 4.3.

Multiagent robotic systems can also be successfully demonstrated using the DEVS formalism and DEVSJAVA software [50]. A multiagent simulation framework, Virtual Laboratory (V-Lab[®]) was developed by the authors for such multiagent simulation with DEVS modeling framework [51,52]. The 3-D simulation snapshots of the multiagent robotic systems simulation is presented in [Figure 4.4](#). Figure 4.4 (a) and (b) show four robots navigating through the obstacles to reach to the target location. The robots avoid obstacles using simulated IR sensors, shown as gray lines on the robots. Figure 4.4 (c) and (d) show the simulation of the same concept in a dynamic environment where obstacles can also move. The DEVS environment can be successfully used to simulate more complex problems such as cooperative autonomous systems in a system-of-systems framework.

As described in the SoS simulation framework, we use XML-based language in order to represent and transfer data among the systems in the SoS. In order to test the SoS architecture in the DEVS environment, the XML-based language needs to be embedded into the simulation environment. The

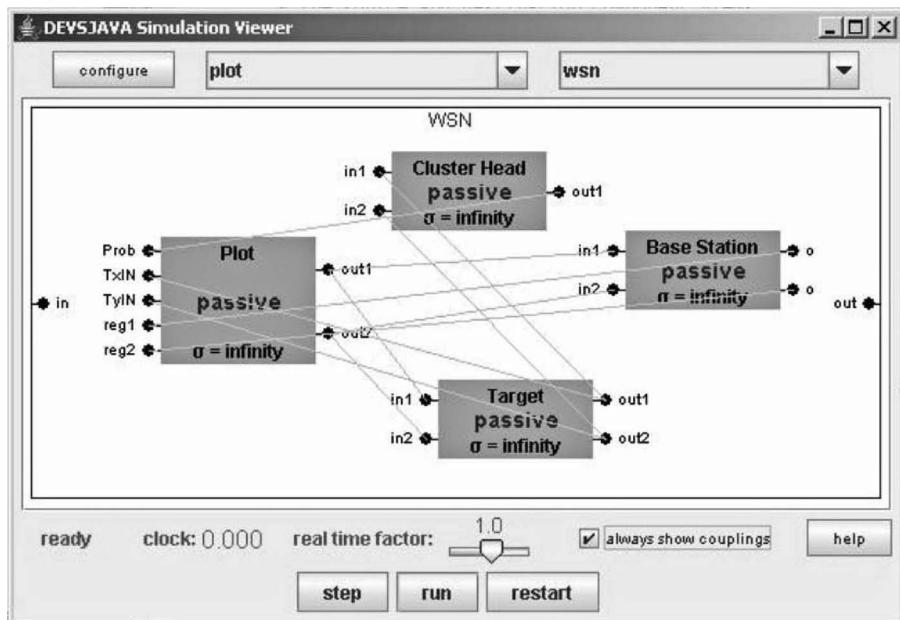


Figure 4.3 A DEVS simulation example for target tracking [4].

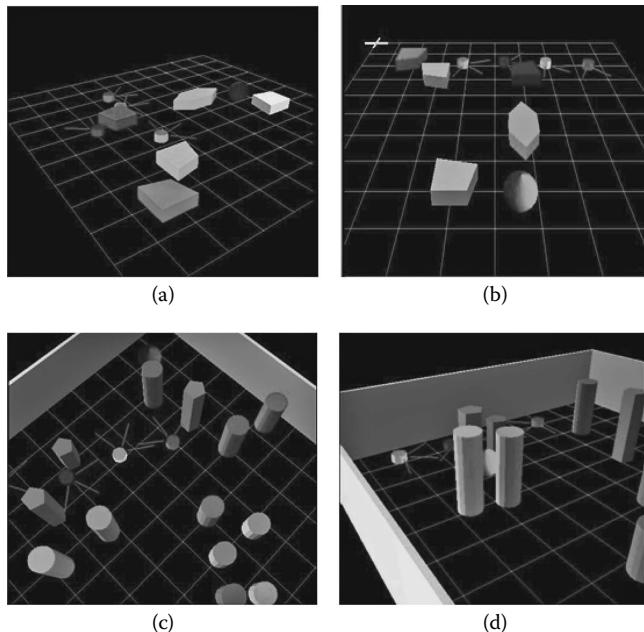


Figure 4.4 Snapshots of the multiagent robotic systems simulation.

following section explores how XML and DEVS environment can be combined in the simulation environment.

4.3.2 XML and DEVS

In DEVS, messages can be passed from one system (coupled or atomic model) to another using either predefined or user-defined message formats. Since the systems within SoS may be different in hardware and/or software, there is a need for a unifying language for message passing. Each system need not necessarily have the knowledge (operation, implementation, timing, data issues, and so on) of another system in an SoS. Therefore, one has to work at a high level (information or data level) in order to understand the present working condition of the system. One such good fit for representing different data in a universal manner is XML. Figure 4.5 describes conceptually an SoS simulation example to demonstrate the use of XML as a message passing paradigm using DEVS formalism.

In Figure 4.5, there are three systems in a hierarchy where systems A and B send and receive data from system C. System C sends and receives data from a higher level as described in the message of system C. The data sent by system C has data from systems A and B. In addition, it has information about system A and B being system C's subsystems.

With the above-mentioned XML-based integration architecture and the DEVS environment, we can then offer solutions to system of systems which have heterogeneous complex systems such as mobile sensor platforms or microdevices.

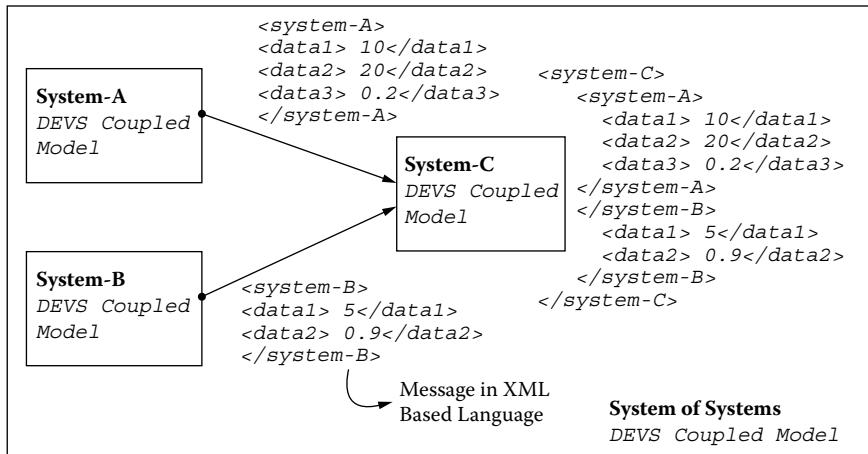


Figure 4.5 A SoS simulation example with three systems and XML-like message passing [4,18].

4.4 SoS simulation framework examples

Based on the presented system-of-systems architecture, we have simulated two cases. The first case is a data aggregation scenario where there is a base robot, a swarm robot, two sensors, and a threat. The second case is a robust threat detection scenario with a base robot, two swarm robots, five sensors, and a threat.

4.4.1 Case study 1: Data aggregation simulation

In this scenario, the sensors are stationed such that they represent a border. The threat is moving in the area and it can be detected by the sensors. When the sensors detect the threat, they notify the base robot. Then the base robot notifies the swarm robot the location of the threat based on the information sent by the sensors.

4.4.1.1 DEVS-XML format

As mentioned before, the hardware and/or software architectures of these systems will not be the same in reality. Thus, they may not be able to talk to each other successfully, even though they can operate independently. We have implemented XML-based SoS message architecture in DEVSJAVA software [50]. In this XML-based message architecture, each system has an XML-like message consisting of their name and a data vector. The names are used in place of the XML keywords. The data vectors are used to hold the data of the systems. The length of the vectors in each system can be different, based on the amount of data each system contains. For example, the XML message of the sensors has the sensor coordinates and the threat level. The threat level is set when a threat gets in the coverage area of the sensors. On the other hand, the base robot's XML message has its coordinates, the threat level, and the coordinates of the sensor who is reporting a threat. Thus, the data vector length of base robot's XML message has five elements, whereas the data vector of an XML message of a sensor has three elements. Table 4.1 presents the names and the length of the vector data of each system in the system of systems.

Table 4.1 XML Message Components for the Systems in the SoS

System	Name	Vector Data Length
Base robot	“Base Robot”	5 ($X, Y, Threat, Xt, Yt$)
Swarm robot	“Swarm Robot”	5 ($X, Y, Threat, Xt, Yt$)
Sensor	“Sensor 1”	3 ($X, Y, Threat$)
Sensor	“Sensor 2”	3 ($X, Y, Threat$)
Threat	“Fire”	2 (X, Y)

The data vectors are made of “double” variables in order to keep track of the positions accurately. The “threat” element in the data vector is a flag representing threat (1.0) or no threat (0.0). The elements X_t and Y_t are used for the destination coordinates in the XML messages of the base robot and swarm robot. These represent the coordinate of the sensor who reported a threat.

4.4.1.2 Programming environment

In DEVSJAVA environment, a data structure, called “XmlEntity” is created based on the “entity” data structure in order to create XML messages of each system as shown in Listing 4.3. This data structure is used to wrap/represent the data of each system.

The structures/behaviors of the systems in the SoS are created/simulated by DEVSJAVA atomic or coupled models. There are five atomic models in this scenario: BRobot (for base robot), SRobot (for swarm robots), Sensor (for stationary sensors), Threat (for threat), and Plot Module (this is for plotting the system components during the simulation). Figure 4.6 illustrates the DEVSJAVA development environment, all atomic models, and other components.

Listing 4.3 XmlEntity Data Structure in DEVSJAVA [17]

```
public class XmlEntity extends entity{
    Vector value;
    String name;
    /** Creates a new instance of XmlEntity */
    public XmlEntity() {
    }
```

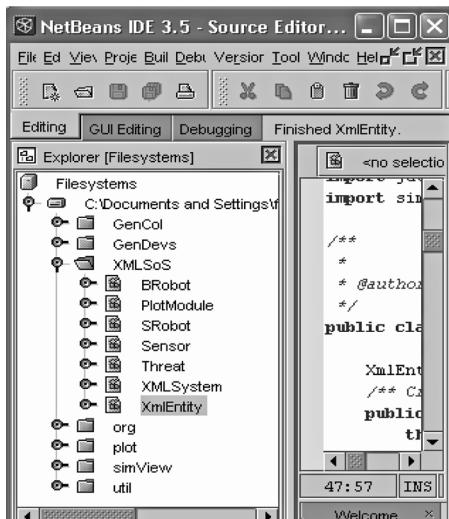


Figure 4.6 Atomic models and DEVSJAVA components.

The simulation framework is developed as a JAVA package, called XML-SoS. Under this package, we have atomic models and two more system components: XmlEntity and XmlSystem. XmlSystem is where the system components and their couplings are created or instantiated.

4.4.1.3 Simulation results

The structures/behaviors of the systems in the SoS are created/simulated by DEVSJAVA atomic or coupled models. Figure 4.7 is a screen shot of the DEVS model of the system of systems described above. Figure 4.8 shows a DEVS simulation step with the XML messages sent among the systems in the system of systems. Finally, Figure 4.9 shows the simulation environment created by the “Plot Module” atomic model in DEVS. In the environment, the two sensors are located next to each other, representing a border. The “threat” (dark gray dot) is moving in the area. When the threat is in one of the sensor’s coverage area, the sensor signals the base robot. Then, base robot signals the swarm robot so that it can go and verify whether the threat is real or not. The behavior of the system-of-systems components can also be seen in Figure 4.8 as the movements of the “threat” and the “swarm robot” are captured. The light gray dot represents the swarm robot. The base robot is not shown, since it does not move. When the threat enters into a sensor area, that sensor area filled with dark gray color to show the threat level. Then the swarm robot (light gray dot) moves into the sensor to check whether the threat is real. When the swarm robot reaches the sensor, it reports the threat

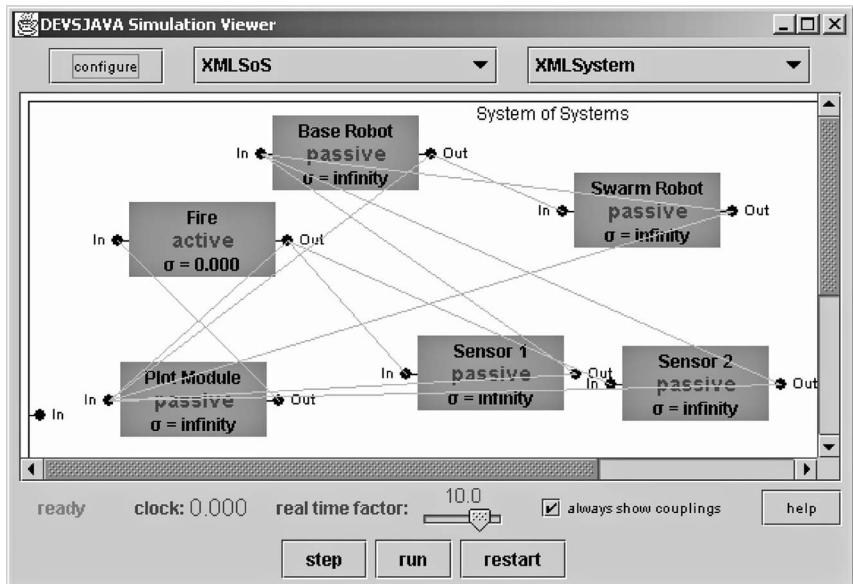


Figure 4.7 The DEVSJAVA atomic and coupled modules for XML base SoS simulation [4].

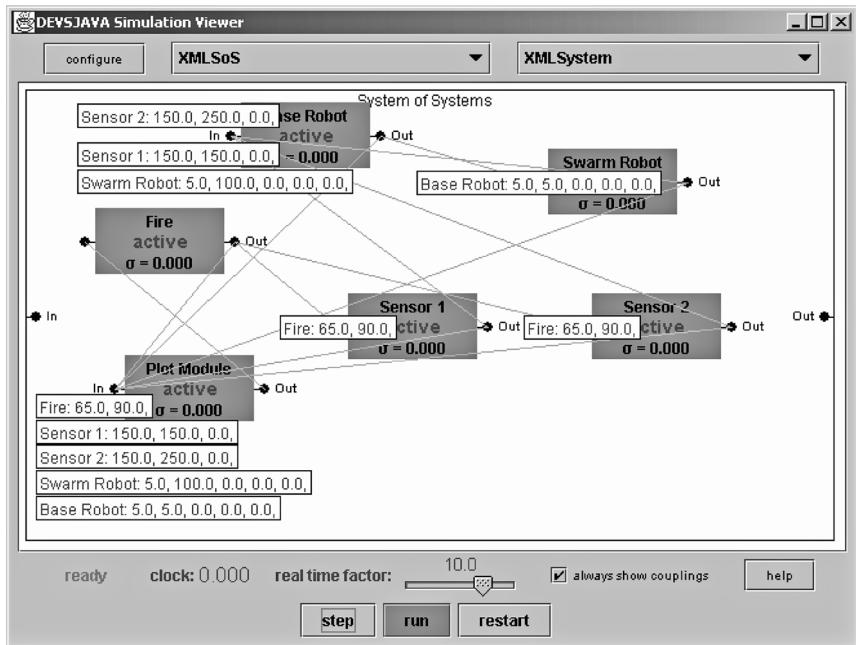


Figure 4.8 The DEVSJAVA simulation with XML-based messages shown at the destination [4].

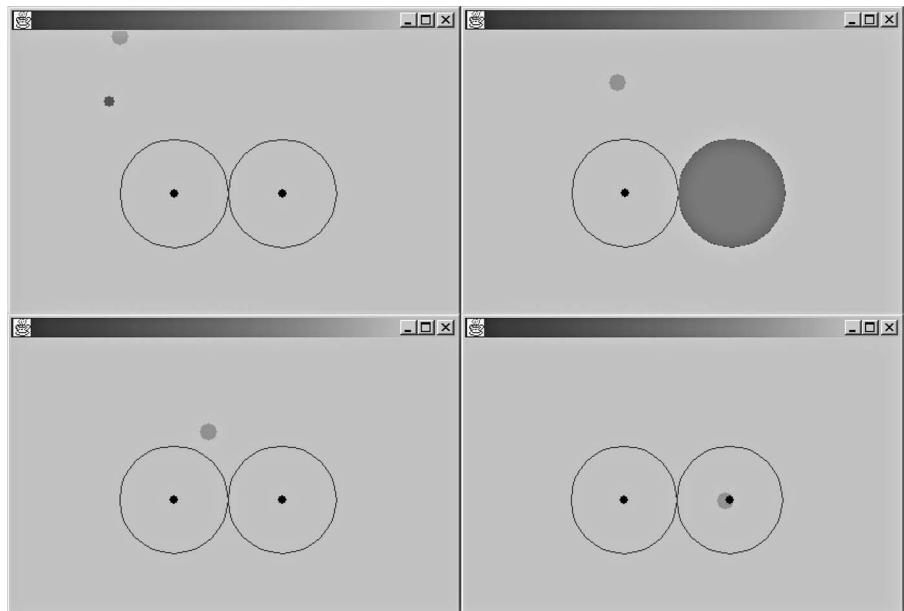


Figure 4.9 The progress of the DEVSJAVA simulation on data aggregation [4].

level to the base robot. If the threat is not real, the swarm robot moves away from the sensor's coverage area.

4.4.2 Case study 2: a robust threat detection system simulation

In this case, we have extended the simulation framework such that multiple swarm robots and several sensor nodes can be simulated for a robust threat detection scenario. In the robust threat detection (data aggregation scenario), there are one base robot, two swarm robots, five sensors, and one threat. The sensors are stationed such that they represent a sensor network covering an area. The threat is moving in the area and is detected by sensors when the threat is sufficiently close to a sensor node. All sensors communicate with a base robot, which does not actively seek threat. Instead, the base robot processes the information using sensor data and data sent by swarm robots. A command center can collect processed data (aggregated data) from the base robot without interacting with other components of the SoS such as sensors and swarm robots.

The robust threat detection works as follows. When a sensor detects the threat, they notify the base robot. Then the base robot sends the swarm robots the location of the threat based on the information sent by the sensors. Swarm robots are assumed to be equipped with the same sensors as the stationary sensors, so that they can verify the threat when they reach the sensor area. This is crucial in order to get rid of false detections as they can cause inefficient usage of system components. For example, if a sensor sends false threat information to the base robot, the command center will automatically send systems which are designed to disable the threat without verifying the threat. This will cause spending valuable system resources and can make the system more vulnerable to real threats. Thus, the verification of the threat by deploying several small swarm robots can help reduce the false threat detections and save system resources. Thus, when a threat is detected, the swarm robots are notified by the base robot. Then, swarm robots move toward the threat area and communicate among each other, so that they do not go to the same threat location.

4.4.2.1 XML format

As mentioned before, the hardware and/or software architectures of these systems will not be the same in reality. Thus, they may not be able to talk to each other successfully even though they can operate independently. We have implemented XML-based SoS message architecture in DEVSJAVA software [50]. In this XML-based message architecture, each system has an XML-like message consisting of their name and a data vector. The name of each system represents an XML tag. The data vectors are used to hold the data of the systems. The length of the data vectors in each system can be different based on the amount of data each system contains. For example, the XML message of the sensors has the sensor coordinates and the threat level. The

Table 4.2 The XML Components for the Systems in the SoS

System	Name	Vector Data Length
Base robot	“Base Robot”	5 ($X, Y, Threat, Xt, Yt$)
Swarm robot	“Swarm Robot 1”	6 ($X, Y, Threat, Xt, Yt, Threat V$)
Swarm robot	“Swarm Robot 2”	6 ($X, Y, Threat, Xt, Yt, Threat V$)
Sensor	“Sensor 1”	3 ($X, Y, Threat$)
Sensor	“Sensor 2”	3 ($X, Y, Threat$)
Sensor	“Sensor 3”	3 ($X, Y, Threat$)
Sensor	“Sensor 4”	3 ($X, Y, Threat$)
Sensor	“Sensor 5”	3 ($X, Y, Threat$)
Threat	“Fire”	2 (X, Y)

threat level is set when a threat gets in the coverage area of the sensors. On the other hand, the base robot’s XML message has its coordinates, the threat level, and the coordinates of the sensor who is reporting a threat. Thus, the data vector of base robot’s XML message has five elements, whereas the data vector of an XML message of a sensor has three elements. Table 4.2 presents the names and the length of the vector data of each system in the system of systems.

The data vectors are made of “double” variables in order to keep track of the positions accurately. The “*Threat*” element in the data vector is a flag representing threat (1.0) or no threat (0.0). The elements *Xt* and *Yt* are used as the destination (target) coordinates in the XML messages of the base robot and swarm robot. These represent the coordinates of the sensor who reported a threat. The “*Threat V*” element in the data vector is a flag representing whether or not a swarm robot verified a threat.

4.4.2.2 Simulation setup

[Figure 4.10](#) is a screen shot of the DEVS model of the system of systems described above. [Figure 4.11](#) shows a DEVS simulation step with the XML messages sent among the systems in the system of systems. In order to evaluate the performance of the robust threat detection, we have plotted the locations of base robot, swarm robots, sensors, and the threat. The sensor coverage areas are represented by a circle in order to illustrate complete sensor network coverage area. The threat is represented as a dark gray dot. The swarm robots are represented as medium gray (Swarm Robot 1) and light gray (Swarm Robot 2) dots. In addition, the sensor locations are presented as black dots. When a threat enters into a sensor area, the sensor location becomes dark gray, meaning threat detection. The black dot at the upper left corner represents the base robot, and it does not move during the simulation. [Figure 4.12](#) shows the initial positions of all the SoS elements.

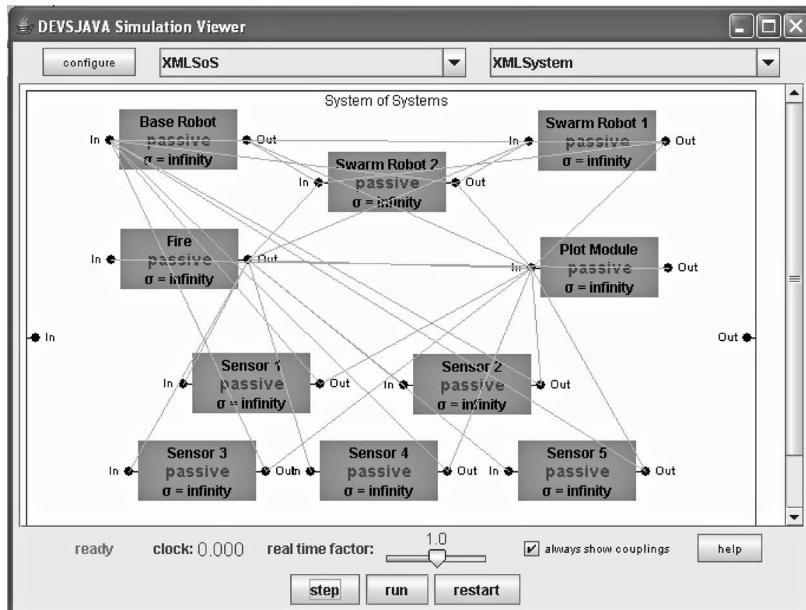


Figure 4.10 The DEVSJAVA atomic and coupled modules for XML base SoS simulation.

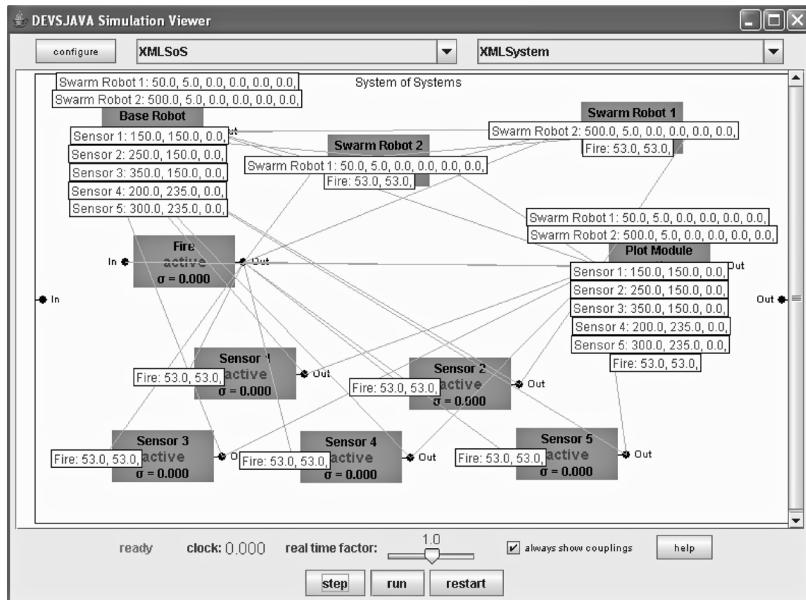


Figure 4.11 The DEVSJAVA simulation with XML-based messages shown.

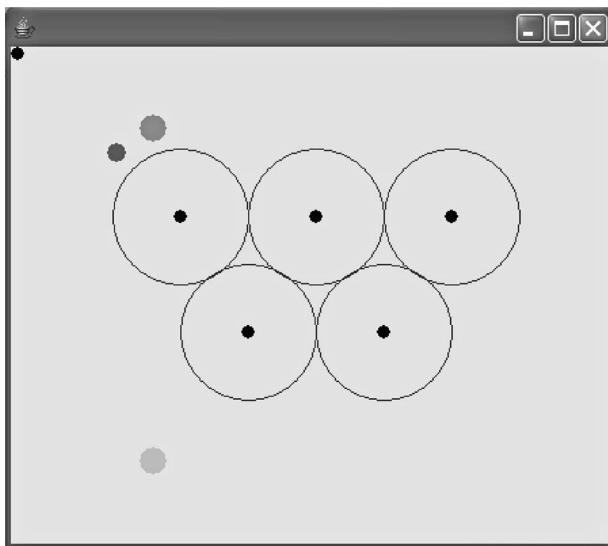


Figure 4.12 Initial conditions of XML system simulation.

Swarm robots are stationed on the top and bottom of the sensor field in order to have faster threat verification. A threat is generally verified by the closest swarm robot at the time of the threat detection. If the swarm robots are far away from each other, they both may go toward a threat location. However, when they are close enough they can communicate, and the second swarm robot generally goes back to its original location or tries to verify another threat. All five sensors can be seen in the figure as well. Sensors 1–3 are on the top row from left to right, respectively. Sensor 4 and Sensor 5 are on the bottom row of the cluster, from left to right as well.

4.4.2.3 Robust threat detection simulation

In this section, we will present screenshots of the simulation in order to illustrate the behaviors of the robust threat detection system of systems. We set the threat “Fire” moving randomly in the field in order to capture the behaviors of the swarm robots and sensors. Figure 4.13 shows the threat moving within the range of Sensor 1. Sensor 1 shows its activation by changing its center from dark gray to black. As soon as a threat is detected, a message is sent to the base robot that has sensors X and Y coordinates of the sensor and the threat flag. This flag is normally “0,” and it becomes “1” when a threat is detected. Once the base robot receives an XML message from Sensor 1, it checks the threat flag and sends the coordinates of Sensor 1 to Swarm Robot 1 and Swarm Robot 2 if the flag is set to “1”. The swarm robots receive the XML message containing the threat location (X_t, Y_t) and the threat level. When a swarm robot receives an XML message from the base robot, it checks the threat level and moves toward the threat destination if the threat flag is

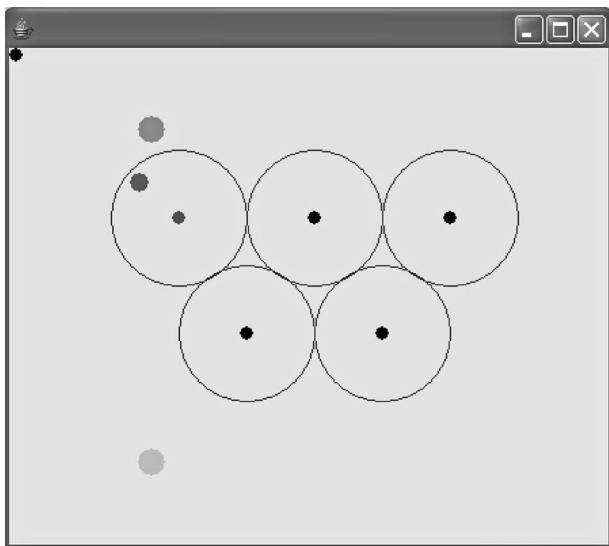


Figure 4.13 Sensor 1 detects a threat.

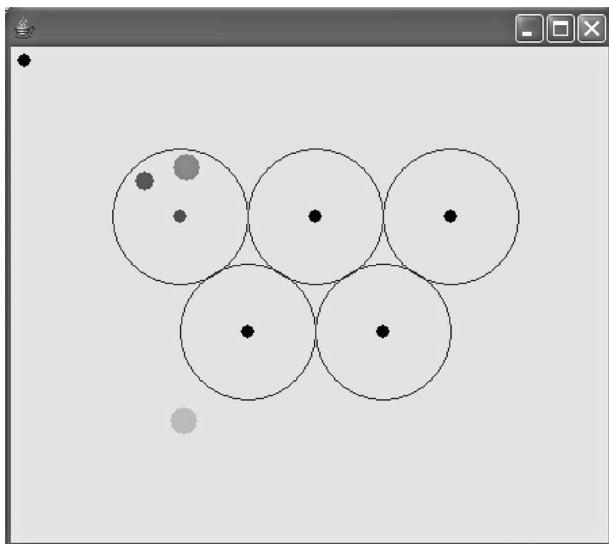


Figure 4.14 Swarm robots move to verify threat.

on. In Figure 4.14, Sensor 1 shows up as dark gray, and swarm robots start to move toward Sensor 1.

In Figure 4.14, we can still see that the threat is within Sensor 1's range, and that the swarm robots have moved to verify the threat's presence. Swarm Robot 1 has reached the sensor first and has verified the threat's presence

using its own sensor. It will remain at Sensor 1 for as long as the threat is within the range of Sensor 1. Swarm Robot 2 is still moving toward Sensor 1, because the two swarm robots are not yet within communication range of one another, and it does not know that Sensor 1 has reached the active sensor already. In Figure 4.15, the threat has moved out of every sensor's range, and no threat is detected by the Swarm Robot 1.

Once a swarm robot verifies that there is no threat, it sets "Threat V" flag to "0" and sends its XML data to base robot. This is the major reason for the robustness of the threat detection as false alarms are handled properly. Once the threat is out of the sensor area, all five sensors report to the base robot saying all is clear, and the swarm robots are called off. After being called off (having their "Threat V" variables set to "0" and their X_t and Y_t values set to the neutral location), the swarm robots travel to their respective positions and await another threat.

In [Figure 4.16](#), the threat has moved into the range of Sensor 5, causing it to activate and signal the base robot. The base robot again signals the swarm robots, and they move in to verify the threat's location, much like in [Figure 4.13](#).

Figure 4.16 differs from Figure 4.13, since in this scenario both swarm robots are within communication range of each other. Thus, the swarm robots will exchange their target locations (threat locations). Each swarm robot checks other swarm robot's location compared to the threat target. If the swarm robot finds the other swarm robot's position closer to the threat destination, it decides to move away. It would go to its neutral position if

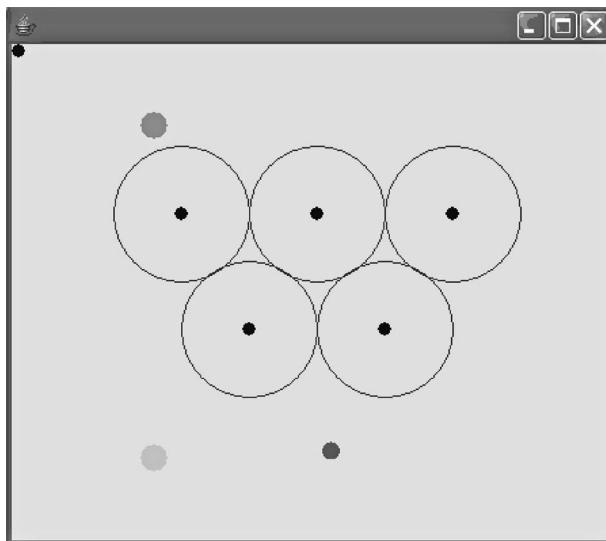


Figure 4.15 Swarm robots move to neutral location.

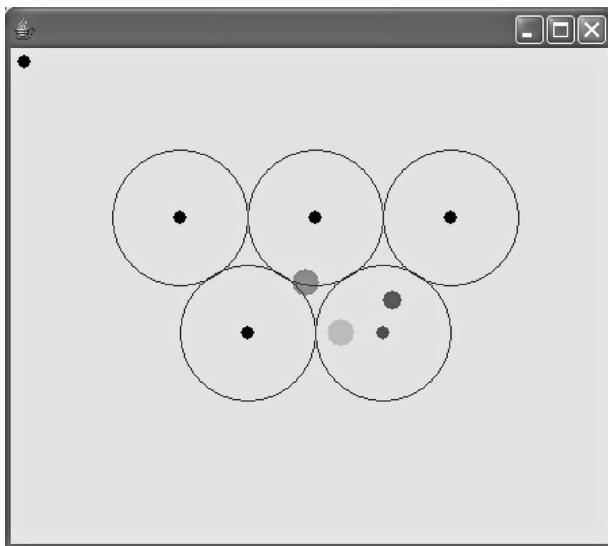


Figure 4.16 Swarm robots move toward Sensor 5.

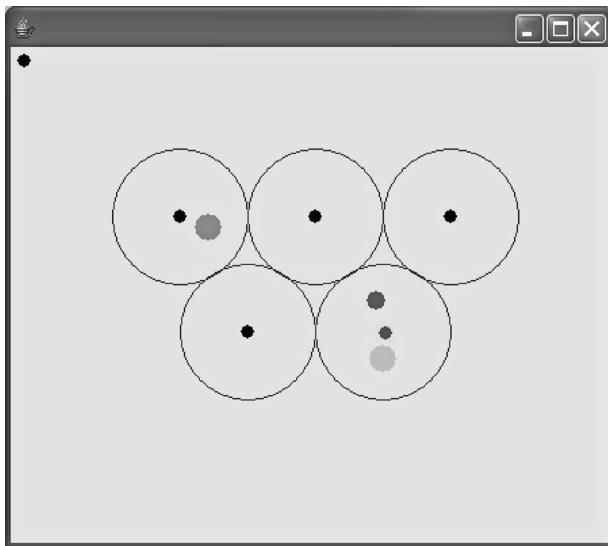


Figure 4.17 Swarm Robot 1 breaks off verification run.

there is not another threat in a different location. Thus, two swarm robots will never try to verify the same threat.

Figure 4.17 shows Swarm Robot 1, after communicating with Swarm Robot 2, breaking away from verifying the threat at Sensor 5 because it determined that Swarm Robot 1 was already closer to the target. Swarm Robot 2 will now

continue to verify the threat at Sensor 5, and Swarm Robot 1 will travel to a neutral location until it is called again by the base robot.

4.5 Conclusion

In this chapter, we have presented an extension to an XML-based SoS simulation framework and discussed two simulation case studies of robust threat detection and data aggregation with multiple autonomous systems working collaboratively as an SoS. Through multiple swarm robots in the field, the simulation results showed that the false alarms can be avoided by verifying the threats with swarm robots. This would increase the robustness of the data aggregation and lead to the efficient usage of the system resources. While DEVS formalism helps to represent the structure of an SoS, the XML provides a way to represent the data generated by each system. Together, DEVS formalism and XML form a powerful tool for simulating any given SoS architecture. In the future, we plan to extend the XML data representation and make it more generic and dynamic, so that when a new system is added to an SoS simulation, it will generate its XML message automatically and send it to other components of the SoS.

References

1. Crossley, W. A. 2004. System-of-systems: an introduction of Purdue University Schools of Engineering's Signature Area. *Engineering Systems Symposium*, March 29–31 2004, Tang Center–Wong Auditorium, MIT.
2. Lopez, D. 2006. Lessons learned from the front lines of the aerospace. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, CA.
3. Department of Defense. 1992. MIL-STD-499B, *Military Standard, Systems Engineering*, Draft, Department of Defense.
4. Sahin, F., Jamshidi, M., and Sridhar, P. 2007. A discrete event XML based simulation framework for System of Systems Architectures. *Proceedings of the IEEE International Conference on System of Systems*, April 2007.
5. Lopez, D. 2006. Lessons learned from the front lines of the aerospace. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, CA.
6. Wojcik, L. A. and Hoffman, K. C. 2006. Systems of systems engineering in the enterprise context: a unifying framework for dynamics. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, CA.
7. Azarnoush, H., Horan, B., Sridhar, P., Madni, A. M., and Jamshidi, M. 2006. Towards optimization of a real-world robotic-sensor system of systems. *Proceedings of World Automation Congress (WAC) 2006*, July 24–26, Budapest, Hungary.
8. Abel, A. and Sukkarieh, S. 2006. The coordination of multiple autonomous systems using information theoretic political science voting models. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, CA.
9. DiMario, M., J. 2006. System of systems interoperability types and characteristics in joint command and control. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, CA.
10. Zeigler, B. P., Kim, T. G., and Praehofer, H. 2000. *Theory of Modeling and Simulation*. Academic Press, New York.

11. Zeigler, B. P., Fulton, D., Hammonds, P., and Nutaro, J. 2005. Framework for M&S-based system development and testing in a net-centric environment. *ITEA Journal of Test and Evaluation* 26(3):21–34.
12. Mittal, S. 2006. Extending DoDAF to allow DEVS-based modeling and simulation. Special issue on DoDAF, *Journal of Defense Modeling and Simulation JDMS*, Vol 3, No. 2.
13. Mittal, S. 2007. DUNIP: A prototype demonstration. <http://www.acims.arizona.edu/dunip/dunip.avi>.
14. Mittal, S., Risco-Martin, J. L., and Zeigler, B. P. 2007. DEVSML: Automating DEVS simulation over SOA using transparent simulators, DEVS Symposium.
15. Mittal, S., Risco-Martin, J. L., Zeigler, B. P. 2007. DEVS-Based Web services for net-centric T&E. Summer Computer Simulation Conference.
16. Mittal, S. 2007. DEVS Unified Process for Integrated Development and Testing of Service Oriented Architectures. Ph. D. dissertation, University of Arizona.
17. Parisi, C., Sahin, F., and Jamshidi, M. 2008. A discrete event XML based system of systems simulation for robust threat detection and integration. *IEEE International Conference on System of Systems*, Monterey, CA, June 2008 (submitted).
18. Sahin, F., Sridhar, P., Horan, B., Raghavan, V., and Jamshidi, M. 2007. System of systems approach to threat detection and integration of heterogeneous independently operable systems. *Proceedings of IEEE Systems, Man, and Cybernetics Conference (SMC 2007)*, Montreal.
19. Jamshidi M. (Ed.). 2008. *System of Systems—Innovations for the 21st Century*. Wiley and Sons, New York.
20. Jamshidi M. (Ed.). 2008. *System of Systems Engineering*, CRC Press, Boca Raton, FL.
21. Meilich, A. 2006. System of systems (SoS) engineering & architecture challenges in a net centric environment. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, CA.
22. Abbott, R. 2006. Open at the top; open at the bottom; and continually (but slowly) evolving. *Proc. of IEEE International Conference on System of Systems Engineering*, Los Angeles, CA.
23. Jamshidi, M. 2005. Theme of the IEEE SMC 2005. Waikoloa, HA. <http://ieeesmc2005.unm.edu/>.
24. Sage, A. P. and C. D. Cuppan. 2001. On the systems engineering and management of systems of systems and federations of systems. *Information, Knowledge, Systems Management* 2(4):325–334.
25. Kotov, V. 1997. Systems of systems as communicating structures. Hewlett Packard Computer Systems Laboratory Paper HPL-97-124, pp. 1–15.
26. Carlock, P. G. and R. E. Fenton. 2001. System of systems (SoS) enterprise systems for information-intensive organizations. *Systems Engineering* 4(4):242–261.
27. Pei, R. S. 2000. Systems of Systems integration (SoSI)—a smart way of acquiring Army C4I2WS systems. *Proceedings of the Summer Computer Simulation Conference*, pp. 134–139.
28. Luskasik, S. J. 1998. Systems, systems of systems, and the education of engineers. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*. 12(1):11–60.
29. Manthorpe, W. H. 1996. The emerging joint system of systems: a systems engineering challenge and opportunity for APL. *John Hopkins APL Technical Digest* 17(3):305–310.
30. Jamshidi, M. 1983. *Large-Scale Systems—Modeling and Control*. North-Holland Publishing Company, New York.

31. Jamshidi, M. 1997. Large-Scale Systems—Modeling, Control and Fuzzy Logic. Prentice Hall, Saddle River, NJ.
32. Jamshidi, M. 2006. Class notes on System of Systems Engineering Course. University of Texas, San Antonio, TX, Spring, 2006.
33. Blanchard, B., and Fabrycky, W. 1998. Systems Engineering and Analysis, 3rd ed., Prentice-Hall, Saddle River, NJ.
34. Checkland, P. 1981. *Systems Thinking, Systems Practice*, 1st ed. John Wiley, Chichester.
35. Checkland, P. 1999. *Systems Thinking, Systems Practice*, 2nd ed. John Wiley, Chichester.
36. European Cooperation for Space Standardization. 1996. ECSS-E-10-01, System Engineering Process, European Cooperation for Space Standardization.
37. Electronic Industries Alliance. 1998. EIA/IS 632, Systems Engineering, EIA (1994). EIA/IS 731.1, Systems Engineering Capability Model, EIA.
38. Grady, J. O. 2000. *Systems Engineering Deployment*. CRC Press, Boca Raton, FL.
39. IEEE. 1994. IEEE P1220, Standard for Application and Management of the Systems Engineering Process, IEEE.
40. Martin, J. N. 1997. *Systems Engineering Guidebook*. CRC Press, Boca Raton, FL.
41. Department of Defense. 1992. MIL-STD-499B, Military Standard, Systems Engineering, Draft, Department of Defense.
42. Rechtin, E. and Maier, M. 2000. *The Art of Systems Architecting*, 2nd ed. CRC Press, Boca Raton, FL.
43. Sage, A. P. 1992. Systems Engineering, John Wiley and Sons, New York.
44. Keating, C., Rogers, R. Unal, R., Dryer, D., Sousa-Poza, A., Safford, R., Peterson, W., and Rabadi, G. 2003. System of systems engineering. *Engineering Management Journal* 15(3):36–45.
45. Mittal, S., Zeiglar, B. P., Sahin, F. and Jamshidi, M. 2008. Modeling and simulation for systems of systems engineering. In *System of Systems—Innovations for the 21st Century*, Jamshidi, M. (Ed.), Wiley and Sons, New York.
46. Matlab Simulink simulation and model-based design. <http://www.mathworks.com/products/simulink/>.
47. OMNET++ community site. <http://www.omnetpp.org/>.
48. The network simulator—NS-2. <http://www.isi.edu/nsnam/ns/>.
49. Defense Modeling and Simulation Office. 2008. High level architecture. U.S. Department of Defense. <https://www.dmso.mil/public/transition/hla/>.
50. Zeiglar, B. P. and Sarjoughian, H. [2000] Introduction to DEVS modeling and simulation with JAVA—a simplified approach to HLA-compliant distributed simulations, ACIMS—Arizona Center of Integrative Modeling and Simulation. <http://www.acims.arizona.edu>.
51. Sridhar, P. and Jamshidi, M. 2004. Discrete event modeling and simulation: V-Lab. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*.
52. Sridhar, P., Sheikh-Bahaei, S., Xia, S., and Jamshidi, M. 2003. Multi-agent simulation using discrete event and soft-computing methodologies. *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*.

chapter five

Technology evaluation for system of systems

Patrick T. Biltgen

Contents

5.1	Introduction.....	133
5.1.1	The challenge of systems of systems	136
5.1.2	From performance to effectiveness.....	137
5.2	Using modeling and simulation to quantify technology potential.....	138
5.2.1	Introduction to modeling and simulation	138
5.2.2	Using an integrated, hierarchical, modeling and simulation environment	139
5.3	Using surrogate models to accelerate technology evaluation studies....	142
5.3.1	Introduction to the concept of surrogate models.....	142
5.3.2	Response surface methodology.....	143
5.3.3	Artificial neural networks.....	144
5.3.4	Using a design of experiments (DoE) to generate data for a surrogate model.....	148
5.3.5	Process for executing the DoE and generating surrogates.....	149
5.4	Performing technology trade studies with surrogate models.....	150
5.5	Summary	157
	References	158

5.1 Introduction

Technology, from the Greek *techne* meaning “craft” and *logia* meaning “ordering,” is “the knowledge of the manipulation of nature for human purposes” [1]. Technology is so critical to the development of civilizations that archaeologists and anthropologists divide human prehistory into epochs based on the technology of the period. Since the end of World War II, exploitation of advanced technology has been the cornerstone of national power, in both the civil and military domains, and the development of new technology is an ongoing imperative driven by market forces and national needs.

Technology evaluation is defined as the assessment of the relative benefit of a proposed technology for a particular purpose with respect to one or more metrics [2]. The goal of technology evaluation is to rapidly and efficiently identify high-payoff technologies and establish a required level of resources to mature technologies to the point of implementation. In practice, the complexity introduced by the myriad interactions in the systems-of-systems problem space makes it difficult to precisely identify these technologies. Since the uncertainties related to the operational environment, assumptions, technology readiness, system interactions, and human interaction with constituent systems cannot easily be captured at the concept exploration phase of design, it may be sufficient to enumerate technology sensitivities and eliminate regions which exhibit little payoff using first-order methods. Essentially, the technology evaluation methods discussed herein are most appropriate to bound the space of feasible technologies and guide further analysis efforts as opposed to revealing a 1-to-*n* list of candidate technologies that enhance value with 100% certainty.

Over the past twenty years, many methods for technology evaluation have been proposed, and some are valid for application in the systems-of-systems domain. The least elegant, but arguably the most effective, method for evaluating technologies is physical experimentation. Epitomized by the X-series of prototype aircraft, the experimental approach is used heavily in the electronics and software industry, where candidate technologies and new features are incorporated into rapidly refreshed products. If accepted by the marketplace, their use becomes widespread. A typical example is the ubiquitous camera phone, which began life in Japan as the J-Phone in 2000. Luce refers to the rapid spread of technology into society by this means as *technology diffusion* [3].

The United States Air Force (USAF) and other organizations use a panel of expert scientists, engineers, and senior leaders called the Scientific Advisory Board (SAB) to formulate a long-term plan for technology utilization using a committee approach. Originally instituted in 1944 and led by Theodore von Karman, the original purpose of the SAB was to examine advances in basic science and analyze how these discoveries may affect the employment of airpower. An SAB study commissioned in 1994 directed the SAB to identify “technologies that will guarantee the air and space superiority of the United States in the 21st Century” [4]. The fifteen-volume, 2000-page document recommends technologies that support a number of Air Force missions; however, the study is tightly focused on specific vehicles, qualitative information, brainstorming, and anecdotal evidence. The vehicle-centric nature of the recent studies makes it difficult to assess the overall effectiveness of proposed technologies *in a system-of-systems context*. According to historian Michael Gorn, more recent studies have relied less on outside input from scientific leaders and have been very tightly focused on specific vehicles, qualitative information, brainstorming, and anecdotal evidence [5].

The Technology Development Approach (TDA), developed by Dr. Donald Dix, is a *qualitative* method for identifying expected technology impacts at the system-of-systems level. The TDA examines several technology efforts and objectives and proposes point-estimates for the key performance parameters (KPPs) for each technology. These technologies are then rolled up into the subarea goals for the proposed system and extrapolated to expected improvements in measures of effectiveness (MoEs). The TDA is constructed using expert opinion, brainstorming, and qualitative analysis [6].

Technology Identification, Evaluation, and Selection (TIES), developed by Kirby and Mavris, is a “comprehensive and structured method to allow for the design of complex systems which result in high quality and competitive cost to meet future, aggressive customer requirements” [7]. This technique uses physics-based modeling to quantitatively assess the impact of technologies by representing the technology KPPs as “k-factors,” which are essentially scale factors on discipline-level metrics. While TIES can be seen as a quantitative extension of the TDA approach, traditional applications of the method have been primarily focused on the evaluation of performance for a system and have to date not addressed the issue technology evaluation for large-scale systems of systems [7].

Currently, the Air Force Research Laboratory (AFRL) is actively engaged in a research effort to “integrate new methodologies and tools with existing ‘industry-standard’ tools to effectively test the effects of new technologies on air vehicle capability” [8]. An AFRL program, Quantitative Technology Assessment (QTA), an extension of TIES to the capability level, is enabled through constructive simulation and parametric modeling [9]. This technique, well suited for system-of-system studies and evaluation of technologies with respect to capability-level MoEs, serves as a model of a superior process for technology evaluation.

Although many processes for technology forecasting have been proposed, a formalized process for technology evaluation that provides a rigorous, justifiable, and traceable approach to decision making is needed. Some desired attributes of a process for technology evaluation include the following:

- Quantitative: Compare solutions based on measurable quantities; limit influence of subjective judgment and bias.
- Traceable: Identify the source of benefit and the relative sensitivities to changes.
- Flexible: Generalizes to multiple problems and easily adds new variables and dimensions.
- Reusable: Analysis environment can be used to study multiple attributes of the same problem or to study the same problem in a different context.
- Agile: Can be applied in a reasonable time frame; decisions can be made before the technologies under consideration are obsolete.
- Parametric: Avoids point solutions and provides visibility into behaviors previously obscured by the complexity of the problem.

- Affordable: Produces valid results without extensive manpower commitments and uses commercial off-the-shelf tools when possible.
- Simple: To the highest degree possible, the process must be logical and teachable to enable widespread adoption.

The aforementioned attributes are generally desirable in any physics-based multidisciplinary analysis; however, the challenge of technology evaluation is compounded by the increasing focus of the systems engineering process on large-scale heterogeneous systems of systems.

Technologies are the foundation of the systems-of-systems hierarchy shown in Figure 5.1. A process for technology evaluation ideally must not only examine the influence of proposed technologies from a bottom-up impact analysis standpoint but must also demonstrate applicability for a top-down decomposition of requirements to an unknown suite of potential technologies. This chapter discusses enabling techniques to examine technology infusion at various locations in the hierarchy shown in Figure 5.1 and summarizes approaches for both bottom-up and top-down technology evaluation for systems of systems.

5.1.1 *The challenge of systems of systems*

A system, from the Greek *sunistanai* meaning “to combine” is “a combination of interacting elements organized to achieve one or more stated purposes” [10]. In recent years, the term “system of systems” (SoS) has become increasingly

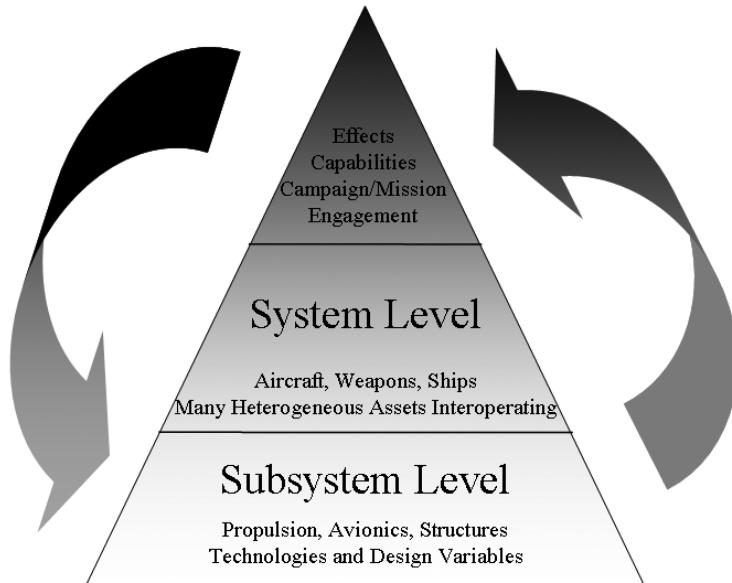


Figure 5.1 Systems-of-systems hierarchy [2].

popular terminology for a large-scale system that is comprised of a variety of heterogeneous, interoperable, collaborative systems [11]. While the precise origin of this term is unclear, a 1964 paper by Berry on New York City refers to “cities as systems within systems of cities” [12]. Additional definitions of the term abound in the literature:

- The Department of Defense defines a system of systems as “a set or arrangement of systems that are related or connected to provide a given capability” [13].
- The International Council on Systems Engineering (INCOSE) refers to the definition by Krygiel: “a system-of-systems is a set of different systems so connected or related as to produce results unachievable by the individual systems alone” [14].
- The Air Force Scientific Advisory Board defines a system of systems as “a configuration of systems in which component systems can be added/removed during use; each provides useful services in its own right; and each is managed for those services. Yet, together they exhibit a synergistic, transcendent capability” [15].

Maier has proposed five criteria for the identification of a system as a system of systems [16]; however, ten years after his publication, disagreements bristle regarding the classification of systems as systems of systems, families of systems, federations of systems, complex systems, complex adaptive systems, coalitions of systems, collaborative systems, interoperable systems, netcentric systems, supersystems, and others [17]. For the purposes of this chapter, the term “systems of systems” is used to define “a class of systems wherein a set of independent systems, each having unique behavior and performance, is organized to perform collaboratively and coherently to achieve a purpose” [18].

Evaluation of systems-of-systems effectiveness is compounded by a number of challenges, including the independence of constituent systems, the interdependence between constituent systems, the complexity of the SoS and its operation in an environment, the fuzzy boundaries between elements, and the lack of engineering models at the SoS level. Techniques for quantitative evaluation of technology potential for systems of systems must address these challenges, and many of these issues are topics for current research in the field.

5.1.2 *From performance to effectiveness*

In the study of systems of systems, it is necessary to highlight a shift from evaluation of performance to evaluation of effectiveness. Performance describes what a system does. Effectiveness describes what a system does in a relevant context or scenario. For example, if an aircraft can fly 1,000 nautical miles at 40,000 feet, this is a statement of its performance. The same aircraft

has different effectiveness at completing a mission if the 1,000-nautical-mile distance is over water or over a hostile country. Traditional systems engineering primarily focuses on the calculation of system performance. When operational context is considered, it is usually rare for multiple scenarios or operating conditions to be considered in the evaluation of a product or its constituent technologies, primarily due to the difficulty in establishing a relevant testing environment for the myriad conditions.

The term *measure of performance* (MoP) refers to a metric used to evaluate how well a system performs a task, while a *measure of effectiveness* (MoE) is “a criterion used to assess changes in system behavior, capability, or operational environment that is tied to measuring the attainment of an end state, achievement of an objective, or creation of an effect” [19]. MoPs are usually more appropriate for system-level evaluation, and MoEs are more relevant to systems of systems–level evaluation. While these terms are standard definitions used by the U.S. Department of Defense, their meaning is also apt for the description of commercial systems of systems.

Effective technology evaluation for systems of systems requires a method that quantifies the benefit of a proposed technology against one or more MoEs, taking into account the complexities of the SoS analysis problem. This often implies that one or more scenarios simulating the relevant operational conditions for the SoS must be developed.

5.2 *Using modeling and simulation to quantify technology potential*

5.2.1 *Introduction to modeling and simulation*

Many proven techniques for resource allocation are based on qualitative information and subjective analysis. For example, the Technology Development Approach (TDA) evaluates system of systems–level MoEs using a committee approach [6]. Unfortunately, a committee approach is only valid when the physics of the problem are well understood and seldom extends to the system-of-systems level where the complex interactions between heterogeneous elements do not always follow intuitive or predictable patterns. Also, since the experience base of subject matter experts is bounded by tacit information based on known situations and scenarios (essentially a partial derivative), an expert-driven process is often not appropriate to produce quantitative estimates of system effectiveness in uncertain operating conditions. To address these shortcomings, an approach based upon modeling and simulation is proposed.

Modeling, “a simplified description of a complex entity or process,” is literally the creation of a model or alternative representation [20]. Its complement is simulation, defined as “the process of imitating a real phenomenon with a set of mathematical formulas” [21]. Simulation can also be described as the repeated exercise of a model under various conditions. The use of

modeling and simulation in the aerospace community is not new and has coevolved dramatically with advancement in digital computers [22]. For instance, since the introduction of Integrated Product and Process Development (IPPD) in late 1993, Schrage has advocated a generic methodology that leverages a computer-integrated environment to enable robust design simulation [23]. This methodology “provides the means for conducting parallel process/product (cost/performance) design trades at various levels (system, component, part)” and enables “distributed design and development” [24].

Furthermore, the National Science Foundation, in its 2006 report on “Simulation-Based Engineering Science” notes that simulation “can be used to explore new theories and to design new experiments to test these theories” and “also provides a powerful alternative to the techniques of experimental science and observation when phenomena are not observable or when measurements are impractical or too expensive” [25]. Modeling and simulation are enabling techniques that provide a means to calculate MoEs for candidate technologies and system architecture and essentially act as a transfer function from technology performance to system-of-systems effectiveness.

5.2.2 *Using an integrated, hierarchical, modeling and simulation environment*

While an appropriate constructive simulation can be used to translate system-level MoPs to system of systems–level MoEs, it is first necessary to discuss the creation of a hierarchical modeling and simulation environment. For illustrative purposes, the example of a long-range bomber aircraft will be used. The modeling and simulation environment for this example can be divided into three levels, subsystem, system, and system of systems, as shown in [Figure 5.2](#). In this simplified example, the system level is typified by the design of the bomber aircraft and the weapon it fires. Note that, while it can be argued that the weapon is a subsystem of the aircraft, the exact enumeration of level names is immaterial for this example. Subsystems that contribute to both system designs include aerodynamics and propulsion, although other components such as sensors, flight controls, structures, and the like could also be included if such models are readily available. As shown in Figure 5.2, subsystem-level analyses such as aerodynamics and propulsion calculate quantities such as lift coefficients, drag coefficients, fuel consumption, and engine thrust that are used by the system-level analyses. Inputs to these analyses include component efficiencies, airfoil characteristics, pressure ratios, and material limits, which can be affected by technologies. While each of the boxes at the system and subsystem level are typical of *models*, the mission evaluation box is more appropriately termed a simulation. In the mission evaluation tool, an aircraft and weapon combination designed using lower-level models is assessed against a given threat laydown, in a certain geographic region, with particular concepts of operations (CONOPS), rules of engagement, and battle

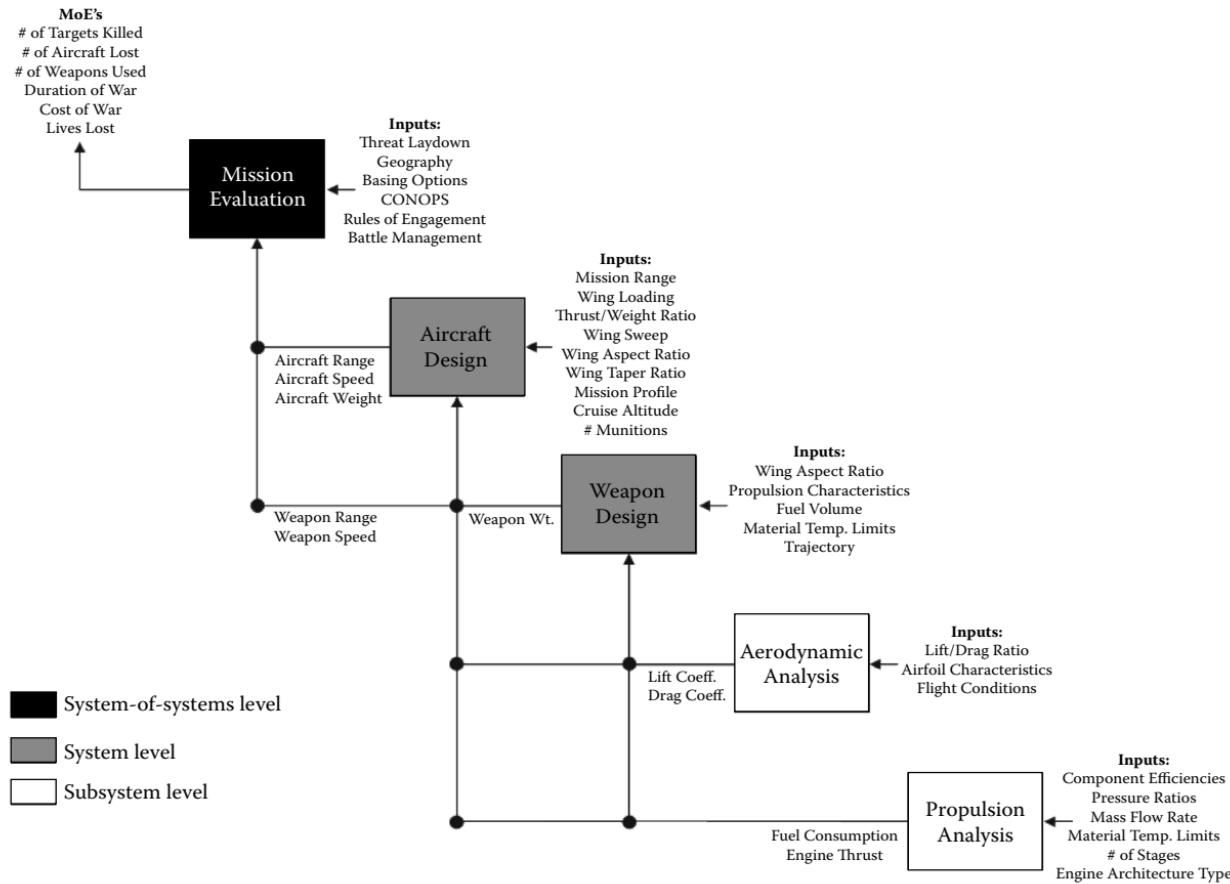


Figure 5.2 Example of a hierarchical modeling and simulation environment for aircraft design and effectiveness evaluation.

management doctrine. MoEs such as the number of targets killed, number of friendly aircraft lost, number of weapons fired, duration of the war, and the cost of the war are calculated as a function of system-level parameters such as range, speed, and number of weapons per aircraft.

To illustrate how performance and effectiveness differ in this environment, assume that a baseline aircraft configuration can be defined by specifying all the input parameters on the right side of each box in Figure 5.2. Using a different airfoil in the aerodynamic analysis that results in a lower drag coefficient at the cruise flight condition will impact the drag estimates used in the aircraft design box. When the aircraft synthesis and sizing code uses this modified value in its analysis, the aircraft range may increase, maximum speed may increase, and weight may decrease. This is an example of how a change in performance can be quantitatively evaluated. In an analysis of the results, an aircraft designer would summarily conclude that reduced drag produces a “better” aircraft; however, this does not mean that the aircraft is better in terms of effectiveness. Using the mission evaluation constructive simulation, a different analyst can assess the change in MoEs as a result of a faster, long-range aircraft. Depending on the scenario specified, the aircraft may or may not be effective; for example, supersonic aircraft generally require more maintenance than subsonic aircraft. A faster aircraft may prosecute more targets in a given amount of time, but the cost of the war may increase due to the operations and support of the high-speed aircraft. In summary, an optimal configuration at the system level is not necessarily optimal at the system-of-systems level. The same logic can be applied to the system/subsystem level. Extending a well-known paradigm from the multidisciplinary optimization community, the optimization of individual discipline-level performance parameters may result in a globally suboptimal solution.

A linked, hierarchical modeling and simulation environment can be created by linking the outputs at one hierarchical level to the inputs at another hierarchical level. Although this may seem straightforward, it was not until the 1990s that software tools for such linkage emerged, and Kroo refers to the connection of different multidisciplinary analyses as “first generation multidisciplinary optimization” techniques [26]. An example, shown in Figure 5.3, links the NASA Flight Optimization System (FLOPS), an aircraft sizing and synthesis code, to the FLAMES simulation framework. The FLAMES aircraft flight dynamics model requires input parameters that are generated by FLOPS such as aircraft range, speed, and weight. In practice, while there are many ways to link modeling and simulation tools, commercial integration

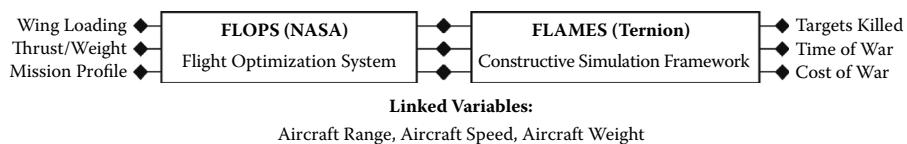


Figure 5.3 Model linking to translate MoPs to MoEs.

frameworks such as Engineous's iSIGHT, Phoenix Integration's ModelCenter®, and Technosoft's Tool Integration Environment have become increasingly adept at this task.

Creating a hierarchical modeling and simulation environment to develop a transfer function from atomic-level technology parameters to system of systems-level MoEs is the first step in enabling technology evaluation. With such an environment, quantitative trade studies can be performed one point at a time. In fact, even the linking of computerized tools across a simulation hierarchy speeds the analysis process and reduces the propensity of errors to develop in nonintegrated environments. The next step in performing technology evaluation studies is to use this linked hierarchical environment as an enabler to conduct large domain-spanning exploratory studies on technology potential.

5.3 Using surrogate models to accelerate technology evaluation studies

The previous section introduced the notion of a linked, hierarchical physics-based modeling and simulation environment. This environment, in and of itself, provides two attributes required for technology evaluation for systems of systems: quantification and traceability. Using proven modeling and simulation tools, the precise benefit of proposed technologies on top-level MoEs can be calculated based on the physics of the problem. Sensitivities of each MoE to the key performance parameters at lower levels can be used to trace impacts across one or more levels. Unfortunately, what the aforementioned environment lacks is speed; evaluation of a single case requires that all tools in the hierarchy be executed. This is at best a partially parallel process and, for tightly coupled problems, often a serial one with iteration and convergence loops.

One way to increase the speed of the analysis process is to decrease the fidelity of all the models in the hierarchy. This approach is common in the analysis of many scientific processes, for example, the assumption of a "billiard ball" model of collision in kinetic theory; however, since the complex interactions between the various systems and subsystems in the hierarchy may lead to complex emergent behaviors that may be lost through linearization and simplification, an alternate approach is desired. A technique that lends itself to high-speed, high-fidelity analysis is called *surrogate modeling*, which relies on a reduction in the degrees of design freedom as opposed to a reduction in model fidelity as a means to decrease run time.

5.3.1 Introduction to the concept of surrogate models

Surrogate models are an approximation technique for replacing existing analytical models with a suitable substitute. Surrogate models in the form

of response surface equations were first introduced by Box and Wilson in 1951 and developed extensively throughout the 1950s [27]. After several failed attempts in the 1970s and 1980s, the first successful widespread application of surrogate models in the aerospace community was initiated by Tai, Mavris, and Schrage in 1995 [28]. Over the last five years, the term “surrogate model” has gradually replaced “metamodel” in the literature, since the latter is often associated with Unified Modeling Language (UML) diagrams and other low-fidelity approximation of codes in the software engineering field.

The basic idea behind a surrogate model can be abstracted from a transfer function. A modeling and simulation tool is one type of transfer function from input variables to output variables. Through a series of predefined mathematical and physics-based relationships inside the tool, a set of input variables can be transformed into an appropriate set of output variables. The concept of the surrogate model addresses the question “is there an alternative transfer function that maps the same inputs to approximately the same outputs?” Since the exact relationship between responses and input variables may be difficult to define analytically and inconsequential to the creation of a surrogate mapping, a surrogate model is an empirically assumed model that approximates these relationships. These “models of models” can be highly accurate if appropriately created and form the basis of modern advanced design for their wide range of applicability. In practice, many types of surrogate models exist, including response surface equations, artificial neural networks, radial basis functions, Kriging models, Gaussian processes, and others. The first two of these are the most popular for systems-of-systems analysis and are discussed below [29–34].

5.3.2 Response surface methodology

One process by which surrogate models are created is called response surface methodology (RSM). RSM approximates the inherent dependence of functional responses (outputs) to a series of design variables (inputs) using a least-squares regression approach to the determination of unknown model coefficients. The resulting equation takes the shape of a multidimensional *surface*, leading to the term “response surface equation” or RSE. According to Myers and Montgomery, a second-order RSE based on a Taylor series expansion is a functional form that benefits from flexibility, ease of creation, and considerable practical experience that demonstrates its effectiveness [35]. The traditional form of a second-order RSE for response, R , coefficients, b , independent variables, x , and an error term ϵ is shown in Equation 5.1.

$$R = b_o + \sum_{i=1}^k b_i x_i + \sum_{i=1}^k b_{ii} x_i^2 + \sum_{i=1}^{k-1} \sum_{j=i+1}^k b_{ij} x_i x_j + \epsilon \quad (5.1)$$

Response surface equations have been used in a wide variety of system-modeling activities including propulsion systems [36,37], automobile components [38], power systems [39], commercial aircraft [40–43], unmanned vehicles [44], helicopters [45,46], tiltrotors [47], missiles [48–50], surface ships [51,52], network switches [53], and torpedo design [54,55]. RSEs have also found use in the design of systems of systems including the U.S. air transportation system [56–58] and military aircraft survivability [59,60]. Response surface equations can be easily created for a wide range of problems; however, they may perform poorly when nonlinear or discontinuous responses endemic to systems of systems must be modeled. For this reason, the mechanics behind response surface models will not be detailed in this chapter.

5.3.3 Artificial neural networks

An artificial neural network is an interconnected group of mathematical functions that is patterned on the connections between nerve cells. The fundamental idea in this approach is that the computational elements themselves are very simple, but like biological neurons in human brains, the connections between the neurons define very complex behaviors and computational abilities. The technique can trace its origin to a 1943 article by neurophysiologist Warren McCulloch and mathematician Walter Pitts entitled “A Logical Calculus of Ideas Immanent in Nervous Activity” [61]. As in biological systems, a single neuron can be connected to many other neurons to create very complex networks of structures. Artificial neural networks have found widespread application in pattern recognition and classification, control processes, speech recognition, optical character recognition, autonomous robots, and the development of adaptive software agents. Their ability to model processes also makes them ideal for regression tasks, especially those with discontinuous or highly nonlinear responses. Introductory works include references [62] and [63]. Although there are many types of neural networks including stochastic neural networks, recurrent networks, Hopfield networks, radial basis functions, instantaneously trained networks, cascading neural networks, and committees of machines, the most common type of neural network and the technique used with success in the modeling of systems of systems is a feedforward neural network. This type consists of several layers of interconnected neurons. Typically, three layers are used: the input layer, the hidden layer, and the output layer, in a construct called a multilayer perceptron as shown in [Figure 5.4](#).

As noted in the figure, a single response has a given number of inputs, X_m , and an unknown number of hidden nodes, H_m , whose optimum configuration is problem dependent. This number can be found iteratively or through numerical optimization. The power of neural networks comes from their ability to model nonlinear behaviors. This is often accomplished through the use

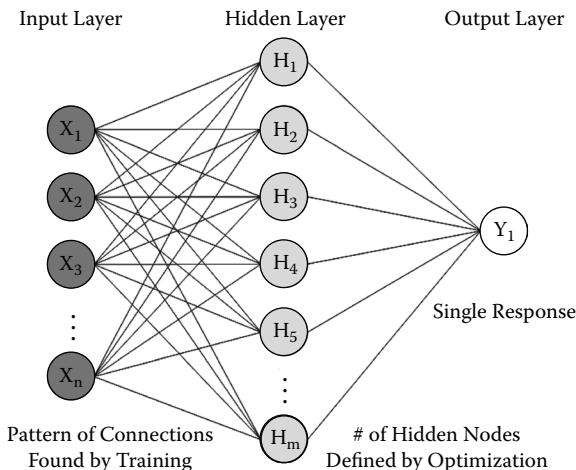


Figure 5.4 Structure of a feedforward neural network [2].

of a sigmoid function, a special case of the logistic curve, as the transfer function between the input layer and the hidden layer, shown in Equation 5.2.

$$S(z) = \frac{1}{1 + e^{-z}} \quad (5.2)$$

Also called the “squish” or “squash” function, the sigmoid reduces the neuron’s activation level to the range of [0,1] and stimulates the property of nonlinearity in the network. As an added benefit, the sigmoid function has a very simple derivative, which is needed to perform the backpropagation feature during the training process. Other relations such as a step function or hyperbolic tangent function can be used in place of the logistic sigmoid equation. To formulate a neural network equation, the value of each hidden node, H_j , is calculated using the sigmoid function where the parameter z in Equation 5.2 is replaced by a linear function of the input variables, X_i , as shown in Equation 5.3.

$$H_j = \frac{1}{1 + e^{-\left(a_j + \sum_{i=1}^N (b_{ij} X_i)\right)}} \quad (5.3)$$

Finally, the unified form of the neural network equation (Equation 5.4) can be written by encapsulating each hidden node within a linear function which transforms the numerical values from the hidden layer to the output layer. Note that one such equation is needed to calculate the value of each response, R_k .

$$R_k = e_k + \sum_{j=1}^{N_H} \left(f_{jk} \left(\frac{1}{1 + e^{-\left(a_j + \sum_{i=1}^N b_{ij} X_i \right)}} \right) \right) \quad (5.4)$$

where:

- a_j is the intercept term for the j th hidden node
- b_{ij} is the coefficient for the i th design variable
- X_i is the value of the i th design variable
- N is the number of input variables
- e_k is the intercept term for the k th response
- f_{jk} is the coefficient for the j th hidden node and k th response
- and N_H is the number of hidden nodes

Until recently, the development of robust neural network equations was considered more of an art than a science, and few computational tools existed to automate the training process. Several challenges led to a lack of process standardization for neural network creation. For instance, the topology of the neural network consisting of the number of layers and number of hidden nodes must be user specified, and the network must be “trained” through an iterative procedure for a user-specified time. Furthermore, the selection of the number of hidden nodes is often problem dependent and can be difficult without an optimizer: too few nodes incorrectly captures the behavior of the code, while too many leads to overfit problems [64].

To provide a more rigorous approach to neural network creation and validation, Johnson and Schutte developed the Basic Regression Analysis for Integrated Neural Networks (BRAINN) module, which combines optimization algorithms with the MATLAB neural network toolbox and a simple GUI [65]. Using this tool, data from the integrated simulation environment can be quickly analyzed and an accurate neural network equation produced. The training and optimization process used by BRAINN is shown in [Figure 5.5](#). Here, initial guesses for the unknown model coefficients a, b, c, d, e , and f are assumed by the training algorithm. These coefficients are adjusted according to the difference between the actual and estimated response using the backpropagation supervised learning technique [66]. The training process is encapsulated within an optimization algorithm that varies the number of hidden nodes, adjusting the topology of the network to minimize the error in the estimated response.

To use a neural network surrogate model in place of the actual simulation tool for high-fidelity high-speed analysis demands that the network accurately captures the behavior of the simulation tool. This is especially critical when the equation is used for parametric and probabilistic analysis.

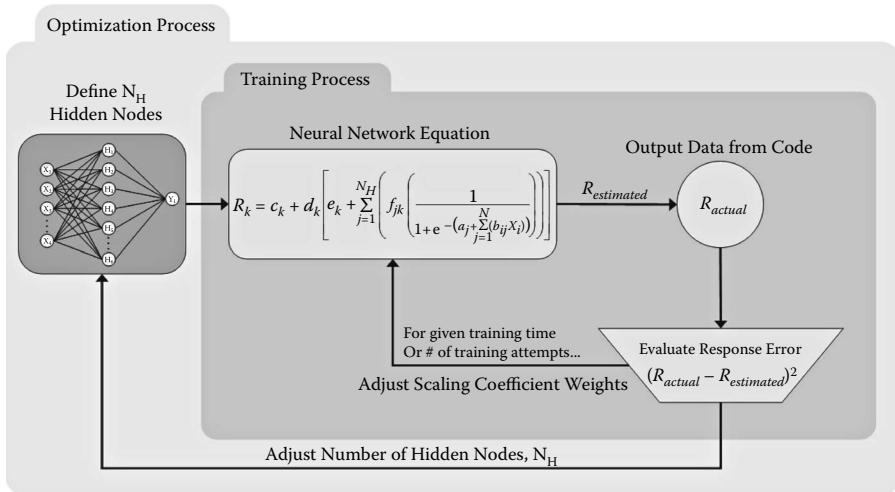


Figure 5.5 Procedure for neural network training [67].

Traditionally, simulation data is divided into two groups, a training set used to directly train the neural network and a smaller validation set used to assess the error of the model at off-design conditions. This technique provides a means to ensure that the equation is valid across the design space with maximum confidence and minimum overfit. In addition to the built-in checks in the BRAINN routine, the five-step goodness-of-fit procedure developed by Kirby and Barros should be used to ensure that the neural net accurately predicts points within the sampled design space [68]. Finally, it is important to note that, like all approximations, the neural network equation is not valid outside the ranges used to create it: neural networks can be used for interpolation, but never for extrapolation.

While neural networks have been used for many years in the design of control systems for aerospace applications, they have become increasingly popular for systems-of-systems studies and have been applied to the design of gas turbine propulsion systems [69], aerodynamics [70], air defense weaponry [71], the space shuttle main engine [72], spacecraft trajectory design [73], and a long-range strike system architecture [74].

Although computerized tools exist for the creation of neural networks, response surface models, and other surrogates, the selection of a surrogate type is merely another hypothesis in the analysis process. Neural networks have demonstrated effectiveness for systems-of-systems technology evaluation; however, simpler models can be used if they adequately capture the behaviors of the desired design space.

5.3.4 *Using a design of experiments (DoE) to generate data for a surrogate model*

The linked hierarchical suite of modeling and simulation tools provides a means to calculate MoE changes based on technology parameters, and the surrogate modeling technique enables rapid high-fidelity analysis. One remaining challenge is the need to generate data for surrogate creation in an efficient and repeatable manner. For this, the concept of “design of experiments” is introduced.

A design of experiments (DoE) is “a systematic, rigorous approach to engineering problem solving that applies principles and techniques at the data collection stage so as to ensure the generation of valid, defensible, and supportable engineering conclusions” [75]. This statistical technique is concerned with selecting experiments to be performed that generate the maximum amount of data with the minimal expenditure of time and money. The concept of a DoE originated in 1918 when the director of the Rothamsted Agricultural Experiment Station in the United Kingdom hired statistician Ronald A. Fisher to analyze historical records of crop yields. The station “had records extending over decades for crop yields from extensive plots of land each of which was treated with the same particular fertilizer” [76]. Additionally, they had records of temperature, rainfall, and other environmental factors over the same time period. This data, collected in a haphazard manner, did not answer some critical questions despite the analysis technique applied. Fisher invented the design of experiments to standardize the process by which data is collected for analysis [77]. Experimental design techniques have also been refined by Yule [78], Box and Hunter [76], Scheffé [79], Cox [80], and Taguchi [81].

While there are many different types of experimental designs with various benefits and deficiencies, space-filling designs, which literally fill an n -dimensional space, “should be used when there is little or no information about the underlying effects of factors on responses” and are “useful for modeling systems that are deterministic or near-deterministic” such as computer simulations [82]. While random points can be used to fill a space, an alternative scheme called “sphere-packing” is used to minimize the maximum distance between any two points in an n -dimensional space, akin to placing billiard balls into an n -dimensional box [83]. Mathematical techniques to assess this distance have been developed extensively in the literature [84–86]. According to Cioppa, “A good space-filling design is one in which the design points are scattered throughout the experimental region with minimal unsampled regions; that is, the voided regions are relatively small” [87]. As a result, space-filling designs can be effective for neural network models when the exact location of inflection points in the design space is unknown. Other variations on space-filling designs include Latin hypercube, uniform, minimum potential, maximum entropy, and integrated mean square optimal designs; however, unless the designer has familiarity with a specific type of experimental

design, a generic sphere-packing design is often sufficient [83]. Although it may seem that a neural network only needs a large number of random points, Biltgen demonstrated that neural network surrogate models for system-of-systems also require low independent variable correlation in the DoE for reasonable model fits [2]. In contrast to the hand-designed experiments that were prevalent through the 1970s, many computerized tools exist for the rapid creation of space-filling DoEs including several MATLAB® toolboxes, JMP® by the SAS Institute, and Design-Expert® by Stat-Ease.

5.3.5 Process for executing the DoE and generating surrogates

The process for integrating the aforementioned steps to generate surrogates usable for parametric technology evaluation for systems of systems is shown in Figure 5.6. Step 1 is to integrate the various simulation tools used to evaluate MoEs by mapping inputs to outputs, as illustrated in Figure 5.3, and identifying the ranges of important input parameters. Step 2 uses these ranges to create an appropriate DoE table to act as the run matrix for the integrated environment that is executed through the hierarchical modeling and simulation environment in Step 3 to develop a matrix of output data. The surrogation process in Step 4 uses the input DoE and the output data from the

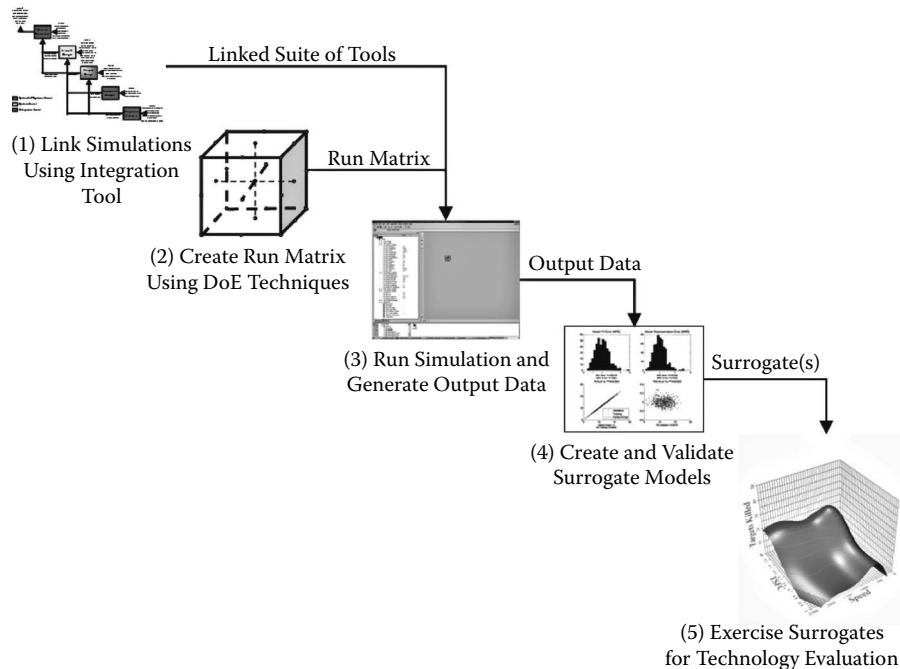


Figure 5.6 Process for generating surrogate models for parametric technology evaluation for systems of systems.

simulation to generate a suitable surrogate using one of the techniques previously mentioned. After confirming the validity of the surrogates to address the problem at hand, Step 5 uses these surrogate models for parametric technology evaluation. The subsequent section describes how graphical analysis tools are leveraged for this purpose.

5.4 *Performing technology trade studies with surrogate models*

The previous sections described several enablers for rapid, high-fidelity analysis of systems of systems:

1. An integrated, hierarchical, physics-based modeling and simulation environment with technology factors and system performance attributes as the inputs and capability-based MoEs as the outputs
2. Neural network surrogate models to enable accurate high-speed analysis of the problem space while retaining the fidelity of the original modeling and simulation tools
3. A computational tool that applies a rigorous and repeatable mathematical procedure to the generation of valid neural network equations
4. Space-filling design of experiments to efficiently generate the data for neural network regression

The next step is to *use* the integrated surrogate-enabled environment for rapid parametric technology trade studies. In parametric design, surrogate models are purposefully varied in a deterministic manner to examine the impact of a specific change in design variables or technology factors on the overall response. When validated surrogates of high-fidelity tools are used, this procedure is identical to running the actual simulation as an analysis tool, with the notable exception of decreased run time for an acceptable degradation in accuracy and reduced degrees of freedom.

Recall that the surrogate model is simply an equation whose coefficients are calculated using the aforementioned computational tools. Use of the surrogate model for decision-making purposes is enabled by graphical tools that exercise the surrogate across the range of the input variables and technology factors. One such tool is the Prediction Profiler, a feature introduced in the JMP Statistical Package in the 1990s. An example of the use of surrogate models for aircraft design using the Prediction Profiler is shown in [Figure 5.7](#).

Each box in the prediction profiler depicts the “profile trace” of each X variable. These curved lines depict the change in the predicted response (Y-axis) across the range of an input variable (X-axis) as all other X variables are held constant and can be interpreted as a matrix of partial derivatives. The bold numbers in the middle of each box on the Y-axis highlight the current value

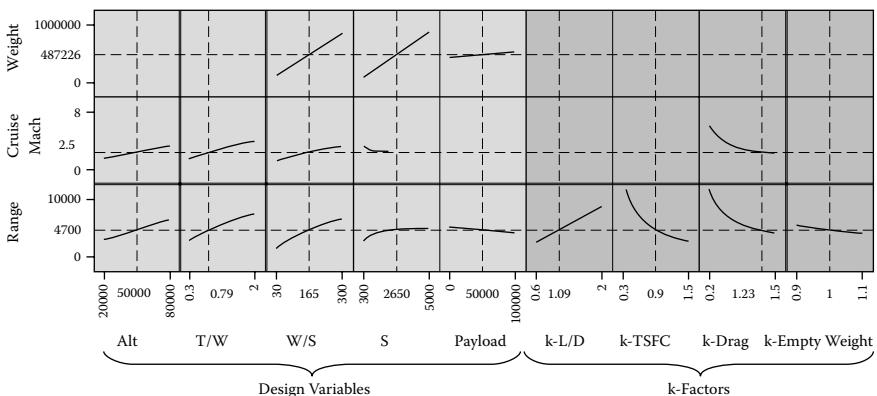


Figure 5.7 Prediction profiler for aircraft technology parameters [2].

for each of the three responses, weight, cruise Mach, and range. Moving the vertical dotted lines for any X variable recalculates the value of each of the three responses instantaneously by evaluating the surrogate model. In addition to the calculation feature, the slopes of the profile traces indicate the relative contribution of each X variable to each response at the current settings of all other variables and provide a measure of sensitivity. The prediction profiler in Figure 5.7 divides the X variables into two categories: design variables and technology k-factors. The former set includes design variables such as the aircraft cruise altitude (Alt), the thrust-to-weight ratio (T/W), wing area (S), and the like. The concept of “k-factors” was introduced by Mavris, Mantis, and Kirby to account for the fact that many computational tools “are typically based on regressed historical data, limiting or removing their applicability to exotic concepts or technologies” [88]. k-Factors can be interpreted as scale factors on the baseline values of discipline level metrics throughout the simulation hierarchy and affect variables related to technologies. In the set depicted here, technology factors that impact the lift-to-drag ratio, the thrust specific fuel consumption* (TSFC), the aircraft profile drag, and the aircraft empty weight are shaded in the figure. Using the prediction profiler’s calculation feature, technologists can quickly examine combinations of technology parameters that yield beneficial system-level performance by exercising the underlying surrogate models. From a sensitivity standpoint (by examining the slopes of the profile traces), a reduction in k-TSFC appears to result in the most significant change in aircraft range, and k-Drag reduction has the greatest benefit on cruise Mach. This method for quantitative assessment for technology factors has been demonstrated on several integrated engine/airframe combinations [7,89] and also across a hierarchy of military systems of systems [60].

* For those unfamiliar with the terminology, TSFC is a measure of fuel consumed per pound of thrust produced over time and is a typical technology metric for gas turbine engines. High TSFC corresponds to low fuel economy.

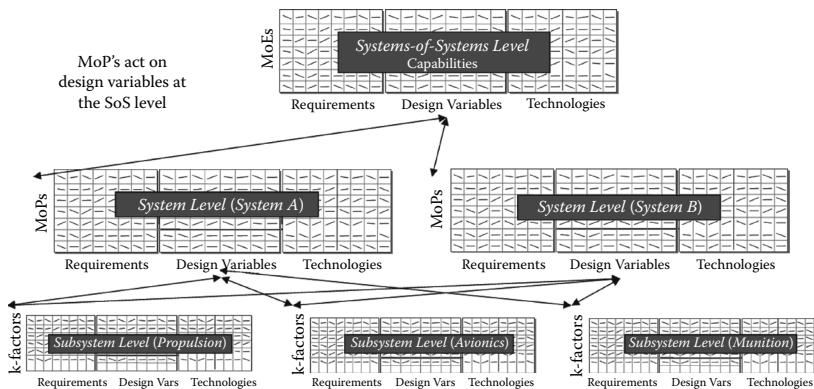


Figure 5.8 Multilevel linkage of surrogate models.

Whether the tradeoff environment wraps around the full integrated model or integrates individual surrogates at each hierarchical level, when the concept of the prediction profiler is extended to the systems-of-systems construct, the tradeoff environment takes the form depicted in Figure 5.8. Here, the Unified Tradeoff Environment proposed by Mavris, Baker, and Schrage is extended to a multilevel construct with discrete system and technology options where the outputs of each lower level act on the design variables at the next level [45]. At each level, the technology variables can be adjusted parametrically using the inputs to the surrogates, and different system architectures identify which technology suites and system concepts are included at the systems-of-systems level to provide capabilities. Using such an environment, it is possible to evaluate the impact of requirements against MoPs or MoEs, MoPs against MoEs, technology against MoPs for individual systems, or a portfolio of technologies across MoEs at the system-of-systems level in a rapid and traceable manner.

The concept of the profiler can also be extended to two and three dimensions using the JMP software. The same neural network equations can be used to understand trends in the technology space and how they relate to user-imposed constraints and scenarios. For the long-range bomber example, a parametric three-dimensional contour profiler for the systems of systems-level MoE “Targets Killed” is shown in Figure 5.9. In this case, the X and Y axes are speed and TSFC, respectively, but can be changed using the radio buttons to the right. The slide bars and text boxes to the right of the figure show the current input variable values for the baseline case. Manipulation of the slide bars changes the input values to the surrogate model and reevaluates the shape of the contour across the X and Y variables of speed and TSFC.

To illustrate how the surrogate models enable dynamic visualization, Figure 5.10 depicts the change in the contour when thrust-to-weight ratio (T/W) is increased to its maximum value. Here, the magnitude of the response

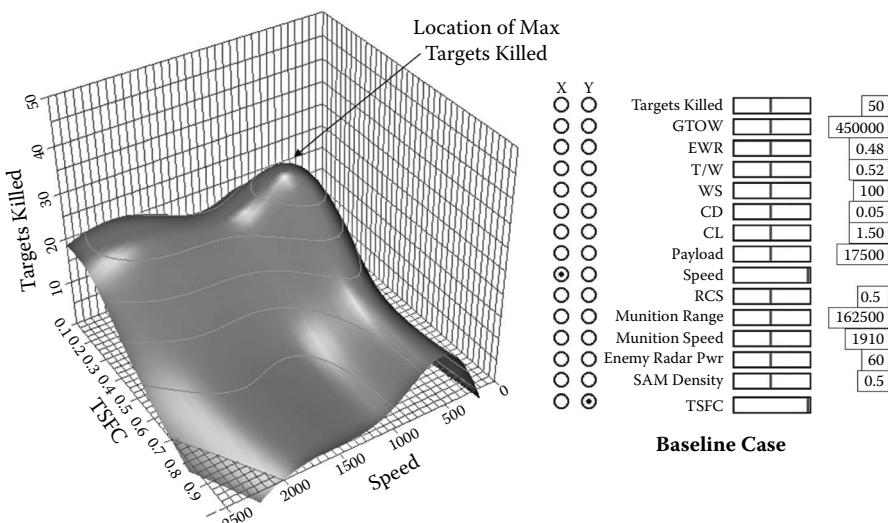


Figure 5.9 Three-dimensional contour profiler, baseline case [2].

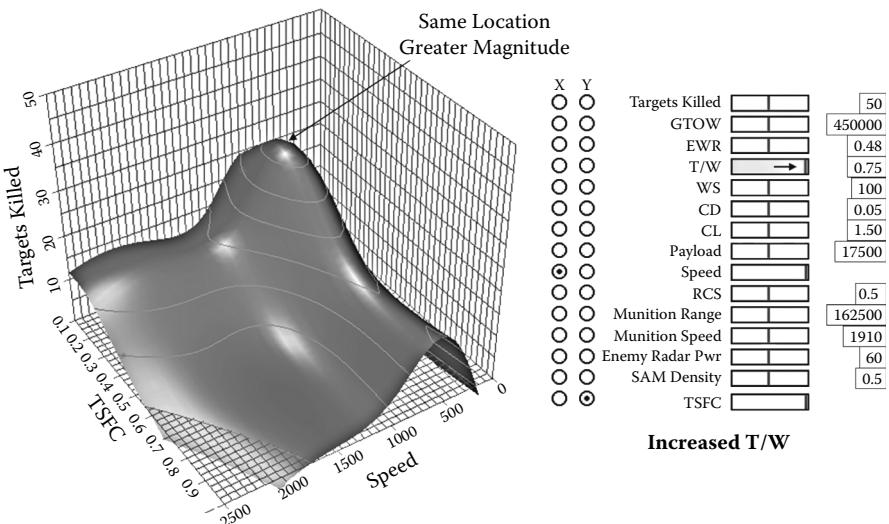


Figure 5.10 Three-dimensional contour profiler, increased T/W [2].

increases at the optimum speed and TSFC and decreases slightly for low-TSFC, high-speed concepts. Note also that there are some regions where the contour is intersected by the grey meshed area, notably at high speed and high TSFC in the lower left corner. For these settings of the input variables, no viable aircraft configurations can successfully complete the required mission. Finally, the contour profiler can also be used to assess sensitivities by

comparing the variation in the Z-axis response as the design variables move in either the X or Y dimensions. Here technologies which reduce TSFC have much greater benefit than those that increase speed, which actually penalize the “Targets Killed” MoE beyond a certain value. It is also interesting to note that the magnitude of the impact of TSFC-reducing technologies varies across a spectrum of vehicle speeds. In this example, there is an “optimum speed” for the baseline vehicle around 750 knots at which TSFC reduction appears to have a maximum benefit on the “Targets Killed” metric. To obtain a true estimate of the impact of these technology factors, this exercise must be repeated across multiple MoEs and multiple scenarios.

The parametric exercise of surrogate models using the graphical features of the JMP profiler are most appropriately termed “exploratory analysis,” which Davis notes is “particularly useful for gaining a broad understanding of the problem domain before dipping into details” [90]. Other analysis tools have moved to incorporate graphical data displays including the Prefuse Visualization Toolkit [91] and the Phoenix Integration® Visualization Pak for ModelCenter [92]. Exploratory analysis is also useful for eliminating regions of the technology space which provide little value at the systems-of-systems level due to insignificant contribution to identified MoEs.

While surrogate models provide value for quick-look exploratory analysis studies, the techniques described above are primarily used for assessing a desired portfolio of technologies or answering the question “what does this technology do?” This approach can be referred to as a bottom-up type of gap analysis: if thresholds are set on the MoEs, technology parameters can be changed to assess how close a proposed solution comes to meeting the thresholds. An alternative approach would be to set objectives at the systems-of-systems level and identify all the technology sets that meet those objectives. In contrast to the bottom-up approach, this method answers the question “what technologies do *this*?” and can be referred to as a top-down analysis or “inverse design.” Exploratory design offers a way to perform rapid parametric trades using a brute force approach or one-variable-at-a-time optimization, while inverse design uses probabilistic techniques to partially automate the search for an elegant solution to the same problem.

In practice, the notion of inverse design is not easily addressed by computerized analysis tools: they run in an inputs-to-outputs mode, not the other way around. One approach to address this might be to create a surrogate model (a mathematical function that maps inputs to outputs) and invert the surrogate. Unfortunately, direct mathematical inversion of the surrogate models to give the inputs as a function of a particular response is a very difficult problem. The inverse problem is nonunique, because any particular response often depends on multiple inputs; that is, the inverse mapping is one-to-many. Direct inversion is also complicated by nonlinearities in the surrogate models, which would require the identification of many multiple roots using branching techniques. While mathematically possible, a more practicable approach is to simply solve the forward problem over a span of

the input space and then to use this information to identify trends in the inputs that correspond to particular response criteria. This is the intent of a method known as Filtered Monte Carlo, first introduced by Kuhne et al. in 2005 [93]. In this approach, a large number of probabilistic cases are generated using Monte Carlo simulation (MCS), but those points that do not meet user-defined thresholds are discarded *a posteriori*. Essentially, inverse design is accomplished by performing forward design many times using probabilistic techniques. Here again, the surrogate models enable advanced analysis, since an MCS on the full analysis tool would require a prohibitively long time.

To demonstrate this technique, using the aforementioned surrogate models as the analysis engine, a uniform distribution is defined across the range two input parameters (speed and TSFC) to flood the design space with all possible technology sets within a discretized bound. Several thousand design points can be generated in seconds due to the rapid execution speed of the neural network equations. As opposed to the continuous surfaces observed in the exploratory analysis techniques previously described, the inverse design technique can be manifested in a multivariate scatterplot matrix as shown in the left side of [Figure 5.11](#).

In this simplified example, only two MoPs (speed and TSFC) are shown against one MoE, “Targets Killed.” The scatterplot matrix depicts each of the potential Y by X plots between the variables across the simulation hierarchy. The views shown are:

1. Targets Killed vs. Speed
2. Targets Killed vs. TSFC
3. Speed vs. TSFC

This trade shows how one variable at the technology factor, MoP, and MoE level can be related through physics-based surrogate models. Each of the points in Figure 5.11 represents an individual design that has been executed through the neural network equation from the subsystem to systems-of-systems level. Box 3 shows the two-dimensional uniform flood of points resulting from the MCS across speed and TSFC. Typically the lower-level parameters have a uniform scattering across their range unless other distribution types are used in the MCS. In contrast to the uniform spread in Box 3, both Boxes 1 and 2 exhibit a pattern. This is due to the fact that “Targets Killed” is calculated using the neural network. As with the other graphical examples in this chapter, the multivariate scatterplot matrix shows (Box 1) that there is a particular speed for which targets killed is maximized, and the MoE generally decreases as TSFC increases.

The right side of Figure 5.11 shows how the multivariate scatterplot matrix differs in character by applying a filter on MoEs using the Filtered Monte Carlo technique. Possible filters are highlighted by establishing color-coded thresholds in Box 1 of Figure 5.11 representing constraints on the desired number of targets killed. For example, if the highest threshold is used, indicated by

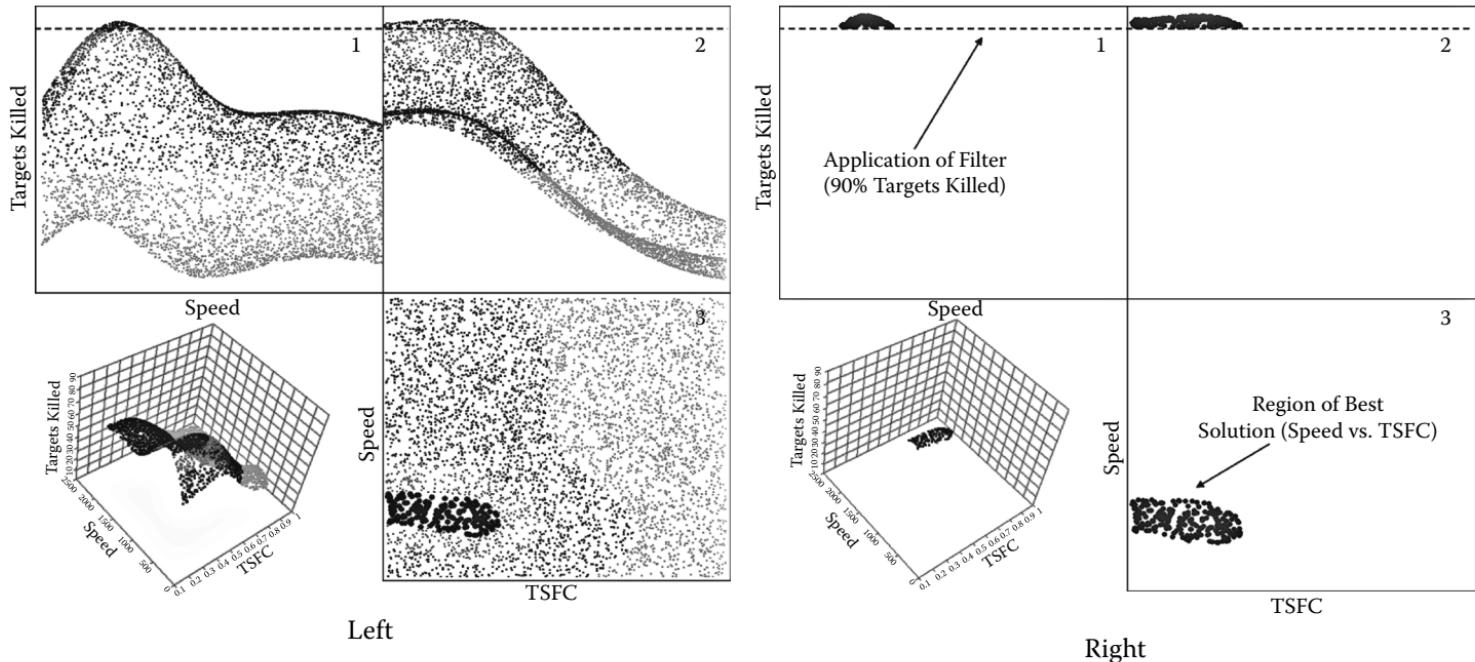


Figure 5.11 Using the multivariate scatterplot matrix and Filtered Monte Carlo for inverse design.

eliminating all points below the dashed line, only a few points in the left-center of Box 3 remain after the noncompliant points are discarded. The Filtered Monte Carlo technique is analogous to throwing ping pong balls into a box: each toss is a single case, and any balls that land outside the box are not considered. The number of points included in the final sample varies depending on the size of the box, essentially the constraints applied as filters on the process. If the surrogate models used to perform this type of analysis are inaccurate, valid points may miss the box, and invalid points may be included in the sample size, underscoring the need for accuracy in surrogate generation. In this allegorical example, the farther back you stand, the harder it is to hit the target. The difficulty of hitting finite targets increases when more analyses are included in the systems-of-systems simulation hierarchy, because there is more opportunity for the results of one analysis to be overshadowed by the results of another.

The use of probabilistic techniques in conjunction with surrogate models provides an unprecedented ability to rapidly play “what-if?” games and assess the benefit of technologies and their interactions under different conditions. These trades are extraordinarily time consuming when surrogates are not used for exploratory analysis. In most cases, the time taken to create the surrogate models is negligible when compared to the benefits of using them for parametric bottom-up analysis and probabilistic top-down inverse design and solution discovery.

5.5 Summary

The techniques outlined in this chapter for hierarchical modeling and simulation, surrogate model creation using artificial neural networks, multidimensional visualization of the problem, and inverse design using probabilistics can be used to quantify the benefit of potential technologies on systems of systems-level MoEs using physics-based analyses. In the simplified examples used in this chapter, technologies which reduced the fuel consumption of a notional aircraft solution were of inherently more value to the identified MoEs than technologies used to increase vehicle speed. Further dimensions such as cost, risk, and technology readiness can also be examined to make cost/benefit decisions on a portfolio of technologies across one or more scenarios, and the most valuable technology suites can be further assessed using traditional analysis methods and systems engineering techniques.

The surrogate modeling technique is extremely valuable for increasing the speed of the evaluation process and enabling exploratory analysis that would otherwise have been impossible due to the long run times of the simulation tools used. It should be clear that the optimization of the surrogates to find a single ideal point may be of less value than using the surrogates to eliminate inferior regions of the design space and understand sensitivities. Perhaps surrogates are best used to identify where the answer is *not* so that

resources can be focused toward additional analysis in promising regions of the design/technology space. While no technique for technology evaluation is perfect, the surrogate model-based approach contributes to all of the criteria identified in the introduction, most notably traceability, reusability, affordability, and agility.

One critical observation of the aforementioned techniques is that the “answers” to technology-related questions are not readily apparent. In fact, these techniques underscore the fact that an understanding of the vast opportunity space and enumeration of technology sensitivities is of inherently more value than a single optimized point. This is due primarily to the fact that systems of systems rarely have a “design point.” The long time-scales for development, uncertainties arising from many factors, and the fact that systems engineering is never finished make it extraordinarily difficult to forecast the exact performance of a suite of technologies on a system of systems. However, with enough information from subject matter experts and correctly constructed models of the appropriate fidelity, the methods in this chapter can be extended using probabilistic techniques to account for these sources of uncertainty. Future research is aimed at linking the results of parametric analysis to roadmapping tools for technology maturation through test and experimentation.

Finally, the importance of multifidelity or variable-fidelity models cannot be underemphasized. The level of fidelity used in the tool suite across the system-of-systems hierarchy shown in [Figure 5.1](#) should be tuned to address a set of analysis questions and should be appropriately allocated toward the regions of the greatest uncertainty and most technological promise. The fidelity level should also be set to answer the analysis questions within the time and cost constraints allowed; analysis should not be high fidelity, it should be right fidelity. This implies that the process enumerated in this chapter is an iterative one that relies on revisititation of the problem with subject matter experts and decision makers to arrive at the “best” decisions for technology infusion into systems of systems.

References

1. Betz, F. 1998. *Managing Technological Innovation: Competitive Advantage from Change*. John Wiley and Sons, Inc. New York.
2. Biltgen, P. T. 2007. A Methodology for Capability-Based Technology Evaluation for Systems-of-Systems. Ph.D. thesis, Georgia Institute of Technology.
3. Luce, B. R. 1993. *Medical Technology and its Assessment*. Del Mar Publishers, Albany, NY.
4. United States Air Force Scientific Advisory Board. 1996. *New World Vistas: Air and Space Power for the 21st Century*. United States Air Force Scientific Advisory Board. Washington, DC.

5. Gorn, M. H. 1997. Technological forecasting and the Air Force. In *Technology and the Air Force, a Retrospective Assessment*, ed. J. Neufeld, G. M. Watson Jr., and D. Chenoweth, 41–48. Air Force History and Museums Program, United States Air Force, Washington, DC.
6. Department of Defense. 1994. Rotary Wing Vehicle Technology Development Approach (TDA) 4.0, Technology Efforts and Objectives (TEO), U.S. Department of Defense.
7. Kirby, M. R. 2001. A Methodology for Technology Identification, Evaluation, and Selection in Conceptual and Preliminary Aircraft Design, Ph.D. thesis, Georgia Institute of Technology.
8. Withrow, M. 2005. AFRL demonstrates quantitative technology assessment. news@afrl, July 2005.
9. Zeh, J. 2005. Net-Centric Modeling, Simulation and Analysis. Air Force Research Laboratory, Presented at the 2005 FLAMES User's Conference, June 2005.
10. International Council on Systems Engineering (INCOSE) Technical Board. 2006. *Systems Engineering Handbook*. International Council on Systems Engineering, INCOSE-TP-2003-002-03, Version 3.0, June 2006.
11. Department of Defense. 2004. Defense Acquisition Guidebook. U.S. Department of Defense.
12. Berry, B. J. L. 1964. Cities as systems within systems of cities. *Papers of Regional Sciences Association* 13:147–163.
13. Schwartz, N. A. 2005. Joint Capabilities Integration and Development System, Chairman of the Joint Chiefs of Staff Instruction CJCSI 3170.01E.
14. Krygiel, A. J. 1999. Behind the Wizard's Curtain: An Integration Environment for a System of Systems. DoD C4ISR Cooperative Research Program.
15. United States Air Force Scientific Advisory Board. 2005. Report on System-of-Systems Engineering for Air Force Capability Development, Executive Summary and Annotated Brief, tech. rep., SAB-TR-05-04, United States Air Force Scientific Advisory Board.
16. Maier, M. W. 2006. Architecting principles for systems of systems. In Proceedings of the Sixth Annual International Symposium, International Council on Systems Engineering, Boston, MA.
17. Pohlmann, L. D. 2006. Is systems engineering for systems-of-systems really any different? PowerPoint presentation at the INCOSE International Symposium, Orlando, FL.
18. Crisp, H. and Ewald, B. 2005. Capability engineering for systems of systems: a coalition perspective. *INCOSE Insight* 8(1):3, 7.
19. U.S. Department of Defense. 2001 (amended 2008). DoD Dictionary, Defense Technical Information Center, Joint Publication 1-02, Online at <http://www.dtic.mil/doctrine/jel/doddict>, Updated April 14, 2006.
20. *The American Heritage Dictionary of the English Language*, 4th Edition. Houghton Mifflin Company, 2000.
21. Definition: Simulation, Online at <http://www.webopedia.com/TERM/S/simulation.html>.
22. National Academy of Sciences. 2002. *Modeling and Simulation in Manufacturing and Defense Acquisition: Pathways to Success*. National Academies Press, Committee on Modeling and Simulation Enhancements for 21st Century Manufacturing and Defense Acquisition, National Research Council.

23. National Center for Advanced Technologies. 1993. Technology For Affordability: A Report on the Activities of the Working Groups—Integrated Product/Process Development (IPPD), Simplified Contracting, Dual-Use Manufacturing, tech. rep., National Center for Advanced Technologies.
24. Acquisition Reform Office, ASN (RD&A), Department of the Navy. 1997. *Work Book for Video Series on Integrated Product And Process Development*, National Center for Advanced Technologies.
25. National Science Foundation. 2006. Simulation-based engineering science: revolutionizing engineering science through simulation. Tech. rep., National Science Foundation.
26. Kroo, I. 1997. Multidisciplinary optimization applications in preliminary design—status and directions. Invited Paper, AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, 38th, and AIAA/ASME/AHS Adaptive Structures Forum, Kissimmee, FL.
27. Box, G. E. P. and Wilson, K. B. 1951. On the experimental attainment of optimum conditions (with discussion). *Journal of the Royal Statistical Society Series B* 13(1):1–45.
28. Tai, J. C., Mavris, D. N., and Schrage, D. P. 1995. An application of response surface methodology to the design of tipjet driven stopped rotor/wing concepts. Presented at the 1st AIAA Aircraft Engineering, Technology, and Operations Congress, Anaheim, CA., September 19-21, 1995.
29. Skillen, M. and Crossley, W. 2005. Developing response surface based wing weight equations for conceptual morphing aircraft sizing. AIAA-2005-1960, 46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and 13th AIAA/ASME/AHS Adaptive Structures Conference, Austin, TX.
30. Carty, A. 2002. An approach to multidisciplinary design, analysis and optimization for rapid conceptual design. AIAA-2002-5438, Presented at the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA.
31. Jianjiang, C., Renbin, X., and Yiafang, Z. 2005. A response surface based hierarchical approach to multidisciplinary robust optimization design. *International Journal of Advanced Manufacturing Technology* 26(4):301–309.
32. Stewart, P., Fleming, P. J., and MacKenzie, S. A. 2002. On the response surface methodology and designed experiments for computational intensive distributed aerospace simulations. In *Proceedings of the 2002 Winter Simulation Conference*, Ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 476–482.
33. Chen, W., Allen, J. K., and Mistree, F. 1995. Robust concept exploration by combining Taguchi and response surface models. In *Proceedings of the 36th Structures, Structural Dynamics, and Materials Conference*, New Orleans, LA.
34. Chen, W., Allen, J. K., Mistree, F., and Tsui, K. L. 1995. Integration of response surface methods with the compromise decision support problem in developing a general robust design procedure. In *Proceedings of the 21st ASME Design Automation Conference*, Boston, MA.
35. Myers, R. H. and Montgomery, D. C. 2002. *Response Surface Methodology: Process and Product Optimization Using Designer Experiments*, 2nd Edition, Wiley, New York.
36. Koch, P. N., Mavris, D. N., and Mistree, F. 1998. Multi-level, partitioned response surfaces for modeling complex systems. Presented at the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO.

37. Roth, B., Mavris, D., and Elliott, D. 1998. A probabilistic approach to UCAV engine sizing. Presented at the 34th Joint Propulsion Conference, Cleveland, OH.
38. Lee, K., and Lee, T. H. 2001. Fuzzy multi-objective optimization of an automotive seat using response surface model and reliability method. Presented at the 4th World Congress of Structural and Multidisciplinary Optimization, Dalian, China.
39. Nixon, J. and Mavris, D. 2002. A multi-level, hierarchical approach to technology selection and optimization. Presented at the 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA, Atlanta, GA.
40. Mavris, D. N. and Kirby, M. R. 1999. Technology identification, evaluation, and selection for commercial transport aircraft. Presented at the 58th Annual Conference of Society of Allied Weight Engineers.
41. Mavris, D. N., Briceno, S. I., Buonanno, M., and Fernandez, I. 2002, A parametric exploration of supersonic business jet concepts utilizing response surfaces. Presented at the 2nd AIAA ATIO Forum, Los Angeles, CA.
42. Olson, E. D. and Mavris, D. N. 1997. Development of response surface equations for high-speed civil transport takeoff and landing noise. Presented at the 2nd World Aviation Congress and Exposition, Anaheim, CA.
43. Hosder, S., Watson, L. T., Grossman, B., Mason, W. H., Kim, H., Haftka, R., and Cox, S. E. 2001. Polynomial response surface approximations for the multidisciplinary design optimization of a high speed civil transport. *Optimization and Engineering* 2:431–452.
44. Mavris, D. N., Soban, D. S., and Largent, M. C. 1999. An application of a technology impact forecasting (TIF) method to an uninhabited combat aerial vehicle. Presented at the 4th World Aviation Congress and Exposition, San Francisco, CA.
45. Mavris, D. N., Baker, A. P., and Schrage, D. P. 2000. Simultaneous assessment of requirements and technologies in rotorcraft design. Presented at the 56th Annual Forum of the American Helicopter Society, Virginia Beach, VA.
46. Schrage, D. 1999. Technology for rotorcraft affordability through integrated product/process development (IPPD). Alexander A. Nikolsky Lecture, presented at the American Helicopter Society 55th Annual Forum, Montreal, Canada.
47. Mavris, D. N., Baker, A. P., and Schrage, D. P. 2000. Technology infusion and resource allocation for a civil tiltrotor. Proceedings of the AHS Vertical Lift Aircraft Design Conference, San Francisco, CA.
48. Biltgen, P. T. et al. 2004. Proteus: A Long Range Liquid Booster Target Vehicle, tech. rep., AIAA Missile Systems Technical Committee Graduate Strategic Missile Design Competition, Final Report, 2004.
49. Ender, T. R., McClure, E. K., Mavris, D. N. 2002. A probabilistic approach to the conceptual design of a ship-launched high speed standoff missile. Presented at the AIAA 2002 Missile Sciences Conference, Monterey, CA.
50. Kumpel, A. E., Barros, P. A., and Mavris, D. N. 2002. A quality engineering approach to the determination of the space launch capability of the peacekeeper ICBM utilizing probabilistic methods. Presented at the Missile Sciences Conference, Monterey, CA.
51. Mavris, D. 2004. Multi-Disciplinary Design Optimization Support for Surface Ship Projects, tech. rep., Georgia Institute of Technology, School of Aerospace Engineering.
52. Mavris, D. 2004. Multidisciplinary Optimization of Naval Ship Design and Mission, tech. rep., Georgia Institute of Technology, School of Aerospace Engineering.
53. Mavris, D. 2004. Application of Parametric Analysis to Aircraft Bus Timing, tech. rep., Georgia Institute of Technology, School of Aerospace Engineering.

54. Fitzgerald, C. J., Weston, N. R., Putnam, Z. R., and Mavris, D. N. 2002. A conceptual design environment for technology selection and performance optimization for torpedoes. Presented at the 9th Multi-Disciplinary Analysis and Optimization Symposium, Atlanta, GA.
55. Frits, A. P. 2005. Formulation of an Integrated Robust Design and Tactics Optimization Process for Undersea Weapon Systems. Ph.D. thesis, Georgia Institute of Technology.
56. DeLaurentis, D., Lim, C., Kang, T., Mavris, D. N., and Schrage, D. 2002. System-of-systems modeling for personal air vehicles. Presented at the 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Atlanta, GA.
57. Garcia, E. 2003. Development of a Framework for the Assessment of Capacity and Throughput Technologies within the National Airspace System. Ph.D. thesis, Georgia Institute of Technology.
58. Lewe, J. 2005. An Integrated Decision-Making Framework for Transportation Architectures: Application to Aviation Systems Design. Ph.D. thesis, Georgia Institute of Technology.
59. Soban, D. S. and Mavris, D. N. 2001. The need for a military system effectiveness framework—the system of systems approach. AIAA-2001-5226, presented at the 1st Aircraft, Technology Integration, and Operations Forum, Los Angeles, CA.
60. Soban, D. S. 2001. A Methodology for the Probabilistic Assessment of System Effectiveness as Applied to Aircraft Survivability and Susceptibility. Ph.D. thesis, Georgia Institute of Technology.
61. McCulloch, W. S. and Pitts, W. H. 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5:115–133.
62. Picton, H. 1994. *Introduction to Neural Networks*. Macmillan Press, New York.
63. Stanley, J. 1990. *Introduction to Neural Networks*. Scientific Software, Pasadena, CA.
64. Daberkow, D. D. and Mavris, D. N. 1998. New approaches to conceptual and preliminary aircraft design: a comparative assessment of a neural network formulation and a response surface methodology. Presented at the 3rd World Aviation Congress and Exposition, Anaheim, CA.
65. Johnson, C. and Schutte, J. 2005. *Basic Regression Analysis for Integrated Neural Networks (BRAINN)* Documentation, Version 1.2. Georgia Institute of Technology, Atlanta, GA.
66. Rojas, R. 1996. *Neural Networks—A Systematic Introduction*. Springer-Verlag, Berlin, New York.
67. Biltgen, P. T. 2006. Using FLAMES to enable capability-based design and technology evaluation. Presented at the 2006 FLAMES User Group Conference, Huntsville, AL.
68. Barros, P. A., Kirby, M. R., and Mavris, D. N. 2004. Impact of sampling technique selection on the creation of response surface models. Presented at the 2004 SAE World Aviation Congress, AIAA 2004-01-3134.
69. Engler, W. O. 2005. Creation of a Set of Parametric Engine Models Utilizing Neural Networks in a Systems-of-Systems Context. AE8900 Special Topics Report, Georgia Institute of Technology, School of Aerospace Engineering.
70. Rai, M. M. 2001. A rapid aerodynamic design procedure based on artificial neural networks. AIAA-2001-0315, presented at the 39th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.

71. Ender, T. R. 2006. A System-of-Systems Approach to the Design of an Air-Defense Weapon. Ph.D. thesis, Georgia Institute of Technology.
72. Saravanan, N., Duyar, A., Guo, T.-H., and Merrill, W. C. 1994. Modeling space shuttle main engine using feed-forward neural networks. *Journal of Guidance, Control, and Dynamics* 17(4):641–648.
73. Kranzusch, K. M. 2006. Abort determination with non-adaptive neural networks for the Mars Precision Landers. AIAA-2006-0149, presented at the 44th AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.
74. Mavris, D. N., Biltgen P. T., Ender, T. R., and Cole, B. 2005. Technology assessment and capability tradeoff using architecture-based systems engineering methodologies. Presented at the 1st International Conference on Innovation and Integration in Aerospace Sciences, Queens University Belfast, Northern Ireland, UK..
75. NIST/SEMATECH. 2006. e-Handbook of Statistical Methods. Online at <http://www.itl.nist.gov/div898/handbook/>, Updated July 18, 2006.
76. Box, G. E. P., Hunter, J. S., and Hunter, W. G. 2005. *Statistics for Experimenters: Design, Innovation and Discovery*, 2nd Edition. John Wiley and Sons, New York.
77. Fisher, R. A. 1921. Studies in crop variation. I. An examination of the yield of dressed grain from broadbalk. *Journal of Agricultural Science* 11:107–135.
78. Yule, G. U. 1969. *An Introduction to the Theory of Statistics*, MacMillan Publishing Company, New York.
79. Scheffe, H. 1959. *The Analysis of Variance*. John Wiley and Sons, New York.
80. Cochran, W. G., and Cox, G. M. 1950. *Experimental Designs*, Wiley, New York.
81. Taguchi, G. 1986. *Introduction to Quality Engineering: Designing Quality into Products and Processes*, Hong Kong: Asian Productivity Organization.
82. Mathworks. 2006. Creating Space-Filling Designs, Online at http://www.mathworks.com/access/helpdesk/help/toolbox/mbc/mbc_gs/f3-7640.html.
83. Sall, J. et al. 2005. *JMP® Design of Experiments*. SAS Institute, Inc., Cary, NC.
84. Fang, K. T. and Wang, Y. 1994. *Number-Theoretic Methods in Statistics*, Chapman and Hall, London.
85. Johnson, M., Moore, L., and Ylvisaker, D. 1990. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference* 26:131–148.
86. Ye, K. Q. 1998. Orthogonal column Latin hypercubes and their application in computer experiments. *Journal of the American Statistical Association Theory and Methods* 93(444):1430–1439.
87. Cioppa, T. M. 2002. Efficient Nearly Orthogonal and Space-Filling Experimental Designs for High-Dimensional Complex Models. Ph.D. thesis, Naval Post-graduate School, Monterey, CA.
88. Mavris, D. N., Mantis, G., and Kirby, M. R. 1997. Demonstration of a probabilistic technique for the determination of economic viability. SAE-975585, presented at the 2nd World Aviation Congress and Exposition, Anaheim, CA.
89. Buonanno M. and Mavris, D. 2005. A new method for aircraft concept selection using multicriteria interactive genetic algorithms. AIAA-2005-1020, presented at the 43rd AIAA Aerospace Sciences Meeting and Exhibit, Reno, NV.
90. Davis, P. K. 2000. Exploratory analysis enabled by multiresolution, multiperspective modeling. In *Proceedings of the 2000 Winter Simulation Conference*, ed. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick. A preliminary version appeared in *Proc. SPIE* Vol. 4026, pp. 2–15, *Enabling Technology for Simulation Science IV*, ed. A. F. Sisti. SPIE-International Society for Optical Engine.

91. Berkeley Institute of Design. 2007. Prefuse Visualization Toolkit. Online at <http://prefuse.org>, Last Accessed August 28, 2007.
92. Phoenix Integration. 2007. VisualizationPak, White Paper, Phoenix Integration, Philadelphia, PA.
93. Kuhne, C., Wiggs, G., Beeson, D., Madelone, J., and Gardner, M. 2005. Using Monte Carlo simulation for probabilistic design. Proceedings of the 2005 Crystal Ball User Conference.

chapter six

Enterprise system of systems

George Rebovich, Jr.

Contents

6.1	Classical systems engineering	166
6.2	System of systems engineering	167
6.2.1	Classical system of systems engineering	167
6.2.2	Systems of systems: a changing landscape.....	168
6.2.3	SoS system engineering: emerging principles	169
6.2.4	SoS systems engineering: toward a new view	170
6.3	Enterprise systems engineering	174
6.3.1	Enterprise and enterprise capabilities.....	175
6.3.2	Evolution of enterprise capabilities.....	176
6.3.3	Enterprise engineering	178
6.3.4	Achieving outcomes through interventions.....	179
6.3.5	A framework for evolving enterprise capabilities	181
6.3.6	Guiding and monitoring enterprise evolution.....	182
6.3.7	An example enterprise level engineering process.....	184
6.3.8	Governing and measuring enterprise capabilities	184
6.3.9	The enterprise market: changing the risk balance with service-oriented architectures.....	186
6.4	Summary and conclusions.....	187
	References	189

The 21st century is an exciting time for the field of systems engineering. Advances in our understanding of the traditional discipline are being made. At the same time new modes of systems engineering are emerging to address the engineering challenges of systems-of-systems (SoS) and enterprise systems. Even at this early point in their evolution, these new modes are evincing their own principles, processes and practices. Some are different in degree than engineering at the system level while others are different in kind.

While it is impossible to predict how the traditional and new forms of systems engineering will evolve, it is clear even now that there is a long and robust future for all three. Increases in technology complexity have led to

new challenges in architecture, networks, hardware and software engineering, and human systems integration. At the same time, the scale at which systems are engineered is exceeding levels that could only have been imagined a short time ago. As a consequence, all three forms of systems engineering will be needed to solve the engineering problems of the future, sometimes separately but increasingly in combination.

This chapter defines three modes of systems engineering, discusses the challenge space each addresses, describes how they differ from and complement each other, and suggests what their interrelationships should be in solving engineering problems of the future.

6.1 *Classical systems engineering*

Classical systems engineering is a sequential, iterative development process used to produce systems and subsystems, many of which are of unprecedented technical complication and sophistication. The INCOSE (ANSI/EIA 632) Systems Engineering process is a widely recognized representation of classical systems engineering [1].

An implicit assumption of classical systems engineering is that all relevant factors are largely under the control of or can be well understood and accounted for by the engineering organization, the system engineer, or the program manager, and this is normally reflected in the classical systems engineering mindset, culture, and processes.

Within most government agencies, systems are developed by an acquisition community through funded programs using classical system engineering methods and processes. The programs create a plan to develop a system and execute to the plan. The classical process works well when the system requirements are relatively well known, technologies are mature, the capabilities to be developed are those of a system, per se, and there is a single individual with management and funding authority over the program. It is estimated that the United States Department of Defense manages hundreds of systems of record being developed or modernized through funded programs of record using classical systems engineering methods and processes, as depicted in [Figure 6.1](#).

There are numerous variations on this classical systems engineering approach, including build-a-little, test-a-little incremental or spiral developments, to mitigate uncertainties in long-range requirements, technology maturity, or funding of the system.

The prevailing business model in most government development or modernization acquisition programs is to contract for the promise of the future delivery of a system that meets specified performance requirements, contract cost, and delivery schedule. These program parameters are set at contract award, and they form the basis for success or failure of the program and the individuals working on the program. This model of success shapes the organization's engineering processes, management approaches and the

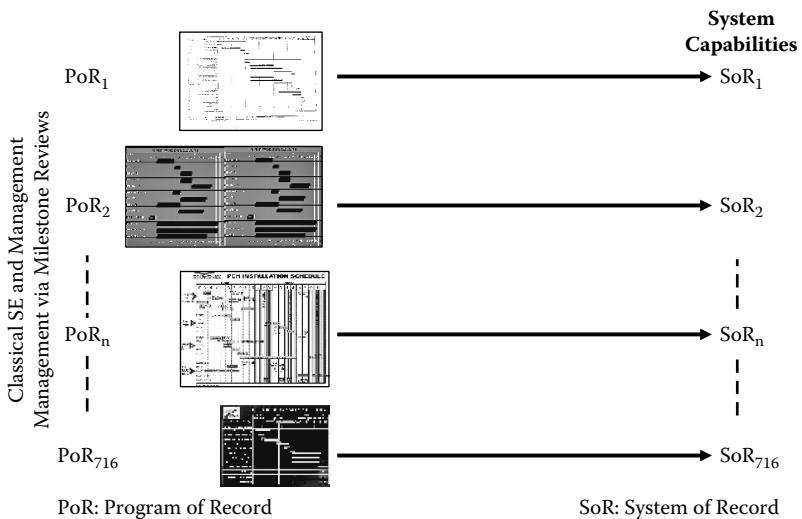


Figure 6.1 Classical approach to developing system capabilities.

motivations of program staff to place emphasis on tracking the progress of program parameters, uncovering deviations, and taking corrective action to get back on course to deliver according to the contract. This is normally accomplished through milestone reviews and other events that illuminate progress in the framework of the system performance requirements, cost, and delivery schedule.

6.2 System of systems engineering

6.2.1 Classical system of systems engineering

The classical approach to developing multisystem capabilities is through an executive oversight agency that aligns and synchronizes the development of individual systems to develop a capability that is greater than the sum of the individual systems. This is depicted in [Figure 6.2](#).

This approach works well for systems of systems (SoSs) comprised of individual systems that are being developed together as a persistent, coherent, unified whole, particularly when the identity and reason for being of the individual elements of these SoSs are primarily tied to the overarching mission of the SoS, the operational and technical requirements are relatively well known, the implementation technologies are mature, and there is a single program executive with comprehensive management and funding authority over the constituent systems. Examples of these types of SoS include the Atlas Intercontinental Ballistic Missile system, an air defense system, and the United States National Air and Space Administration's original Apollo Moon Landing capability.

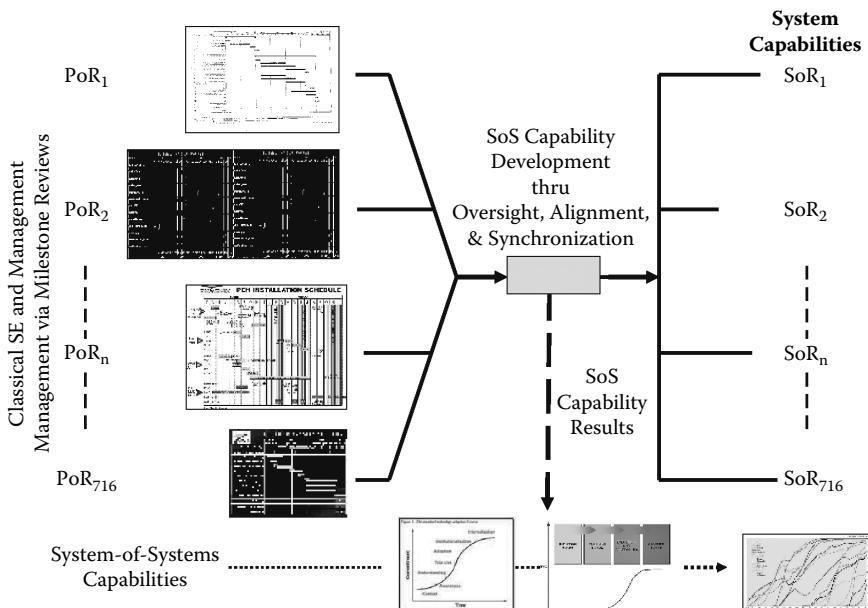


Figure 6.2 Classical approach to developing multisystem capabilities.

The classical approach to SoS development or modernization works for this category of SoS because it satisfies all the essential conditions and attributes of engineering at the system level, only it is larger. The community culture, organizational norms, rules of success, engineering and business processes, and best practices in this category of SoS either are essentially the same as at the system level or they scale appropriately.

6.2.2 *Systems of systems: a changing landscape*

In recent years SoS development has focused on engineering of capabilities across multiple existing systems. Capabilities, like the United States Army Future Combat System in which both the SoS and its constituent systems are built from the ground up are rare and, in practice, almost always require integration with legacy systems.

The more common SoS capability is typically an overlay on a collection of constituent systems. For the most part, constituent systems are existing systems that had been developed for specific users to support particular missions in certain environments. The different user communities each have their own culture, norms, and terminology. Missions may be different across the users of the various systems, and even if they are the same, the concepts of operation and employment likely are not. An example is the United States Defense Department's Single Integrated Air Picture program.

The development of an SoS capability tends to be an ongoing effort to build an overlaying cross-system capability involving an ensemble of individual systems. The overlay may touch on only a fraction of each constituent system's capabilities, but frequently it is a critical fraction. The focus of this overlay type of SoS development is on evolution of capability over time, with the capability taking one of several forms: (1) enhancing the way the underlying, existing systems work together, (2) adding new functionality through the incorporation of new systems or changes to existing systems, (3) providing enhancements that anticipate change in the SoS environment, or (4) reengineering systems (and, in some cases, eliminating systems) to provide a more efficient or effective capability.

Another distinctive attribute of these types of SoS developments is that the SoS manager typically does not control the requirements or funding for the individual systems. More often than not, the individual systems each have their own acquisition and development management organizations, structures, and funding lines. Sometimes the SoS capability development is funded as a collection of individual system modernizations. Thus, in an SoS environment a premium is placed on the ability to influence rather than direct outcomes, and it also affects the way in which SoS systems engineering is conducted.

From the single-system community's perspective, its part of the SoS capability represents additional obligations, constraints and complexities. Rarely is participation in an SoS seen as a net gain from the viewpoint of single-system stakeholders.

At the same time the technical complexity of SoS engineering is increasing dramatically, leading to new challenges in architecture, networks, hardware and software engineering, and human–system integration.

6.2.3 *SoS system engineering: emerging principles*

A set of principles is emerging from a United States Defense Department initiative to understand and differentiate engineering of these complex, increasingly common systems of systems [2]. Some are different in degree than engineering at the single-system level, while others are different in kind.

System engineering is performed in the context of a particular organizational structure and culture, and so the discipline has always had to be attuned to and aligned with the realities of organizations and their cultures. But in an SoS environment there are increased complexities in both organizational and technical dimensions. There is an increased need to foster relationships among the engineers and users of the systems. Additional complexity comes from a need to achieve a technical understanding of the systems, including their interrelationships and dependencies. Systems engineering trades at the SoS level factor into the objectives, plans, and motivations of the individual systems and vice versa.

Another emerging principle is that SoS systems engineering should focus on areas critical to the SoS, *per se*, and leave as much of the rest as possible to systems engineers of the individual systems. For example, the SoS integrated master schedule should focus on key synchronization (or intersection) points and dependencies among systems. Key processes and issues at the SoS level include configuration management, risk, and data interoperability.

The technical counterpart to the principle in the preceding paragraph is that SoS design should be based on open systems and loose couplings to the degree possible. The goal is to impinge on the individual systems as little as possible, thus providing them flexibility to address changing needs of the system-level users and apply the technologies best suited to those needs. This design principle provides benefits at the SoS level, as well, including extensibility and flexibility. This enables the addition or deletion of systems and changes in systems without affecting the other systems or the SoS as a whole.

Logical analysis has always been a fundamental process of system engineering. Systems engineering practitioners have found that this one-time, up-front process in a single-system setting becomes a more or less continuous process in an SoS environment. Sources of change, both internal and external, are more pronounced and persistent, with the result that the emphasis of logical analysis in an SoS setting is on foreseeing that change.

Not all best practices in a single-system setting scale well to an SoS when the number of systems involved exceeds a handful. Two examples are participation in other system milestone reviews and one-on-one intersystem meetings, both of which could take substantial time and resources. In these situations, the SoS technical management approach needs to emphasize single-system transparency across the SoS community as a way of achieving trust. One way to increase transparency is to make information that has historically been closely held within a single system's program office and immediate stakeholders accessible to the SoS community, perhaps via a shared web collaboration space. This "passive" transparency can then be augmented by active collaboration between system stakeholders on focused issues that make best use of human-to-human contact time.

6.2.4 SoS systems engineering: toward a new view

Discussions with United States Defense Department SoS systems engineering practitioners illuminate a view of how they perceive and do systems engineering that is different from their single-system counterparts.

For the most part, SoS system engineers view their world and frame their activities through seven megaprocesses which the SoS systems engineering team creates and tailors to the SoS, largely by drawing elements from across the 16 single-system technical and technical management processes of the Defense Department's *Defense Acquisition Guidebook*, depicted in [Figure 6.3](#).

In essence, the 16 *Defense Acquisition Guidebook* processes are a "parts box" used to build the SoS megaprocesses. Another major difference between the

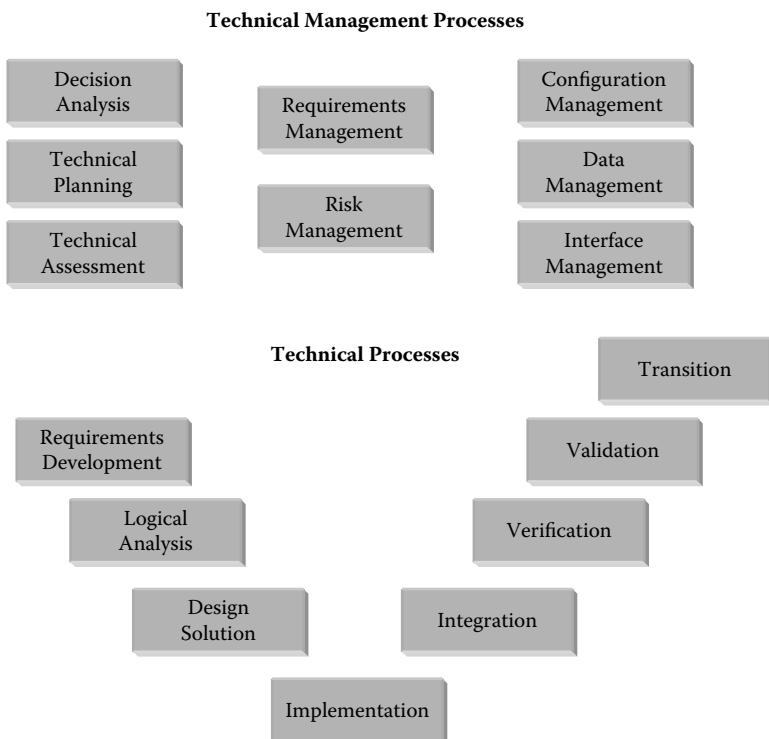


Figure 6.3 Technical and technical management processes.

Defense Acquisition Guidebook processes and the SoS megaprocesses is that, while there is usually some temporal order associated with the implementation of the former, in the latter the megaprocesses are viewed as being applied more or less continuously throughout the SoS lifecycle.

What follows is one view of the SoS systems engineering megaprocesses and their relationship to the *Defense Acquisition Guidebook* technical and technical management processes. It is intended primarily as an illustration of the ideas presented above.

One megaprocess is translating SoS capability objectives into high-level requirements over time. The focus is on developing a basic understanding of the expectations of the SoS and the core technical requirements for meeting those expectations, independent of the systems that will be constituents of the SoS.

A second megaprocess is understanding the systems of the SoS, their relationships, and plans for known change, over time. In SoS system engineering (SE), the focus is on the systems which contribute to the SoS capabilities and their interrelationships. This is differentiated from the single-system engineering focus on boundaries and interfaces.

A third SoS process is developing, evolving, and maintaining a high-level design or architecture of the SoS. This is a relatively persistent representation of the framework overlay of the SoS on the constituent systems. It includes concepts of operations and employment; descriptions of the systems, functions, relationships, and dependencies, both internal and external; end-to-end functionality and data flow. This process is directed toward addressing the evolution of the SoS to meet future needs, including possible changes in system functionality, performance, or interfaces.

A fourth megaprocess is continually monitoring proposed or potential changes and assessing their impacts to SoS or constituent system performance or functionality. This includes internal changes to technology or mission of the constituent systems as well as external demands on the SoS. The ability to discern and deal with external influences, such as changes in mission, technology, unplanned use of or demand for SoS capabilities, is critical. The focus of this process not only is on precluding or mitigating problems for the SoS and constituent systems, but also includes identifying opportunities for enhanced functionality and performance. An output of this process is changes to the understanding of constituent systems, their relationships, and known plans.

A fifth process is evaluating emerging new SoS requirements and options for dealing with them. This process involves reviewing, prioritizing, and determining which SoS requirements to implement next. The output is a detailed implementation for the SoS capability. This process contains a configuration control board type of function.

The sixth process orchestrates upgrades to the SoS. It includes planning and facilitating integration and developmental testing and evaluation activities.

Lastly, there is the ongoing need to assess actual performance of the SoS to the capability objectives. This requires SoS metrics and methods for assessing capability performance as differentiated from capability development.

Note the strong “continuing” aspect of these SoS processes signaled by phrases and words like *over time, evolving, monitoring, emerging, and ongoing*.

[Figure 6.4](#) notionally depicts the high-level relationship between the SoS megaprocesses and the 16 *Defense Acquisition Guidebook* technical and technical management processes. In general, the technical management processes are more heavily represented in the SoS megaprocesses, reflecting the SoS systems engineering role of coordination and orchestration across systems, with detailed engineering implementation taking place primarily at the system level.

[Figure 6.5](#) notionally depicts an SoS system engineering view of the interrelationships among the SoS processes. There is less structure in timing or sequencing than would be indicated by single-system waterfall, incremental, or iterative approaches to implementing systems engineering processes.

	Technical Management Processes								Technical Processes								
	Decision Analysis	Tech Planning	Tech Assess	Rqts Mgt	Risk Mgt	Config Mgt	Data Mgt	Interface Mgt	Rqts Level	Logical Analysis	Design Solution	Implement	Integrate	Verify	Validate	Transition	
SoS Mega-Processes																	
Translating Capability Objectives				x					x								
Understanding Systems & Relationships	x				x	x	x	x		x							
Developing, Evolving & Maintaining SoS Architecture	x	x		x	x	x	x	x	x	x	x						
Monitoring and Assessing Changes	x				x	x	x	x									
Addressing New Req'ts. & Implementation Options	x	x		x	x	x	x	x	x	x	x						
Orchestrating Upgrades to SoS	x	x	x	x	x	x	x	x				x	x	x	x	x	
Assessing Actual Performance to Capability Objectives	x		x		x		x			x					x		
										Reflects the SoS SE role of technical coordination and direction across systems						Reflects the fact that technical processes are primarily implemented by systems	

Figure 6.4 Relationship of SoS systems engineering megaprocesses to Defense Acquisition Guidebook processes.

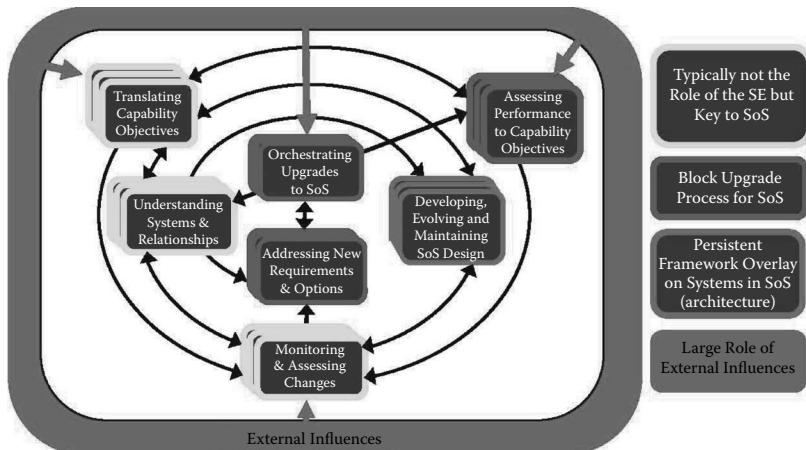


Figure 6.5 Interrelationships among SoS systems engineering megaprocesses (notional).

6.3 Enterprise systems engineering

Governments, large multinational organizations, and indeed all of society are in the midst of a major transformation driven by and deriving its character largely from advances in information technology.

The rate of technical change in information processing, storage, and communications bandwidth is enormous. Expansions in other technologies (e.g., netted sensors) have been stimulated and shaped by these changes. The information revolution is reducing obstacles to interactions among people, businesses, organizations, nations, and processes that were previously separated in distance or time. Surprisingly, future events in this information-abundant world are harder to predict and control, with the result that our world is becoming increasing complex. Why this is so is illustrated by Figure 6.6, in which our increasing interconnectedness (left side) makes us all coproducers of outcomes in airline flight availability as we vie for finite resource like non-stop connections whose accessibility and price can change with astonishing speed, as suggested by the online flight availability screen shots on the right side of the figure.

This new complexity is not only a consequence of the interdependencies that arise when large numbers of systems are networked together to achieve some collaborative advantage. When the networked systems are each individually adapting to both technology and mission changes, then the environment for any given system or individual becomes essentially unpredictable. The combination of large-scale interdependencies and unpredictability creates an environment that is fundamentally different from that at the system or SoS level. As a result, systems engineering success expands to encompass

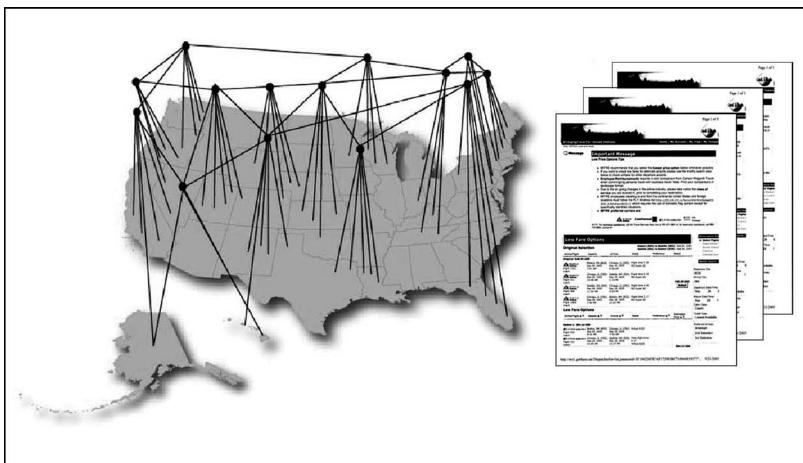


Figure 6.6 Our world is becoming increasingly complex [3].

not only success of an individual system or SoS, but also the network of constantly changing systems. Examples in which this new complexity is evident include the United States Federal Aviation Administration's National Aerospace System, the Defense Department's Global Information Grid, the Internal Revenue Service's Tax Systems, and the Department of Homeland Security's Secure Border Initiative's SBInet.

6.3.1 *Enterprise and enterprise capabilities*

By "enterprise" we mean an association of interdependent organizations and people, supported by resources, which interact with each other and their environment to accomplish their own goals and objectives and those of the association. Resources include manpower, intellectual property, organizational frameworks and processes, technology, funding, and the like. Interactions include coordination of all types, sharing information, allocating funding, and the like. The goals and objectives of the various organizations and individuals in the enterprise will sometimes be in conflict.

In the business literature an enterprise frequently refers to an organization, such as a firm or government agency; in the computer industry it refers to any large organization that uses computers (e.g., as in Enterprise Resource Planning systems). The definition of enterprise in this chapter is intended to be quite broad and emphasize the interdependency of individual systems and systems of systems, and the emergence of new behaviors that arise from the interaction of the elements of the enterprise. The definition includes firms, government agencies, large information-enabled organizations, and any network of entities coming together to collectively accomplish explicit or implicit goals. This includes the integration of previously separate units [4]. Examples of enterprises include:

- A chain hotel in which independent hotel properties operate as agents of the hotel enterprise in providing lodging and related services while the company provides business service infrastructure (e.g., reservation system), branding, and the like.
- A military command and control enterprise of organizations and individuals that develop, field, and operate command and control systems, including the acquisition community and operational organizations* and individuals that employ the systems.

The systems engineering literature is replete with phrases like “system capabilities” and “SoS capabilities,” so the question arises, “what is an enterprise capability, and how does it differ?”

An enterprise capability involves contributions from multiple elements, agents, or systems of the enterprise. It is generally not knowable in advance of its appearance. Technologies and their associated standards may still be emerging, and it may not be clear yet which will achieve market dominance. There may be no identifiable antecedent capability embedded in the cultural fabric of the enterprise, and thus there is a need to develop and integrate the capability into the social, institutional, and operational concepts, systems, and processes of the enterprise.

The personal computer emerged as a replacement for the combination of a typewriter and a handheld calculator, both of which were firmly embedded in our social, institutional, and operational concepts and work processes. The personal computer is not an enterprise capability by this definition. But the Internet is an enterprise capability. Its current form could not possibly have been known in the 1980s. Its technology has been emerging and continues to do so. More fundamentally, there was no identifiable antecedent capability embedded in the cultural fabric of our society before the Internet’s emergence, nor were there associated problems and solutions to issues like identity theft, computer viruses, hacking, and other information security concerns.

6.3.2 *Evolution of enterprise capabilities*

Enterprise capabilities evolve through emergence, convergence, and efficiency phases, as suggested by the stylized s-curve in [Figure 6.7](#). This is similar in its essentials to Rogers’ diffusion of innovation curve [5], which later influenced Moore’s technology adoption curve [6]. Emergence is characterized by a proliferation of potential solution approaches (technical, institutional, and social). Many of these potential solutions will represent evolutionary dead ends and be eliminated (convergence) through market-like forces. This is followed by a final period (efficiency) in which the technology is integrated and

* This example is intended to include government organizations, non-profits, and commercial companies.

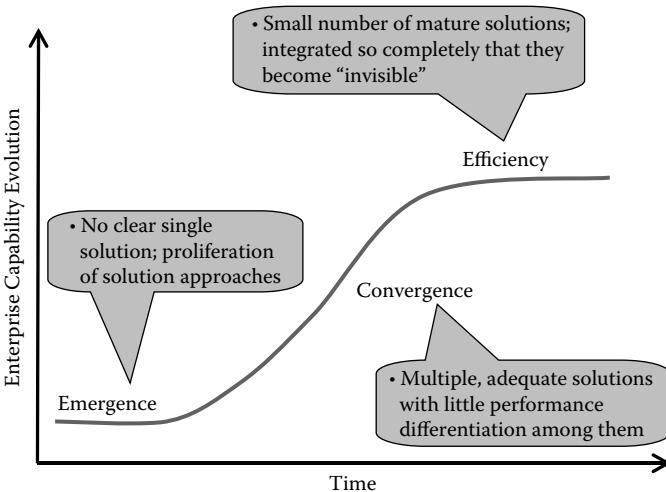


Figure 6.7 Phases of enterprise capability evolution and their characteristics [10].

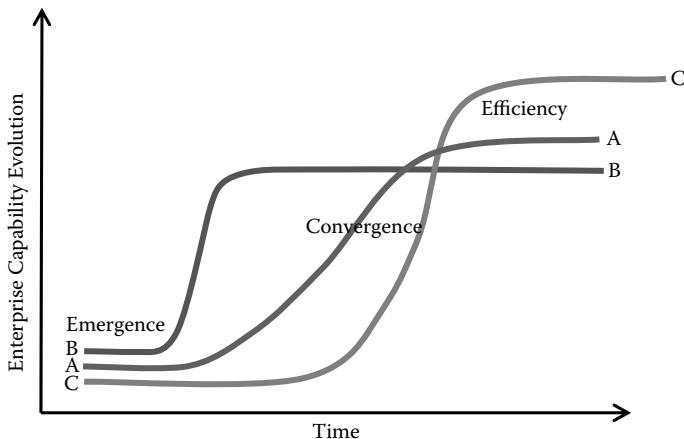
operationalized to such a degree that it becomes invisible to the humans, institutions, and social systems that use them.

Enterprise capabilities evolve through emergence, convergence, and efficiency phases whether or not an enterprise (or society) has intervention processes in place to actively manage them. Interventions, whether purposeful or accidental, can alter the shape of the evolutionary curve, the speed at which evolution progresses, and the level of evolutionary development achieved. This is notionally depicted in Figure 6.8. For illustration purposes, assume that curve A depicts how a capability would evolve in an enterprise without explicit, purposeful interventions.

In curve B, enterprise engineering processes shorten the exploration phase (perhaps by early down-selecting to a small number of acceptable enterprise-wide standards for a particular technology). This provides the benefit of converging more quickly to an efficiency phase, but at the cost of a less optimal efficiency phase (perhaps because superior alternatives were never explored due to the foreshortened emergence phase). Conventional examples of this type of premature convergence include the competition between VHS and Betamax systems of video recording, and QWERTY and Dvorak keyboard arrangements.

Curve C depicts a situation in which exploration of an enterprise capability is extended, perhaps to consider additional emerging technology alternatives. This has the effect of deferring exploitation of a preferred approach beyond either of the other two curves, but in the end it results in the most successful efficiency phase.

Figure 6.8 is not meant to suggest that foreshortened exploration necessarily leads to a less optimal efficiency phase or that extended exploration guarantees a more successful one. There are no hard and fast rules. Too much exploration can leave an organization permanently disorganized so that new



A: Evolution without purposeful interventions (notional)
 B: Foreshortened exploration; quicker convergence; less optimal efficiency phase
 C: Extended exploration; longer convergence; more successful efficiency phase

Figure 6.8 Shaping, enhancing, and accelerating enterprise capability evolution.

ideas have their underpinnings swept away in subsequent change before it is known whether they will work. Aggressive exploitation risks losing variety too quickly, which can happen when fast reproduction of an initial success cuts off future exploration and possible improvement. These are not just two ways that a good concept can go wrong. The two possibilities form a fundamental trade-space. Investments in options and possibilities associated with exploration usually come at the expense of obtaining returns on what has already been learned [8].

The critical role of enterprise engineering processes is to shape, enhance, and accelerate the “natural” evolution of enterprise capabilities. In the emergence phase, interventions should favor and stimulate variety and exploration of technologies, standards, strategies, and solution approaches and their integration and operationalization in and across enterprise organizations, systems, and operations. In shaping convergence, the goal of interventions is to narrow the solution approaches and start to balance exploitation of more robust solutions with exploration of promising, emerging alternatives. In the efficiency phase, interventions favor exploitation of that which is known to work through proliferation of a common solution approach across the enterprise. This is notionally depicted in [Figure 6.9](#).

6.3.3 Enterprise engineering

Enterprise engineering is an emerging mode of systems engineering that is concerned with managing and shaping forces of uncertainty to achieve results through interventions instead of controls. It is directed toward enabling and achieving enterprise-level and cross-enterprise capability

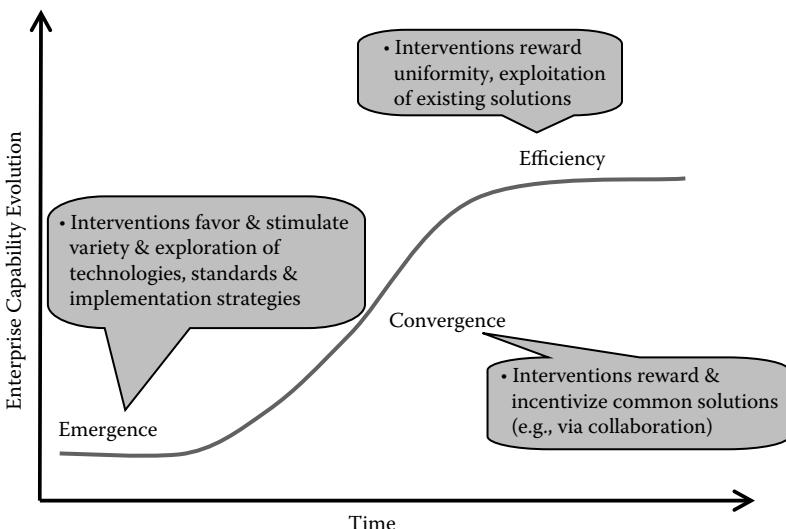


Figure 6.9 Role of purposeful interventions in shaping enterprise capability evolution.

outcomes by building effective, efficient networks of individual systems to meet the objectives of the enterprise. Enterprise engineering manages the inherent uncertainty and interdependence in an enterprise by coordinating, harmonizing, and integrating the engineering efforts of organizations and individuals through processes informed or inspired by evolution (both natural and technology) [9] and economic markets [10].

Enterprise engineering is a multidisciplinary approach that encompasses, balances, and synthesizes technical and nontechnical (political, economic, organizational, operational, social, and cultural) aspects of an enterprise capability.

Enterprise engineering is based on the premise that an enterprise is a collection of agents (individual and organizational) that want to succeed and will adapt to do so [11]. The implication of this statement is that enterprise engineering processes are focused on shaping the outcome space and incentives within which individuals and organizations develop systems, so that an agent innovating and operating to succeed in its local mission will—automatically and at the same time—innovate and operate in the interest of the enterprise. Enterprise engineering processes are focused more on shaping the environment, incentives, and rules of success in which classical engineering takes place.

6.3.4 Achieving outcomes through interventions

The vast majority of government agencies are hierarchical, both organizationally and culturally. Moreover, government statutes and policies for engineering and acquiring systems (contract for the effort that promises to deliver

a future capability within specified performance, cost, and schedule parameters) and the classical approaches to systems engineering have coevolved so that there is great sympathy and harmony between them. Together, these realities lead many to believe that government activities are necessarily top-down, and control-oriented (decision made at the top, execution performed at the bottom and monitored from the top) and that interventional system engineering and management cannot be employed now but must wait for expansive changes in government policy and statutes [12].

This chapter takes the point of view that good systems engineering and management has always been informed by diverse disciplines, usually intuitively and informally, and that there is ample room for expanding and formalizing that practice and applying it to the engineering of government enterprise capabilities. What is needed is a change of mindset that enables engineering and acquisition practitioners to question prevailing, largely implicit assumptions under which most organizations operate [13].

The United States Federal Reserve System provides an example of a government agency that helps manage the enormously complex United States economy to achieve outcomes through interventions. As measured by gross domestic product, the United States economy is estimated at \$12.4 trillion, involves nearly 10,000 publicly traded companies and millions of consumers. All of these companies and consumers are operating in their own self-interests.

By law, the Federal Reserve is charged with maintaining a balance between growth and inflation in the United States economy. Remarkably, the Federal Reserve has basically four tools available to it to maintain this balance. It can sell or purchase government securities, change the reserve requirements for banks, change the discount rate at which banks borrow money from the Federal Reserve, and change the short-term Federal Reserve funds rate at which banks borrow money from each other.

Separately and in combination, these mechanisms serve to increase or decrease the supply of money in the economy. Great economic analysis skill is needed in deciding how many securities to sell or buy and when, and whether and how much to change reserve requirements, discount and Federal Reserve funds rates, and when. But, generally, the economy responds in a way the Federal Reserve intends.

The Federal Reserve harnesses the complexity of the myriad of interconnected organizations and individuals in the United States economy through a handful of interventions to achieve its purpose. Companies and consumers innovate to make and change decisions in response to the Federal Reserve's interventions in a way that serves their own interests and—at the same time—the interests of the Federal Reserve.

Think about managing the acquisition of government enterprise capabilities in a similar way. What are the big levers in government acquisition that could shape outcome spaces and create incentives for individual programs in which they meet their own program goals while solving the problems of the enterprise? A definitive answer to that is not yet known, but the levers

likely surround managing the balance of technology exploration and exploitation to focus and accelerate the evolution of enterprise capabilities through its maturity curve (reference Figure 6.7 through 6.9).

What are the systems engineering disciplines and processes that support decision makers to move levers in one direction or the other? System engineering at the enterprise level may be the counterpart to economic analysis at the Federal Reserve System level (technical analysis and forecasting to support “moving the levers”). And this shapes and changes the environment for classical systems engineering at the program level, which is about skillfully responding to the environment that surrounds the program (which is analogous to company financial experts who provide technical support to senior company management in making financial decisions in changing economic times). An example of what an enterprise-level engineering process might look like is presented later in this chapter.

6.3.5 A framework for evolving enterprise capabilities

Figure 6.10 shows an approach in which enterprise engineering processes shape the evolution of enterprise capabilities through emergence, convergence, and efficiency phases via evolutionary or market-like mechanisms at

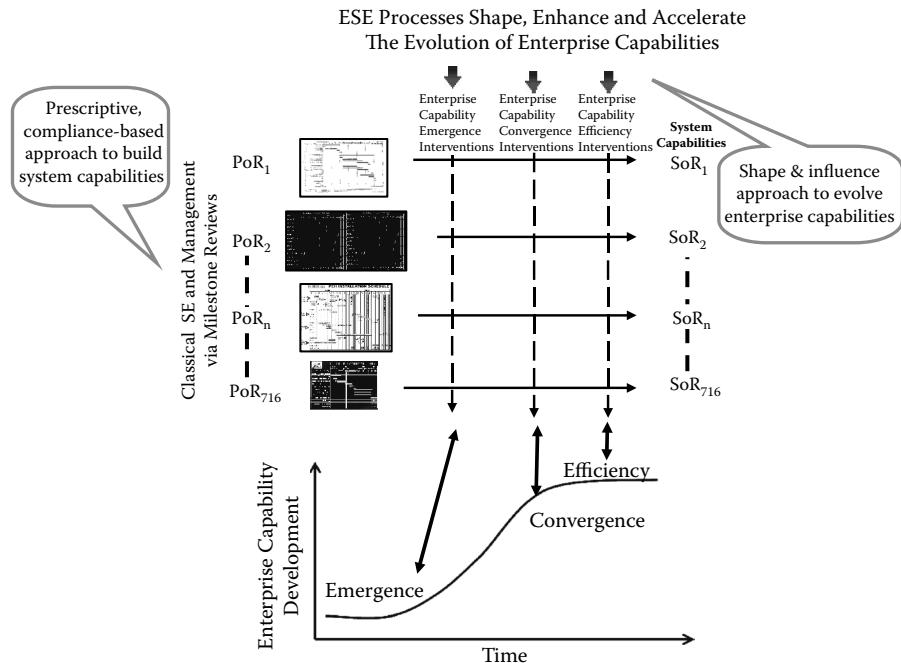


Figure 6.10 A framework for evolving enterprise capabilities.

the same time that individual system capabilities are being developed via the classical system engineering approach of building to a plan.

The basic notion is to stimulate innovation by and interactions among programs of record to move the enterprise toward an enterprise capability at the same time the programs are developing their systems of record. Specific interventions depend on the phase or state the enterprise capability is in.

This approach is similar in its essentials to the Federal Reserve intervening in the United States economy in which the collective response of organizations and individuals operating in their own self interests to those interventions serve their needs and those of the Federal Reserve at the same time.

6.3.6 *Guiding and monitoring enterprise evolution*

Exploration versus exploitation is an important trade between the creation of untested solutions that may be superior to solutions which currently exist and have so far proven best. This trade occurs across a wide range of circumstances in which the exploration of that which is new or emerging (variety) comes at some expense to realizing benefits of that which is already available (homogeneity).

It is not always the case that variety is good and homogeneity is bad, or vice versa. More variety is indicated during the emergence phase of an enterprise capability with a movement toward increasing homogeneity as an enterprise capability moves through convergence and efficiency phases of its evolution. The criteria for shaping that change will differ depending on the phase. [Table 6.1](#) summarizes characteristics of each phase of enterprise capability evolution and ideas for shaping the exploration/exploitation balance.

An enterprise capability is a characteristic of the enterprise in its operation. The implication is that enterprise performance should be strongly tied to the behavior of operational units employing enterprise systems and capabilities in actual operations. Measures intended to monitor the evolution of enterprise capabilities should focus on capturing changes in the way operational units interact. The evolution and utilization of enterprise capabilities have strong elements of social system structure and dynamics. The implication is that the definition of enterprise measures should include sociologists as well as operational and technical experts. Formal verification of the piece-parts of an enterprise capability will still need to be done as part of system sell-offs, but they should not be viewed as the primary indicators of an enterprise capability. Even as simple a system as a wristwatch is primarily evaluated holistically (e.g., does it tell time correctly?) and not as the pair-wise interactions of its myriad mechanical and electrical parts. [Table 6.2](#) suggests some examples of measures for monitoring the evolution of military interoperability at the enterprise level.

Table 6.1 Guiding Enterprise Evolution

Enterprise Capability Phase Characteristics	Examples, Rules of Thumb, and Anecdotes
Emergence	<p>Emergence</p> <ul style="list-style-type: none"> When there are no clear solutions or multiple, emerging solutions. When extensive or long-term use can be made of a solution. When there is low risk of catastrophe from exploration.
Convergence	<p>Convergence</p> <ul style="list-style-type: none"> Narrow the solution space by providing rewards or incentives to programs and contractors that achieve common solutions through collaboration. Solutions are not prescribed from above.
Efficiency	<p>Efficiency</p> <ul style="list-style-type: none"> Reward exploitation of existing solutions Reward use of common solutions Solution is not specified from above.

Table 6.2 Monitoring Enterprise Evolution

Emergence	Convergence	Efficiency
<ul style="list-style-type: none"> Increase total no. of interface control documents among programs of record. Increased volume of voice, email, chat & instant messaging among operational units. Communication emerging among previously noninteracting units. 	<ul style="list-style-type: none"> Decrease in number of interface control documents. Increased use of common standards among programs of record. Less episodic, more continuous interactions among operational units. 	<ul style="list-style-type: none"> Predominant use of single standard among operational units. Predominantly continuous interactions among operational units.

6.3.7 An example enterprise level engineering process

At the system level, technical planning and assessment processes address the scope of the technical effort required to develop a system, and measure the progress and effectiveness of the technical requirements and the plans to achieve them.

Given the relationship between system development and enterprise capability evolution depicted in [Figure 6.10](#), technical planning at the enterprise level may need to be more about capability forecasting and evolution. The forecasting piece is about the identification of what phase a capability is in now and where it is heading in its evolution or adoption (e.g., emergence, convergence, or efficiency phase as depicted in [Figure 6.7](#)). The evolution piece is about shaping the capability evolution curve ([Figure 6.8](#)) within the government enterprise environment and moving the application and institutionalization of the capability up the evolutionary maturity curve ([Figure 6.9](#)). The goal should be to accelerate the natural processes of evolution with an emphasis on application to, e.g., military command and control systems. Innovation and other behavior that moves the enterprise up the evolution curve needs to be incentivized and rewarded. Specific criteria for rewards and incentives depend on the stage of capability evolution ([Table 6.1](#) and [Figure 6.10](#)). For example, to stimulate the emergence phase of an enterprise capability evolution, Defense Advanced Research Projects Agency (DARPA) Grand-Challenge-like events could proliferate a number of technical approaches quickly.

The enterprise view of and process for capability planning should focus on the management of the population of technology initiatives to achieve the right exploration/exploitation balance, not on the outcomes of individual, specific initiatives. In this way, specific technologies, products, and/or standards emerge as the program-of-record marketplace innovates ideas and rewards winners.

Thus, the focus of capability assessment and planning at the enterprise level is on identifying broad technology trends, shaping DoD technology outcome spaces, and managing the interactions of enterprise participants so that solutions emerge through market-like forces, as opposed to the current practice of having high-level government offices identify and mandate specific products or standards as de facto winners in the marketplace of ideas.

6.3.8 Governing and measuring enterprise capabilities

[Figure 6.11](#) depicts a framework for relating governance approach and measures of success to different situations in acquiring government capabilities. Much of the experience in engineering and acquiring government systems falls in the lower left-hand quadrant. Prescriptive, requirements compliance-based approaches work well in delivering systems built on a mature and homogeneous technology base using classical systems engineering processes.

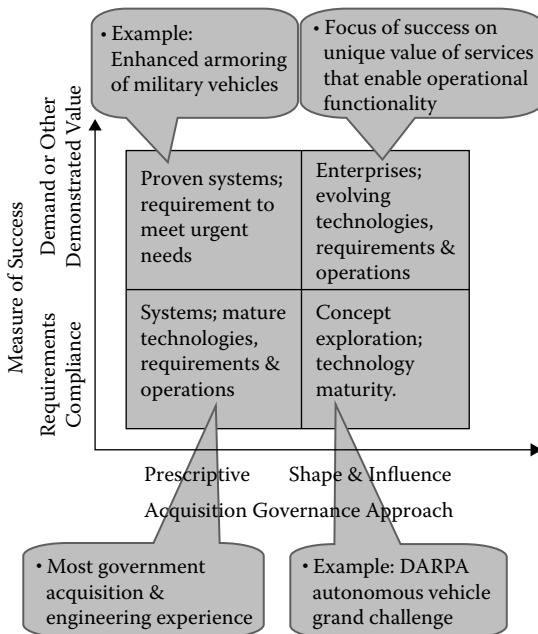


Figure 6.11 Governing and measuring enterprise capabilities.

But their utility in a net-centric environment is less clear. Increasingly, the net-centric environment is characterized by threads of functionality that are put together to serve an immediate operational need and then, just as quickly, are reassembled in another way for another purpose. The focus of success will shift to demonstrated value of services that enable functionality in operational environments. Increasingly, enterprise demand for a service or offering will become its measure of value. Government programs and the contractors that support them will increasingly ask and answer “what is it of unique value?” that we provide to the enterprise. Evolving enterprise capabilities is best served by the approach in the upper right-hand quadrant. While employed less frequently, the approaches represented by the two other quadrants have their uses: the lower right quadrant for concept explorations like the DARPA Autonomous Vehicle Grand Challenge and the upper left quadrant for meeting immediate, critical needs like enhanced armoring of existing military vehicles.

There have been efforts in recent years to develop government enterprise information technology acquisition policies and processes that are less prescriptive and attempt to provide more latitude for programs to collaborate and innovate (lower right quadrant of Figure 6.11). Many understand and appreciate the need for different governance approaches and success models and, indeed, even start in that direction. But there are deep structural issues in government enterprise information technology acquisitions that continue

to drive policy makers, program offices, and contractors to the lower left-hand quadrant of [Figure 6.11](#) regardless of their specific circumstances or intentions. The prevailing business models used in government programs (contract for the promise of a future capability) encourages programs and contractors to ask for more detailed guidance (to minimize cost, schedule, and requirements risk of the contract) and discourages them from creating high-demand services or offering because of the risk of driving the program out of its predefined requirements and expectations.

6.3.9 The enterprise market: changing the risk balance with service-oriented architectures

The advantages normally cited for a service-oriented architecture* approach to networked information technology systems are reduced complexity and cost of integration, enhanced reusability, better identification and management of dependencies among systems, and industry compatibility at the service level. Potentially more significant is the prospect that a service-oriented architecture can move government acquisition away from paying for effort to make good on a promise (the contract) toward a market-like economy in which contractors develop product or service offerings that compete for market share. This has enormous implications for shifting the balance of risk in government enterprise acquisition from being essentially wholly owned by the government to being shared between the government, as consumer, and contractors as producers of services that are competing for market share [14].

What follows is a simple example to illustrate the point. Consider a government community consisting of producers of a unique type of data (e.g., sensor data) and consumers who use the data to accomplish activities ranging from broad situation awareness, precise and immediate location finding, and detailed historical analysis. Taking a service-oriented architecture approach to developing enterprise capabilities for the production and consumption of this sensor data suggests separating data, exploitation tools, and visualization tools in a relatively machine-independent way. Consider all potential government producers and users of the sensor information as the market for data services, exploitation tools, and visualization tools. Commonality for data flow and storage is critical across the entire market. The need for exploitation tools is driven by specific mission and activities, so its market is characterized by some common tools and some specialized ones. The market for visualization tools is probably many common and a few specialized ones. So, the one community behaves as three different markets: one for data,

* A *service* is a functional capability that is made available to consumers via an exposed network access point. From a consumer's point of view, services are black boxes on the network in that their internal implementation is hidden from the consumer.

another for exploitation tools, and a third for visualization tools. This suggests a stakeholder community of practice which governs the three levels of the service-oriented architecture differently: a prescriptive, top-down governance approach for data, and a shaping/influencing governance approach for tools to encourage and enable contractors to innovate and differentiate their products and services and be rewarded with a larger market share.

6.4 Summary and conclusions

This chapter has defined three modes of systems engineering, discussed the challenge space each addresses, described how they differ from and complement each other, and suggested what their interrelationships should be in solving engineering problems of the future.

Classical systems engineering is a sequential, iterative development process used to produce systems and subsystems, many of which are of unprecedented technical complication and sophistication. An implicit assumption of classical systems engineering is that all relevant factors are largely under the control of or can be well understood and accounted for by the engineering organization, the system engineer, or the program manager and this is normally reflected in the classical systems engineering mindset, culture, processes, and measures of success.

The classical approach to developing multisystem capabilities is through an executive oversight agency that aligns and synchronizes the development of individual systems to develop a capability that is greater than the sum of the individual systems. This approach works for SoSs that are comprised of individual systems that are being developed together as a persistent, coherent, unified whole, particularly when the identity and reason for being of the individual elements of these SoSs are primarily tied to the overarching mission of the SoS, the operational and technical requirements are relatively well known, the implementation technologies are mature, and there is a single program executive with comprehensive management and funding authority over the included systems. The classical approach to SoS development or modernization works for this category of SoS because it satisfies all the essential conditions and attributes of engineering at the system level, only it is larger. The community culture, organizational norms, rules of success, engineering and business processes, and best practices in this category of SoS either are essentially the same as at the system level or they scale appropriately.

More recently SoS development has focused on engineering of capabilities across multiple existing systems. The most common SoS capability is typically an overlay on a collection of constituent systems with the development of the SoS capability being an ongoing effort involving an ensemble of individual systems. A distinctive attribute of these SoS developments is that the SoS manager typically does not control the requirements or funding for the individual systems. At the same time, the technical complexity of SoS engineering is increasing dramatically, leading to new challenges in archi-

ture, networks, hardware and software engineering, and human–system integration. In these SoS environments, the SoS system engineers view their world and frame their activities through seven megaprocesses by drawing elements from across the 16 single-system technical and technical management processes of the Defense Department's *Defense Acquisition Guidebook*. The *Defense Acquisition Guidebook* processes are a parts box used to build the SoS megaprocesses. The executions of these processes are more or less contemporaneous, as differentiated from the serial or iterative progression of engineering processes in a single-system context.

An enterprise capability involves contributions from multiple elements, agents, or systems of the enterprise. It is generally not knowable in advance of its appearance. Technologies and their associated standards may still be emerging, and it may not be clear yet which will achieve market dominance. There may be no identifiable antecedent capability embedded in the cultural fabric of the enterprise, and thus there is a need to develop and integrate the capability into the social, institutional, and operational concepts, systems, and processes of the enterprise. Enterprise engineering is an emerging mode of systems engineering that manages and shapes forces of uncertainty to achieve enterprise capabilities through interventions instead of controls. It is directed toward enabling and achieving enterprise-level and cross-enterprise capability outcomes by building effective, efficient networks of individual systems to meet the objectives of the enterprise.

Enterprise engineering manages the inherent uncertainty and interdependence in an enterprise by coordinating, harmonizing, and integrating the engineering efforts of organizations and individuals through processes informed or inspired by evolution and economic markets. Enterprise engineering processes shape the evolution of enterprise capabilities through emergence, convergence, and efficiency phases via evolutionary or market-like mechanisms at the same time that individual system capabilities are being developed via the classical system engineering approach of building to a plan. Enterprise engineering stimulates interactions among programs of record to innovate the enterprise toward an enterprise capability at the same time the programs are developing their systems of record.

An enterprise capability is a characteristic of the enterprise in its operation. The implication is that enterprise performance will become increasingly tied to the behavior of operational units employing enterprise systems and capabilities in actual operations. Measures intended to monitor the evolution of enterprise capabilities will focus on capturing changes in the way operational units interact. The focus of success in an enterprise environment will be on demonstrated value of services that enable functionality in operational environments. Increasingly, enterprise demand for a service or offering will become its measure of value. Service-oriented architectures have the potential for shifting government acquisition away from paying for effort to make good on a contract toward a market-like economy in which contractors develop product or service offerings that compete for market share. This has

implications for shifting the balance of risk in government enterprise acquisition from being owned by the government to being shared between the government, as consumer, and contractors as producers of services that are competing for market share.

References

1. INCOSE. 2004. Systems Engineering Handbook. INCOSE-TP-2003-016-02, version 2a. INCOSE, June 2004.
2. Baldwin, K. 2007. Systems of systems: challenges for systems engineering. INCOSE SoS SE Panel, 28 June 2007.
3. Rebovich, G., Jr. 2006. Systems thinking for the enterprise: new and emerging perspectives. *Proceedings of 2006 IEEE International Conference on Systems of Systems*.
4. MITRE. 2007. Evolving Systems Engineering at MITRE. The MITRE Corporation, Bedford, MA.
5. Rogers, E. M. 2003. *Diffusion of Innovation*, 5th Edition. Free Press, New York.
6. Moore, G. A. 2002. *Crossing the Chasm*. Harper Collins, New York.
7. Rebovich, G., Jr. 2006. Systems thinking for the enterprise: a thought piece. International Conference on Complex Systems, Boston, MA, June 2006. MP 06B0025. The MITRE Corporation, Bedford, MA.
8. Rebovich, G., Jr. 2005. *Enterprise Systems Engineering Theory and Practice*, vol. 2. *Systems Thinking for the Enterprise: New and Emerging Perspectives*. MP05B043, vol. 2. The MITRE Corporation, Bedford, MA.
9. Axelrod, R. and M. D. Cohen, 2000. *Harnessing Complexity: Organizational Implications of a Scientific Frontier*. Basic Books, New York.
10. Schelling, T. C. 1978. *Micromotives and Macrobbehavior*. W. W. Norton, New York.
11. Allison, G. and P. Zelikow. 1999. *Essence of Decision: Explaining the Cuban Missile Crisis*, 2nd Edition, Addison-Wesley, Reading, MA.
12. Rebovich, G., Jr. 2007. Engineering the enterprise, Proceedings of the 2007 1st Annual IEEE Systems Conference, April 2007.
13. Kemeny, J., M. Goodman, and R. Karash. 1994. The Acme story. In *The Fifth Discipline Fieldbook*, eds. P. M. Senge et. al. Currency Doubleday, New York.
14. Flyvbjerg, B. et al. 2003. *Megaprojects and Risk: An Anatomy of Ambition*. Cambridge University Press, Cambridge.

chapter seven

Definition, classification, and methodological issues of system of systems

Marcus Bjelkemyr, Daniel T. Semere, and Bengt Lindberg

Contents

7.1	Introduction.....	191
7.2	System of systems definition and characteristics	192
7.2.1	Evolutionary behavior	193
7.2.2	Self-organization	194
7.2.3	Heterogeneity.....	194
7.2.4	Emergent behavior	194
7.2.5	Network	195
7.3	A production system as a system of systems	196
7.3.1	Definition of production system	196
7.3.2	Assessment of SoS characteristics in production systems	197
7.3.2.1	Evolutionary behavior.....	197
7.3.2.2	Self-organization.....	197
7.3.2.3	Heterogeneity	198
7.3.2.4	Emergent behavior.....	198
7.3.2.5	Network.....	198
7.4	Classification of system-of-systems types	199
7.5	System-of-systems dynamics and methodology	200
7.5.1	Purpose of design methodology	200
7.5.2	Methodology dynamics.....	201
7.6	Conclusion.....	204
	References	205

7.1 Introduction

In this chapter a generic definition of the term system of systems (SoS) is established. This definition is based both on common definitions of SoS and

on the characteristics that systems that are usually labeled SoS exhibit. Due to the inclusive nature of this generic definition, a wide range of sociotechnical systems of very different size and complexity are included in the concept of SoS. In an attempt to delimit the concept, a classification is suggested based on the redundancy of higher-level subsystems. Most systems that traditionally are considered SoS are labeled *SoS type I* (e.g., International Space Station, an integrated defense system, national power supply networks, transportation systems, larger infrastructure constructions), and systems with nonredundant subsystems are simply labeled *SoS type II*, in this chapter exemplified by a production system. The purpose of this classification is partly to advance knowledge of both SoS characteristics and how to address them, and partly to improve transferring of this knowledge from the area of traditional SoS to other areas where SoS characteristics are present.

Many of the systems that can be classified as SoS type II have traditionally been considered technical systems. Increased demands on these systems have forced requirements to interlink the adhering societal system, with a resulting increase in size and complexity, which has introduced SoS characteristics into these former technical systems. New methodologies to address SoS characteristics have, however, not been developed; instead traditional methods are still used in design and development. An introductory discussion on methodological issues in SoS is therefore also presented in this chapter.

7.2 *System of systems definition and characteristics*

The area of SoS is a fairly new research discipline without a fully established taxonomy. Several terms are used as near synonyms with slightly different definitions (e.g., complex system, engineering system, sociotechnical system, and enterprise system). In addition, SoS terms are commonly only vaguely defined. For the purpose of this chapter, both differences and similarities between these different kinds of systems are disregarded, and SoS is used as a generic term for *large and complex sociotechnical system*. This simplistic definition of SoS proves complicated when decomposed; i.e., while *sociotechnical* is fairly unambiguous, neither *large* nor *complex* are easily measured. However, its inclusive nature is useful when discussing systems that exhibit similar characteristics as SoS.

To reach an agreement of what is to be studied within the area of SoS, instantiations of large and complex sociotechnical systems have been evaluated to determine if they are large and complex enough, and if they demonstrate enough societal and technical complexity. Magee and de Weck present an extensive list of a vast range of large systems, which are classified according to societal complexity, technical complexity, and if they are natural or engineered systems [1]. From an agreed upon set of SoS instantiations, common properties, characteristics, and features have then been extracted and included as cornerstones of SoS definitions. All these generic properties of

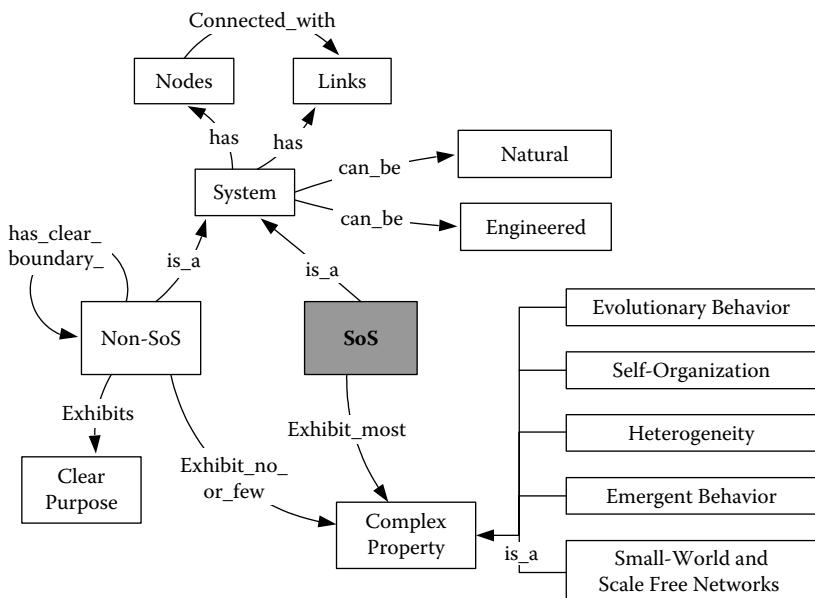


Figure 7.1 Concept model of SoS (adapted from Bjelkemyr, M., D. Semere, and B. Lindberg. 2007. Proceedings of 1st Annual IEEE Systems Conference, Honolulu, HI).

SoS are not necessarily present in all SoS, but all SoS should exhibit most of these properties [2].

The generic properties are naturally dependant on the set of SoS studied, and given the vast number of different SoS instantiations, every researcher will naturally find properties that are similar to those found by others, but not the same. Nevertheless, the following characteristics are in one or another form commonly found in most of the literature on SoS and complex system characteristics: *evolutionary behavior*, *self-organization*, *heterogeneity*, *emergent behavior*, and that SoS are small-world and scale-free networks (cf. [3,4]). Figure 7.1 illustrates these characteristics and how they are related to SoS and to other system properties.

7.2.1 Evolutionary behavior

Evolution is in this context seen as a “trial-and-error process of variation and natural selection of systems” [5]. Similar to Darwinian natural selection, the selection here is not guided by a goal of specific intent; rather, the selection is automatic and a result of the internal or external environment. Consequently, the life cycle of a non-SoS engineering system is not evolutionary; partly because it is regularly encapsulated within the stages Design–Realization–Operation–Recycling, and partly because the decisions are made by a project manager or stakeholders. In an SoS, its subsystems follow the life

cycle above, but the SoS itself is continuously and iteratively going through all stages at the same time. In addition, there are no top-level decision makers that control the decisions in an SoS. During evolution of an SoS, unlike Darwinian evolution, “a [system] configuration can be selected or eliminated independently of the presence of other configurations” as long as the configurations are not subsequent system states [5].

7.2.2 Self-organization

Self-organization is similar to evolution, but where evolution takes place primarily in a system’s interface to its environment, self-organization is an internal system process that is not “being controlled by the environment or an encompassing or otherwise external system” [6]. For SoS, self-organization is primarily decomposed into operational and managerial independence. Operational independence signifies that subsystems of an SoS are independent and useful in their own right. Managerial independence signifies that a system both is able to operate independently and actually is operating independently.

7.2.3 Heterogeneity

Complex sociotechnical systems consist of a multitude of dissimilar or diverse subsystems, structures, and agents. This *heterogeneity* is a strong driver of system complexity; i.e., a system with heterogeneous subsystems is naturally more complex than if the subsystems were homogeneous. A system is often heterogeneous on multiple layers simultaneously (e.g., size, architecture, life cycle, scientific area, and elementary dynamics). This increases the difficulty of modeling an SoS, and requires people from different knowledge and science domains to work side by side. As a result, new demands for communication and information handling are required (i.e., rules for interactions between the interfaces of all nodes in a system).

7.2.4 Emergent behavior

Emergence is the added behaviors that arise due to the interactions between its subsystems and parts, and which cannot be directly attributed to any individual subsystem or part. There are two kinds of emergence: *weak emergence*, which can be predicted by experience or extensive modeling and simulation; and *strong emergence*, in which high-level behaviors are autonomous from the systems and elements on lower levels (e.g., the autonomous relationship between neurological processes and human cognition) [7]. These two kinds are often intertwined, which creates confusion, especially regarding how emergence can be dealt with [8]. Reducing weak emergence is a substantial part of engineering work, and the engineer must always prioritize between knowledge of system behavior on one hand, and time and resources on the

other. Strong emergence, on the contrary, is not addressed in traditional design methods.

7.2.5 Network

A network is usually represented by a graph with a set of nodes that are connected by a set of edges. Both the nodes and edges can be of one or many kinds, and the connecting edges can be either directional or nondirectional (c.f. *heterogeneity*). A sociotechnical system is naturally a combination of multiple kinds of kinds of nodes and edges, and the connections include both nondirectional and directional lines. Depending on the topology of the nodes and edges in a network, different kinds of network properties emerge. These properties are completely independent of what the nodes and edges represent.

For SoS, two of the more interesting network properties are *small-world networks* and *scale-free networks*, both being common in a diverse set of social, information, technical, and biological networks [9].

In a small-world network most nodes are not directly connected to each other, but most nodes can be reached from every other in a small number of steps. This results in a dynamic system with small-world coupling that displays “enhanced signal-propagation speed, computational power, and synchronizability” [10]. In a scale-free network the number of edges of all nodes in the network follows a power-law distribution (i.e., while most nodes have few edges, some nodes are highly connected and function as hubs in the network). A consequence is that scale-free networks are fault tolerant to random failure, but at the same time they are vulnerable to a focused attack on the hubs.

Both small-world and scale-free networks are the result of three statistical network characteristics that are derived using graph theory: *average path length*, *clustering coefficient*, and *degree distribution*. *Average path length* is defined as the average least number of steps between any two nodes in a network, and it is a measure of the efficiency of a network. With a shorter average path length, fewer steps are on average required to distribute information or mass inside the network. The *clustering coefficient* is defined as the probability that two nodes, which both are connected to a third node, also are connected to each other. Consequently, the *clustering coefficient* is a measure of the network’s potential modularity [11]. The third characteristic, *degree distribution*, is the distribution of the number of edges for all the nodes in a network.

To determine if a network is a small-world, its statistical characteristics are related to those of a random network. Two criteria should be fulfilled: the network’s average path length should be similar to that of a random network, and the network’s clustering coefficient should be greater than that of a random network.

To be considered as a scale-free network, the degree distribution should follow a power law distribution (i.e., while a small number of nodes are highly connected hubs, most nodes are considerably less connected).

7.3 A production system as a system of systems

In order to evaluate if a system is to be considered an SoS, a clear definition of the properties and boundaries of the former must be established. In the case of production it is especially important, since *production* and *manufacturing* are commonly used to describe both strictly technical systems and extended enterprises. After defining production system, each SoS characteristic is addressed from a production system point of view.

7.3.1 Definition of production system

The terms *production* and *manufacturing* are commonly used as synonyms for the “processes that transform resources into useful goods and services” [12]. In more specific circumstances, one of the terms is usually reserved for “the pure act or process of actually physically making a product from its material constituents,” and the other for “all functions and activities directly contributing to the making of goods” [13]. Unfortunately, different countries and organizations have not been able to agree upon which term to use for which definition; consequently both terms can be found for both definitions.

On one hand, the International Academy of Production Engineering (CIRP) uses the term *production* as a subset of *manufacturing*, although acknowledging that *manufacturing system* is commonly used as a subset of a *production system* [13]. On the other, a *production system* is in *Encyclopedia Britannica* defined as “any of the methods used in industry to create goods and services from various resources” [12]. Moreover, many acronyms and buzz words commonly use *manufacturing* as a subset of *production*, e.g., Toyota Production system (TPS), Flexible Manufacturing System (FMS), Mass Production.

The etymology of *production* and *manufacturing* is more in line with the latter taxonomical alignment. The Latin roots of *produce* can be traced to *producere*, meaning “to bring forward,” and *manufacture* to *manu factus*, literally meaning “made by hand” [14]. For the purpose of this chapter, the definition of interest is the more inclusive one, which generally includes or affects most processes within a manufacturing company or network involved in transforming resources into useful goods and services. The term used for this definition will be *production*.

The transformed resources in a production system include labor, capital (including machines and materials, etc.), and space; these resources are also labeled “men, machines, methods, materials, and money” [12]. Consequently, design of a production system includes design of “men, machines, methods, materials, and money.” The methods used for production system design should consequently address design of these resources.

Even though the term *production system* has been defined above, its range is too inclusive for the purpose of this chapter. Therefore, a smaller set of production systems must be extracted to enable a comparison with SoS. Given the purpose, larger and more complex production systems are used as

instantiations of the concept *production system*, in particular vehicle producers (e.g., Scania, Volvo Cars, Saab).

7.3.2 Assessment of SoS characteristics in production systems

For a production system to be considered an SoS, the production system should exhibit most of the SoS properties. To determine this, each SoS property has through literary reviews and case studies been evaluated from a production system point of view.

7.3.2.1 Evolutionary behavior

Evolution of a system is closely related to the life-cycle stages of that system, and if the system's life cycle is continuous or not. While continuous systems are required to evolve to answer to changes in their environment, noncontinuous systems are designed to function within a predetermined environment and therefore seldom possess the capability to evolve.

A manufacturing system is usually designed to fulfill a number of pre-set requirements and constraints concerning, for example, product geometry, volume, variability, capacity, capability, economy, space, and time. The requirements restrict the design space to include all physically possible solutions, and the constraints further delimit that space to, for that situation, feasible design solutions. While the requirements rarely disqualify design solutions enabling evolvability, it is seldom economically realistic to design for a changing environment beyond what the requirements necessitate.

From an evolvability perspective, the design of a production system is similar to the design of a manufacturing system, with the exception that changes to the system's environment are not as easily controlled and therefore more unpredictable. Also, a production system has to a greater extent a continuous life cycle with iterative and parallel life-cycle stages. As a result, a production system must possess the ability to evolve in line with its environment to remain competitive.

The evolution of a production system is a "trial-and-error process of variation and natural selection of systems" [5], and the selection is automatic and a result of the internal or external environment. In addition, during evolution of a production system, different configurations can be selected or eliminated independently of other system configurations (e.g., outsourcing and changes to the system strategy).

7.3.2.2 Self-organization

While evolution deals with the interaction between a system and its environment, self-organization is an internal system process that is not "being controlled by the environment or an encompassing or otherwise external system" [6]. For both SoS and production systems, this property is commonly decomposed into operational independence and managerial independence.

Operational independence in a production system means that its subsystems (e.g., manufacturing system, suppliers, product developers) are independent and individually functional. Existence of this property can be demonstrated by answering if subsystems could easily be outsourced or not, i.e., if the subsystem architecture is modular or highly intertwined. For larger production systems, most first-tier subsystems are more or less modular and could be outsourced.

Managerial independence in a production system signifies that its modular subsystems are managed as individual entities that are not controlled from the top system level; i.e., production, production development, product development, et cetera are run as separate companies. This property is heavily dependent on company culture; however, top-controlled management becomes impractical with increased subsystem size and complexity.

7.3.2.3 *Heterogeneity*

A production system is made up by a wide variety of “men, machines, methods, materials, and money” [12] (e.g., managers, operators, process planners, Ph.D.s, master students, high school graduates, manufacturing system, products, computer hardware, transport system, building, process planning tools, product design tools, DFX, production control systems, raw materials, sub assemblies, salaries, product cost). These resources are simultaneously heterogeneous on multiple scales, such as size, architecture, life cycle, scientific area, and elementary dynamics. People with conflicting agendas from different science and knowledge domains are required to work side by side to enable efficient communication and collaboration.

7.3.2.4 *Emergent behavior*

When determining if a production system exhibits emergent behavior, it is imperative to define which kind of emergence. All but the simplest technical systems exhibit *weak emergence*, which can be predicted by experience or extensive modeling and simulation. So the interesting question is if a production system displays *strong emergence*, where high-level behaviors are autonomous from lower level systems and elements.

7.3.2.5 *Network*

Both small-world networks and scale-free networks have been identified in engineering problem-solving networks, e.g., development networks for vehicles, operating software, pharmaceutical facility, and a hospital facility [11]. These types of engineering problems are similar to production system design, both in size and complexity, which indicates that production system development networks are also scale-free and small-world networks.

7.4 Classification of system-of-systems types

As illustrated in the former sections, the inclusive nature of the area of SoS enables inclusion of a wide range of sociotechnical systems of very different size and complexity. In an attempt to both focus the research area of SoS and enable knowledge transferring to areas working with systems that exhibit only some of the SoS characteristics, a classification is suggested. This classification is based on neither the high-level definition nor the appended characteristics; instead, it is indirectly based on the effects of system failure and the tolerance for failure from the society. For systems that are commonly defined as SoS, failure is extremely expensive and often result in loss of key societal functions or fatalities [16]. Failure of other sociotechnical systems, e.g., production system, rarely results in any direct societal consequences. Nevertheless, for the company, workers, customers, and other stakeholders, the economical effects are often quite severe.

Based on the required fail-safe of a system, traditional SoS are often designed with multiple redundant high-level subsystems; i.e., a functional requirement is satisfied by multiple design parameters. This means that, if one subsystem fails, other subsystems will provide the functions necessary for the SoS to fulfill its purpose to a satisfactory degree. These multiple design solutions greatly increase the complexity of the system, both by the added complexity of each redundant subsystem and by the required coordination of all subsystems. Common examples of redundant SoS subsystems are the branches in an integrated defense system, i.e., Army, Navy, and Air Force. In an event where one of them is unable to complete its task, the others could reconfigure and be able to perform the main system function, even though the three are not performing exactly the same task.

For sociotechnical systems that are not required by the society to be fail-safe, redundancy is a question of cost of failure versus cost of safety measures. This becomes especially apparent for industries that compete with cost; customer must in that case be willing to pay for the additional safety. For production systems, most failures are short term and can be managed by stock or overcapacity. Yet, most production systems lack the capacity to meet long-term failure, e.g., extensive fire in a production facility.

To differentiate between these two kinds of sociotechnical system, a classification into *SoS type I* and *SoS type II* is suggested. Both type I and II are large and complex sociotechnical systems, and both types exhibit at least a majority of the presented SoS properties. The differentiating property is instead proposed to be *redundancy* of high-level subsystems. Type I systems have multiple subsystems delivering the same or similar functionality. Type II systems do not have redundant subsystems.

7.5 *System-of-systems dynamics and methodology*

The previous sections in this chapter have dealt with definitions, properties, and classification of SoS. In this chapter, the focus is shifted to how an SoS is developed, in particular methodology for system design. Methodology is here used as a set the methods and rules employed within a discipline, not the study of methods.

7.5.1 *Purpose of design methodology*

The reason for utilizing design methodologies is based on the human inability to conceptualize, relate, and remember vast amounts of information [16]. Without the support of a design methodology, this human inability will result in emergent system behavior caused by system features that were outside of the scope of the designers' ability. To come to terms with this, developers are required to use methods and tools to aid the design process.

Methods for system design can be divided into three categories: strategies, procedures, and techniques. While some methods strictly belong to one category, others are combinations of methods from one or many categories. The focus in this chapter is on holistic methods that aim to address all needs during the design phase. These holistic methods are often based on the whole design process, with amended techniques, strategies, and procedures; e.g., systems engineering, KTH-IPM [17] Design for Six Sigma [18], ModArt [19], Manufacturing System Design Framework Manual-MIT [20]. These methods try to aid the designer through the greater part of the system design phase and interrelate other common support systems wherever necessary.

Even though the need for tools and methods during system design is collectively acknowledged, many of the developed methods have shown to be one-dimensional, cumbersome, and restricting creativity [21]. Therefore, to enable widespread use these negative aspects must somehow be overcome, and system designers must feel that the return of using a system design method is greater than the required additional work.

The aforementioned holistic system design methods all address traditional systems engineering issues in a traditional manner, i.e., the design process resembles how design is executed without the aid of methods. This approach makes the transformation from an experience-based design process to a methodology-based work process rather seamless, even though more experienced personnel might find some process steps superfluous or too simplistic. A downside of keeping a traditional design process is that specific SoS properties are very difficult or even impossible to address without altering the structure of the design process. This added complexity naturally poses a problem for both developers and users of system design methodology. The result is often that holistic methods for system design are kept sequential and hierarchical to meet requirements on user-friendliness.

7.5.2 Methodology dynamics

A system's characteristics, architecture, and dynamics are examples of critical aspects that the selected set of development methods should be able to capture and preferably relate to each other. Traditional engineering methodologies are commonly centrally organized and, as a result, are not able to adequately answer to all requirements of network type systems. Instead, these systems require decentralization in which the utilities of the design objective are defined and where the corresponding decisions are made in a distributed manner, i.e., a local and objective definition and decision making.

This can be further elaborated by comparing the dynamics and properties of non-SoS and SoS and analyzing them with respect to centralized and decentralized methodologies. Both the SoS definition and the amended properties provide insight into the difference between developing an SoS (type I) and a non-SoS. In Table 7.1, the properties are considered from an engineering point-of-view, i.e., how the presence or absence of a specific SoS property affects the actual engineering work.

In Table 7.1 the differences between engineering of an SoS and a non-SoS are illustrated. For SoS type II, the engineering reality is somewhere in-between, and its characteristics, architecture, and dynamics are consequently also in-between. As a result, an SoS type II can in most cases be developed using traditional systems engineering methodology; however, traditional methodological approaches neglect or are unable to fully capture the sources of the SoS properties in a distributed network because they themselves are

Table 7.1 Comparison of Engineering of Non-SoS and SoS

Engineering of Non-SoS	Engineering of SoS (Type I)
Non-SoS are reproducible.	No two SoS are alike.
Non-SoS are realized to meet preconceived specifications.	SoS continually evolve to increase their own complexity.
Non-SoS have well-defined boundaries.	SoS have ambiguous boundaries.
Unwanted possibilities are removed during the realization of non-SoS.	New possibilities are constantly assessed for utility and feasibility in the evolution of SoS.
External agents integrate non-SoS.	SoS are self-integrating and reintegrating.
Development always ends for each instance of non-SoS realization.	SoS development never ends; SoS evolve.
Non-SoS development ends when unwanted possibilities and internal friction are removed.	SoS depend on both internal cooperation and internal competition to stimulate their evolution.

Source: Adapted from Norman, D. O. and M. L. Kurash. 2004. Engineering Complex Systems, MITRE Technical Paper, <http://www.mitre.org>.

not distributed networks. These properties will nevertheless affect the variability, and possibly result in a less robust and optimal system.

To a certain degree, methodologies inherently imply the tools and methods for analysis, decision making, and information system applicable to a particular development process. Consequently the methods and tools applicable to centralized systems may not be satisfactory for an SoS system. Centralized decision making has to be replaced by negotiation-based decision making, one design objective utility has to be replaced by multiple objective utilities, emanated from the interest of multiple individual nodes, thus allowing for simultaneous comparison at both a system and local level. For this to be achieved, methods and tools in decentralized methodologies must be simpler, and perhaps combined with simple rules to guide decisions. Nevertheless, there is a new dimension of coordination that may render decentralized methodologies more complex.

From a system development perspective, the difference in life cycles between an SoS and a non-SoS are critical. Most type I SoS and some type II SoS are developed in a continuous process, unlike the discrete development stage that is common in traditional systems engineering. In a traditional system life cycle, i.e., development, realization, operation, and disposal, each stage is executed separately from the others. For most SoS, all stages are executed concurrently and continuously, which increases the demands for intersystem collaboration.

In addition, subsystems of an SoS are self-organizing nodes in a network. Unlike non-SoS, where subsystems are monitored and managed top-down, the subsystems in an SoS are selfish and put their own interest at first. These system nodes try to maximize their own utility under the influences of and in competition with the other nodes. Self-organizing systems act very similar to humans in a society; much like a society, the longevity of an SoS is based on collaboration, and most subsystems are focused on long-term success. Methods used for SoS must therefore be able to manage negotiation and collaboration between nodes.

To illustrate the deficiencies of traditional methods and processes, development of SoS is below paralleled with high-level production engineering development processes.

Traditionally, the product development process was finalized prior to the start of the production development. This results in a lengthy time to market, especially when the production system is not able to produce the designed product. These two development stages are therefore currently often executed in parallel, so-called concurrent engineering. In the development of complex products and production systems, concurrent engineering methodology has been claimed not only to reduce time to market for products, but also to increase efficiency in the management of non-SoS complex systems. The network model of a concurrent engineering process can be thought of as a bidirectional link between the product development system and the production development system (see [Figure 7.2\(a\)](#)). Furthermore, the

development process for both the product and production can in their own right be considered as densely populated networks. Therefore, the above network can be seen as a bidirectional interaction of two networks (see Figure 7.2(b)).

Product and production are not the only two nodes involved in development; a large amount of activities are related to interactions among all stakeholders (designers, suppliers, consultants, customers, etc.). These systems must collaborate in a decentralized fashion throughout the system's life cycle to answer to the system's evolving requirements, especially during system development stages (see Figure 7.3). A concurrent design methodology must consequently be able to capture all involved networks, something which cannot be achieved in traditional concurrent engineering. More importantly

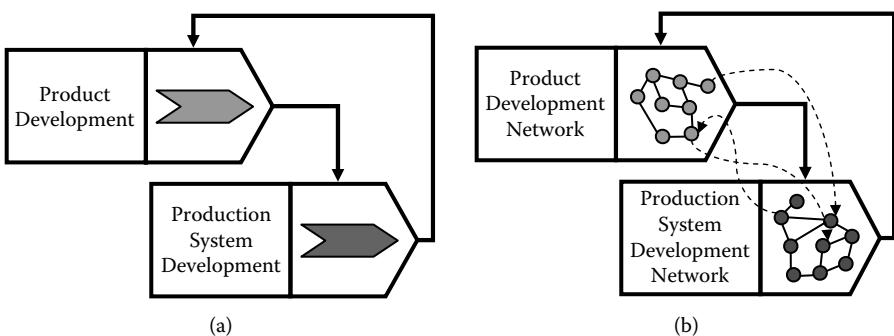


Figure 7.2 Methodological illustration of (a) concurrent engineering and (b) concurrent engineering between two development networks (adapted from Bjelkemyr, M., D. Semere, and B. Lindberg. 2007. Proceedings of 1st Annual IEEE Systems Conference, Honolulu, HI).

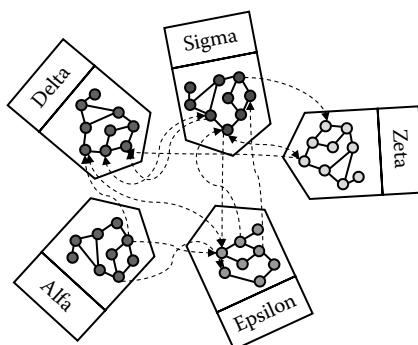


Figure 7.3 Methodological illustration of system development in a distributed network of networked nodes (adapted from Bjelkemyr, M., D. Semere, and B. Lindberg. 2007. Proceedings of 1st Annual IEEE Systems Conference, Honolulu, HI).

though, a concurrent methodology is still mainly sequential, which does not work when the system life cycle itself is iteratively evolving.

Current digital and web-based platforms and tools have transformed the interactions between all subsystems. The most obvious change is from a largely heuristic and experience-based process, to a standardized technology-centered process that does not only rely on individuals. The change from a purely problem-solving technical process into a complex social process has created new requirements for the interactions between nodes in the system. To answer to these changes, both an increased standardization of interfaces between nodes and an ability to negotiate their relationship are required. The ramification of the ability to negotiate is that the nodes in each network become more and more egocentric. With this local decision-making capability, each subsystem is able to make strategic decisions for itself, e.g., decisions regarding if it should remain in the network or leave for a new coalition.

Hence, what once could be modeled as a network of dedicated, loyal, and centrally controlled problem-solving nodes has now evolved into a coalition of autonomous and egocentric nodes which always attempt to maximize their own utilities. However, the sustainability of the SoS and, thereby, the long term profits of the subsystems themselves are dependent on realization by subsystems of the advantages of a sustainable SoS. Systems like these are often termed *collaborative networks*, *extended enterprises*, *virtual networks*, et cetera. This kind of extended network requires different tools, methods, and methodology than the ones proposed for a concurrent engineering process two decades ago.

7.6 Conclusion

The scientific area of SoS aspires to understand and improve development and operation of very large and complex sociotechnical systems. Due to the diversity of different SoS, a well-defined classification of concepts is a necessity to establish a unified area of research and, thereby, to achieve its objectives. A high-level definition has therefore been proposed and amended with generic properties that most SoS exhibit. The amended properties are, however, not enough to separate between different sociotechnical systems. Therefore, a differentiation is proposed between SoS with and without redundant high-level subsystems, respectively labeled SoS type I and SoS type II. While lesser complex systems can be developed with traditional centralized methodology, the characteristics, structure, and dynamics of distributed networks cannot be fully captured without using decentralized methodology. That is, SoS that are developed with traditional methodology do not reach their full potential.

References

1. Magee, C. and O. de Weck. 2004. Complex system classification. Proceedings from Fourteenth Annual International Symposium of the International Council on Systems Engineering—INCOSE, Toulouse, France. INCOSE.
2. Bjelkemyr, M., D. Semere, and B. Lindberg. 2007. An engineering systems perspective on system of systems methodology. Proceedings of 1st Annual IEEE Systems Conference, Honolulu, HI.
3. Maier, M. W. 1998. Architecting principles for systems-of-systems. *Systems Engineering* 1:267–284.
4. Kuras, M. L. and Y. Bar-Yam. 2003. Complex Systems and Evolutionary Engineering. AOC WS LSI Concept Paper.
5. Heylighen, F. 1997. Evolutionary theory. In *Principia Cybernetica Web*, eds. F. Heylighen, C. Joslyn, and V. Turchin. *Principia Cybernetica*, Brussels.
6. Heylighen, F. Self-organization. In: *Principia Cybernetica Web*, eds. F. Heylighen, C. Joslyn, and V. Turchin. *Principia Cybernetica*, Brussels.
7. Bedau, M. A. 1997. Weak emergence. In *Philosophical Perspectives* 11: Mind, Causation, and World. Blackwell Publishers, Oxford, pp. 375–399.
8. Johnson, C. W. 2006. What are emergent properties and how do they affect the engineering of complex systems? *Reliability Engineering and System Safety* 91:1475–1481.
9. Newman, M. 2003. The structure and function of complex networks. *SIAM Review* 45:167.
10. Watts, D. and S. Strogatz. 1998. Collective dynamics of ‘small-world’ networks, *Nature* 393:409–410.
11. Braha, D. and Y. Bar-Yam. 2004. The topology of large-scale engineering problem-solving networks. *Physical Review E*, vol. 69, 2004.
12. Encyclopedia Britannica. 2007. Production system. <http://www.britannica.com/eb/article-9106303>.
13. CIRP. 2004. *Wörterbuch der Fertigungstechnik/Dictionary of Production Engineering/Dictionnaire des Techniques de Production Mechanique* Vol. 3: *Produktionssysteme/Manufacturing Systems/Systèmes de Production*. International Academy of Production Engineering. Springer, New York.
14. Merriam-Webster. 1999. *Merriam-Webster’s Collegiate Dictionary*. Merriam-Webster.
15. Bar-Yam, Y. 2003. When systems engineering fails—toward complex systems engineering. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, vol. 2, pp. 2021–2028.
16. Bjelkemyr, M. and B. Lindberg. 2007. The effects of limits to human abilities on system-of-systems properties. Proceedings of Swedish Production Symposium, Gothenburg, Sweden.
17. Aganovic, D., M. Bjelkemyr, and B. Lindberg. 2003. Applicability of engineering design theory on manufacturing system design in the context of concurrent engineering. In *Methods and Tools for Co-operative and Integrated Design*, S. Tichkiewitch and D. Bressaud, eds. Kluwer Academic Publishers, pp. 145–158.
18. Yang, K. and B. S. El-Haik. 2003. *Design for Six Sigma: A Roadmap for Product Development*. McGraw-Hill Professional, New York.
19. ModArt. 2007. Modeldriven Artikeltillverkning. <http://modart.iip.kth.se/>. November 2007.

20. Vaughn, A., P. Fernandes, and T. Shields. 2002. Manufacturing System Design Framework Manual. Manufacturing Systems Team of the Lean Aerospace Initiative, Massachusetts Institute of Technology Cambridge, MA.
21. Fritz, S. et al. 1994. A survey of current methodologies for manufacturing system design. In Computer Integrated Production Systems and Organizations: The Human-Centred Approach, F. Schmid, S. Evans, A. W. S. Ainger, and R. J. Grieve, eds. Springer, pp. 228–253.
22. Norman, D. O. and M. L. Kuras. 2004. Engineering Complex Systems, MITRE Technical Paper, <http://www.mitre.org>.

chapter eight

Policymaking to reduce carbon emissions

an application of system- of-systems perspective

*Datu Butung Agusdinata, Lars Dittmar,
and Daniel DeLaurentis*

Contents

8.1	Introduction.....	208
8.2	Technical approach	209
8.2.1	System-of-systems (SoS) perspective.....	209
8.2.2	Policy analysis framework	210
8.2.3	Adaptive policy approach.....	211
8.2.4	Exploratory modeling and analysis (EMA)	211
8.3	Case: The Dutch residential sector.....	212
8.4	SoS specification for the residential sector	213
8.4.1	Computer Model.....	214
8.4.1.1	α Level	216
8.4.1.2	β Level.....	216
8.4.1.3	γ Level.....	217
8.4.1.4	δ Level.....	220
8.4.2	Computational experiments	221
8.5	Adaptive policy design.....	222
8.5.1	Trajectories of carbon emission reduction	222
8.5.2	Circumstances required to achieve 2025 target.....	222
8.5.3	Conditions and guidance for policy adaptation	224
8.5.4	Analysis of Case1.....	225
8.5.5	Analysis of Case2	227
8.5.6	Implications for policy design	229
8.6	Concluding remarks	229
	Acknowledgments	230
	References	230

8.1 Introduction

It has been widely acknowledged that tackling the problem of climate change entails dealing with the complex and uncertain nature of the issue [1,2]. More specifically, complexity in policymaking stems from:

a system that includes people, social structures, portions of nature, equipment and organizations; the system being studied contains so many variables, feedback loops and interactions that it is difficult to project the consequences of a policy change. Also, the alternatives are often numerous, involving mixtures of different technologies and management policies and producing multiple consequences that are difficult to anticipate, let alone to predict ([3], p. 12–13).

In addition to this, a successful climate policy increasingly requires considerations that cut across cultural and national boundaries and decisions of consequence over a generation or more [2].

Uncertainty, on the other hand, stems from incomplete knowledge and variability in the scientific, socioeconomic, and political aspects as well as the long-term time horizon involved in policymaking [4,5]. The result is a condition of deep uncertainty, in which analysts and the parties involved to a decision do not know or cannot agree upon (1) the appropriate models to describe interactions among a system's variables, (2) the probability distributions to represent uncertainty about key parameters in the models, or (3) how to value the desirability of alternative outcomes [5].

It is argued here that the notion of complexity and uncertainty are independent of each other. Although in many cases they go hand in hand, an issue that is uncertain is not necessarily complex and vice versa. As a result, handling them requires different conceptual frameworks and methods. Since the issue of climate change policy entails handling of both complexity and uncertainty, an integrative approach that addresses them both is needed. Further, an integrative approach also requires balance between domains of inquiry that are most often distinct. The field of engineering design and optimization has developed very capable methodologies for the development of complex engineering systems under uncertainty. Yet, if these are applied absent the dynamics exogenous to the engineered system (e.g., policy and economics), the solutions obtained are less effective, because the problem formulation is incomplete.

To these challenges, an eclectic number of approaches have been developed and tried out with mixed success. In dealing with complexity, professionals from the various domains are typically trained to solve problems using methods and ideas prevalent to their own domain. The engineering design paradigm is an example. This legacy is the source of the often-used

term “stovepipe” in reference to the narrow-scope thinking in a particular area of specialty knowledge. The real dynamics of the climate change issues, for example, can only be fully understood “across” stovepipes, spanning various columns of knowledge, and thus a holistic frame of reference is required for such transdomain applications. We argue in this chapter that a system-of-systems (SoS) perspective provides such frame of reference.

In dealing with uncertainty, most analyses supporting the design of policy for carbon emission reduction are still based on a “predict-then-act approach” [4]. This approach may result in a pursuit of an optimum policy based on best estimates of the states of the system and future external developments or scenarios. Even worse, it may lead to inactions when it is felt that the uncertainties are too large to warrant any policy [2]. This predict-then-act approach has been successful in conditions in which there is sufficient knowledge and information about the system of interest, usually in the form of probability distributions of relevant variables. But under conditions of deep uncertainty, it has the risk of significant prediction errors, which may lead the chosen policy to fail.

This chapter presents an SoS-oriented approach that addresses complexity and uncertainty for policymaking. The approach is described first, followed by description of its application to a case in the Dutch residential sector. The application illustrates how the approach supports a design of an adaptive policy. The chapter concludes with implications of the approach to policy design.

8.2 Technical approach

We present an approach to deal with complexity and uncertainty that combines several methods and perspectives. This mixture includes a system-of-system perspective for dealing with the complexity and an adaptive policy approach supported by exploratory modeling and analysis for dealing with uncertainty. On top of that, a policy analysis framework is used to conceptually model policymaking.

8.2.1 System-of-systems (SoS) perspective

To be able to provide a basis for abstraction and conceptualization of SoS for policymaking purposes, a lexicon has been developed. DeLaurentis and Callaway [6] define a lexicon in terms of levels and categories, as shown in [Table 8.1](#). The categories highlight the presence of a heterogeneous mix of engineered and sentient systems together constituting the dimensions of the problem. For each category, there is a hierarchy of components. To avoid confusion, the lexicon employs the unambiguous use of Greek symbols to establish the hierarchy. Alpha (α), beta (β), gamma (γ), and delta (δ) indicate the relative position within each category. The collection of α entities and their connectivity determines the construct of a β -level network, and likewise, a

Table 8.1 SoS Lexicon

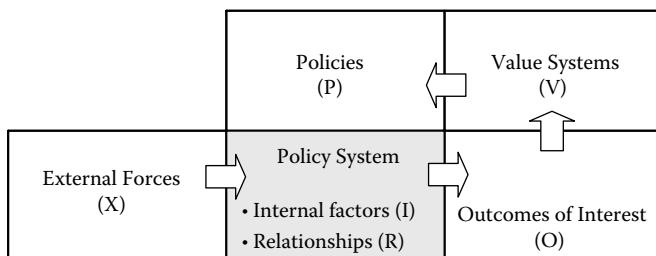
Category	Description
Resources	The entities (systems) that give physical manifestation to the system of systems
Stakeholders	The individual/organizational entities that give intent to the SoS through values
Operations	The application of intent to direct the activity of entity networks
Policies	The functions that guide the operation of resource and stakeholder entities
Level	Description
Alpha (α)	Base level of entities in each category, further decomposition will not take place
Beta (β)	Collections of β -level systems (across categories), organized in a network
Gamma (γ)	Collections of γ -level systems (across categories), organized in a network
Delta (δ)	Collections of δ -level systems (across categories), organized in a network

γ -level network is an organized set of β networks. Hence, the δ -level can be described as a network comprised of all of the lower-level networks, whose constituents span the category dimensions.

The aspiration is that, through use of the lexicon for understanding the multilevel relationships, decisions of one stakeholder may be appropriately tailored in cognizance of the actions of others.

8.2.2 Policy analysis framework

The above lexicon is operationalized and tailored for policymaking purpose. In this regard, we view that policymaking is concerned with making choices regarding a system in order to change the system outcomes in a desired

**Figure 8.1** Policy analysis framework with XPIROV structure.

way [3]. The elements from this framework can be assembled in a structure labeled “XPIROV” (see [Figure 8.1](#)), where:

- X = External forces: factors that are beyond the influence of policymakers (i.e., exogenous).
- P = Policies: instruments that are used by policymakers to influence the behavior of the system to help achieve their objectives.
- I = Internal factors: factors inside the system (i.e., endogenous) that are influenced by external forces and policies.
- R = Relationships: the functional, behavioral, or causal linkages among the external forces, policies, and internal factors that produce the outcomes of interest.
- O = Outcomes of interest: measurable system outcomes that are related to the objectives of policymakers and stakeholders. Hence,

$$O = R(X, I, P) \quad (8.1)$$

- V = Value system of policymakers and stakeholders, which reflects their goals, objectives, and preferences. The value system contains the criteria for indicating the desirability of the various policy outcomes based on the resulting outcomes of interest.

As the policy analysis framework is operationalized within the SoS perspective, the result is a network of interdependence policy systems across various levels (see [Figure 8.3](#)).

8.2.3 Adaptive policy approach

In contrast to the predict-then-act approach, the adaptive approach can be characterized in several ways [7]. First, it does not require that all uncertainties are resolved before a policy can be designed and implemented. Second, it also makes explicit the aspect of learning in resolving the uncertainties. Third, it uses a monitoring system with triggers that call for policy adaptations.

It has been demonstrated, for example, that adaptive policy can correct and therefore avoid the cost as a result of the failure of optimal policy [8].

8.2.4 Exploratory modeling and analysis (EMA)

To support the design of an adaptive policy, exploratory modeling and analysis (EMA) method is employed. EMA helps to reason about long-term system behavior by exploring as broad assumptions about policy system representation as it is useful and resources allow [9]. It is particularly suitable in the conditions under deep uncertainty in which actors involved in a decision do not know or cannot agree on appropriate system representations. These cover, among other things, uncertainty in the external scenarios, model parameters,

and structural uncertainty. Under this condition, no single model can appropriately represent the system. However, even such “bad” or “wrong” models can aid in learning, reasoning, and creating mental models about system behavior [10,11].

The key for performing exploratory modeling is computational experiments. One computer run can be considered as one hypothesis about the system manifestation. Each run will use one unique set of system representations. Under each system representation, the performances of a set of policies are calculated. In this way, exploratory modeling treats one hypothesis of system representation and its policy consequences as one deterministic hypothesis, that is, how the system will behave for a particular policy. This is an important distinctive feature of the method, because rather than trying to predict system behavior, it reasons about the behavior, asking what would happen if the guess were right. With a large number of hypotheses one can have a feel of the possible range of system outcomes.

We now illustrate the application of the approach on policy measures to reduce carbon emissions in the case of the Dutch residential sector.

8.3 Case: The Dutch residential sector

In the Netherlands, residential energy use accounts for 17% of the national final energy consumption in the year 2000 (Figure 8.2(a)). The Netherlands

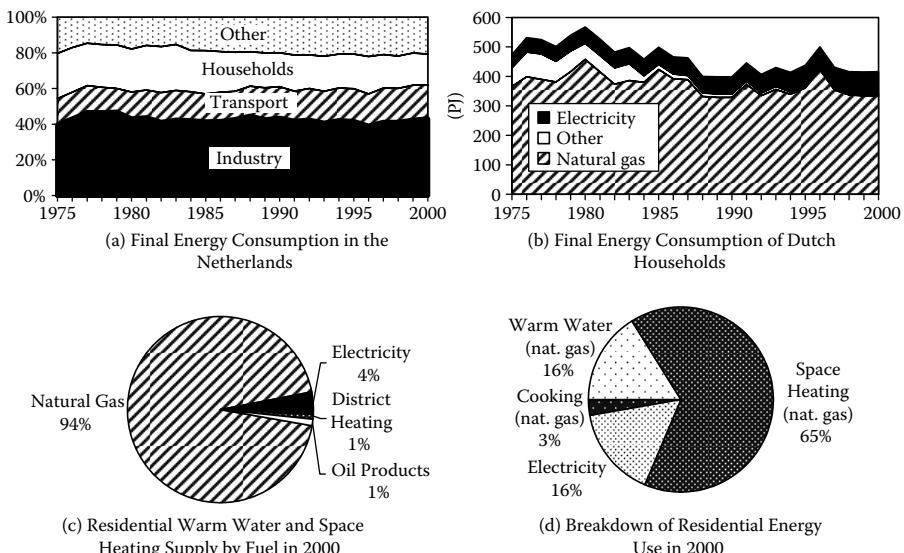


Figure 8.2 Structural indicators of residential energy use in the Netherlands: (a) Final energy consumption by sector [13]. (b) Final energy consumption of Dutch households [13]. (c) Residential warm water and space heating supply by fuel in 2000 [14]. (d) Breakdown of residential natural gas use in 2000 [15].

Table 8.2 Temperature Corrected Direct CO₂ Emissions of Dutch Households [16]

	1990	1995	2000
Residential CO ₂ Emissions [Mton]	22,2	21,9	21,4
Index 1990 = 100	100%	99%	96%

has the second-largest natural gas reserves in Europe, the highest level of natural gas penetration worldwide, and nearly all Dutch households (ca. 97%) are connected to the natural gas grid [12]. As consequence, residential space and water heating demand is met almost exclusively by natural gas, followed by electricity and district heat (see [Figure 8.2\(c\)](#)).

Virtually every Dutch household uses natural gas for space heating, warm water production, and in small proportions for cooking (see [Figure 8.2c,d](#)). Together, these energy demand categories account for about 84% of household final energy consumption, while 16% of final energy consumption can be attributed to electricity used for lighting and appliances (see [Figure 8.2d](#)). Direct residential CO₂ emissions contribute 11% to the total energy-related CO₂ emissions of the Netherlands (Table 8.2) [16].

Policymakers have long recognized the potential contribution the residential sector can make to reducing carbon dioxide emissions. In fact, since the first oil crises in 1973, the residential sector was the subject of a vast number of energy conservation efforts and, more recently, carbon restraint policies [17]. Average energy consumption for space heating and hot water production fell from about 100 GJ of natural gas per year and dwelling in the late 1970s to 67 GJ in 2000 [15] at approximately the same average dwelling area [18]. In spite of these efforts, various studies indicate that the potential for energy efficiency improvements within the built environmental is still enormous, especially in the existing building stock [19,20]. Furthermore, the Dutch government has established the policy ambition to reduce carbon emission by 60%–80% in 2050 compared to the 1990 level [21]. In the meantime, the European Commission has also issued an intermediate target of reducing carbon emissions in 2020 by 20% compared to the 1990 level [22]. The residential sector is considered to be one of the keys to reach these ambitious targets. However, since the energy-saving potential for the existing dwelling stock is much larger than that related to the stricter building codes for new homes, there is a need of policies that do not address new construction but also particularly address the energy-saving potential in the existing stock of dwellings. Amongst others, subsidy schemes for energy refurbishment of existing dwellings provide one such policy measure.

8.4 SoS specification for the residential sector

For each SoS level (denoted by Greek symbols) and from both the supply and demand side, the relevant policy systems are specified by XPIROV elements

(see [Figure 8.3](#)). External forces (X) and policies (P) act upon the policy system, whose structure is defined by its internal factors (I) and relationships (R). The interactions among such factors result in the outcome of interest (O), whose desirability is determined by decision makers' value system (V). This specification is based on the work of Agusdinata [23].

For the supply and demand side, each policy system across the SoS levels is briefly described as follows:

- α level—At the supply side, the policy system is the heating technology innovation system, in which private and public research institutions attempt to produce more efficient heating technology at affordable cost. At the demand side, the policy system is the dwelling thermal envelope system, which comprises the state of heating installations and thermal insulation.
- β level—At the demand side is the heat consumption system in which house occupants decide on, depending on their life style, how they consume energy for heating. The level of energy consumption, which results in energy costs that occupants have to pay, depend also on, among others, weather conditions, energy prices, and to a certain extent their disposable income. For the supply side no relevant policy system is considered.
- γ level—At the supply side is the heating infrastructure system, in which network operators invest on the infrastructure network. At the demand side is the dwelling investment system, in which housing developers and home owners invest on the state of housing stocks. For both policy systems, investment decisions are made based on profitability criteria (e.g., net present value (NPV) and payback period). Investment decisions on energy cost-saving technology are based on various factors such as energy prices and incentives given to them (e.g., subsidy).
- δ level—At the supply side is the renewable energy system in which policies are designed to influence the increase of renewables in the energy supply. At the demand side is the built-environment system in which the main concern is the level of carbon emission and the associated cost to achieve the policy goal of reducing the emission level. Both policy systems have the Dutch government as controlling actor. To influence the performance of the SoS, they can use policy instrument such as subsidy and regulation on building code.

From the SoS perspective, each level requires a unique set of policies. As a whole, the combinations of policies and decisions may form a concerted effort to influence the outcome (i.e., carbon emission level).

8.4.1 Computer Model

For our computational experiments we made use of the Dutch Residential Energy Model (DREM) [24]. DREM is an integrated, dynamic model that

Level	Supply side			Demand side		
α		<ul style="list-style-type: none"> • Efforts in innovation • Commercial value 				
	<ul style="list-style-type: none"> • Incentives 	<p><i>Innovation (R&D) System</i></p> <ul style="list-style-type: none"> • Learning factor 	<ul style="list-style-type: none"> • Efficiency and costs of Heating generator technology 	<ul style="list-style-type: none"> • Rate of dwelling retrofit 	<p><i>Dwelling Thermal Envelope System</i></p> <ul style="list-style-type: none"> • Heating installation • State of thermal insulation 	<ul style="list-style-type: none"> • Dwelling heating efficiency
				<ul style="list-style-type: none"> • Life style 	<ul style="list-style-type: none"> • Individualistic and social values 	
				<ul style="list-style-type: none"> • Weather conditions • Incentives • Energy prices • Disposable income 	<p><i>Heat Consumption System</i></p> <ul style="list-style-type: none"> • Energy demand for heating 	<ul style="list-style-type: none"> • Energy costs
γ		<ul style="list-style-type: none"> • Investment on network accessibility • Profitability 			<ul style="list-style-type: none"> • Investment in dwelling retrofit • Profitability and social responsibility 	
	<ul style="list-style-type: none"> • Regulations 	<p><i>Dutch Heating Infrastructure System</i></p> <ul style="list-style-type: none"> • States of network 	<ul style="list-style-type: none"> • Net present value (NPV) 	<ul style="list-style-type: none"> • Performance standards • Incentives 	<p><i>Dwelling Investment System</i></p> <ul style="list-style-type: none"> • Dwelling stock 	<ul style="list-style-type: none"> • Payback period
δ		<ul style="list-style-type: none"> • Incentives and Regulations for development of renewables • Protection of environment 			<ul style="list-style-type: none"> • Incentives and regulations for energy savings • Energy security • Protection of environment 	
	<ul style="list-style-type: none"> Renewables access to grid 	<p><i>Dutch Renewable Energy System</i></p> <ul style="list-style-type: none"> • Stock of renewables 	<ul style="list-style-type: none"> • Share of renewables 	<ul style="list-style-type: none"> • Energy consumption • No. population • Household size 	<p><i>Dutch Built Environment System</i></p> <ul style="list-style-type: none"> • Emission factor 	<ul style="list-style-type: none"> • Carbon emissions • Policy costs

Figure 8.3 Specification of SoS for policymaking to reduce carbon emissions in the residential sector.

simulates changes to the dwelling stock and associated environmental impact in terms of carbon dioxide emissions. The model spans a simulation horizon of 50 years, from 2000 to the year 2050, discretized into 5-year time segments. DREM is a vintage stock model, dividing the total stock of capital (dwellings and heating equipment) into vintages, based on their period of construction. The different vintages each embody the techno-economical properties of the time in which they were built, staying attached until they are retired or refurbished. DREM simulates the evolution of the residential capital stock over time through retirements, new construction, refurbishment of existing dwellings, and initial as well as replacement purchases of heating equipment [24]. The modeling approach is essentially based on bottom-up principles, explicitly simulating the thermal performance of a set of 26 representative dwelling types and the penetration and impact of different heating systems providing domestic hot water and space heat. Altogether DREM provides a framework for systematically exploring future energy use and associated carbon emissions from the Dutch dwelling stock across a variety of uncertainties such as demographics, investment behavior, fuel prices, energy policy, and technological development.

The SoS framework introduced above can be found back in the model structure in the form of functional relationships and system variables. The specification of the uncertainty space to be explored is represented through a range of system variables, which are described below (see [Table 8.3](#)):

8.4.1.1 α Level

On the supply side, technological devolvement is modeled by the well-known concept of technological experience curves [25,26], describing cost reductions of technologies as a function of accumulated experience in form of units installed.

$$Inv(k, t) = I_0(k) \cdot CU(k, t_{-1})^{-LR} \quad (8.2)$$

where I_0 is the initial investment cost, CU is the experience in form of cumulative installed units, and LR is the learning rate. Uncertainties are expressed through a range of learning rates (LR), which measures the steepness of the experience curve. Large values of LR indicate a steep curve with a high learning rate. Technologies considered for learning are solar thermal systems, electric-driven heat pumps, gas driven heat pumps, and micro-CHP applications.

8.4.1.2 β Level

Energy price uncertainties are captured in a range of annual price growth multipliers (G for natural gas and E for electricity), which are divided over two periods: 2000–2025 and 2025–2050. The changes of energy price are modeled by a compound growth.

Table 8.3 Specification of System Variables and Their Plausible Range

Level	System Variable	Description and Symbol	Value Range
α	Learning rates for four technologies	LR1: electricity-driven heat pump LR2: solar thermal heating systems LR3: micro-CHP LR4: gas-driven heat pumps	[10%, 20%]
β	Annual growth of gas price	G1 (2000–2025)	[2%, 4%]
		G2 (2025–2050)	[2%, 6%]
	Annual growth of electricity price	E1 (2000–2025) E2 (2025–2050)	[1%, 3%] [1%, 5%]
γ	Discount rate	DR	[5%, 20%]
	Parameter lambda technology	LT	[2, 12]
	Demolition rate	RET	[0.4]
	Acceptable payback period	PB	[5, 10] years
δ	Parameter lambda refurbishment	LF	[2, 6]
	Subsidy level	SUB1 (2000–2025) SUB2 (2025–2050)	20 €/ton CO ₂ 20 and 50 €/ton CO ₂
	Standard for building energy performance	CODE	[2.5%, 5%]
	Growth rate of household size	HZ	[-0.4%, -0.2%]
	Rate of population growth	POP	[-0.5%, 0.5%]

8.4.1.3 γ Level

Market shares of competing heating technologies are simulated by a logit market sharing function [27,28], which assumes that energy users face a set of heating technologies following a Weibull cost price distribution with a shape parameter (LT), common to all technologies. The market share of a technology is equal to the probability that this technology shows lower cost than any of the competing options. The closed form of the logit sharing function is given by:

$$MS(k') = \frac{\bar{c}(k')^{-LT}}{\sum_k^n \bar{c}(k)^{-LT}} \quad (8.3)$$

where MS (k') is the market share of technology k', $\bar{c}(k)$ are the "intrinsic costs" [28] of the k-th technology, and LT is Weibull shape parameter. The

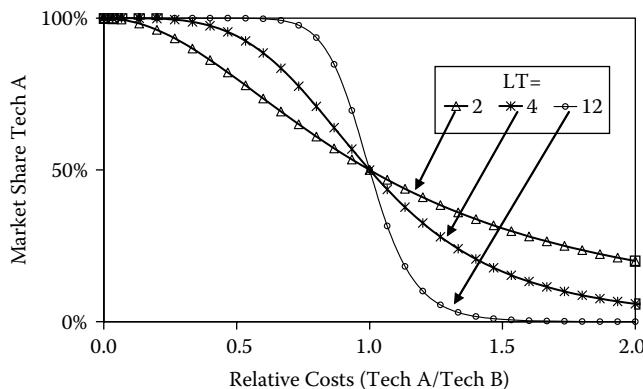


Figure 8.4 Market share function based on parameter LT.

Weibull distribution's shape parameter measures the variance in the market; a low LT implies that technology market shares are distributed relatively uniformly among all competing technologies, even if their costs differ significantly. For high values of LT, on the other hand, the lowest cost technologies gain proportionately higher market shares (see Figure 8.4).

The logit function allocates market shares based on the annual cost equivalents of energy supply, which include annualized investment costs, operation and maintenance costs, fuel costs for space heating and domestic hot water, and finally, the benefits from cogeneration technologies in the form of avoided electricity purchase and sale of surplus electricity. We consider the time preference of capital investment in the form of discount rate (DR), which is used to annualize the investment costs.

We model four energy efficiency refurbishments, viz. improved thermal performance of roof, walls, floor, and glazing. For each dwelling component, four standards, progressively improving the energetic performance of the respective dwelling components, compete for market share. The different standards are given in Table 8.4 in terms of their u-values. The u-value is the measure of the rate of heat loss through a component, the lower the u-value the lower the heat losses.

Table 8.4 u-Values of Refurbishment Options

Component (i) [W/m ² .K]	Standard (st)			
	1	2	3	4
Windows	2.0	1.60	1.20	0.70
Walls	0.77	0.4	0.33	0.25
Roof	0.40	0.33	0.25	0.20
Floor	0.50	0.40	0.33	0.25

DREM links the logit approach to a so called payback acceptance curve (PAC) in order to simulate both the rate of refurbished each period and the market share of the different insulation standards endogenously. The PAC describes the percentage of consumers that would adopt a refurbishment standard if it provided an acceptable critical payback. The simple payback time is given by:

$$PB(i, st) = \frac{\Delta I(i, st)}{\Delta E(i, st) \cdot P_e} \quad (8.4)$$

where $PB(i, st)$ = simple payback time of refurbishment standard st for dwelling component i , ΔI = additional investment costs of the refurbishment standard st , ΔE = resulting energy savings, and P_e = energy price.

The logit function allocates market shares based on the comparison of the paybacks, implying that the market share of a particular refurbishment standard is equal to the probability that this standard shows lower payback than any of the competing ones.

$$MS(st') = \frac{PB(st')^{-LF}}{\sum_{st}^n PB(st)^{-LF}} \quad (8.5)$$

where $MS(st')$ is the market share of standard st' , $PB(st)$ is the “intrinsic payback” of the standard st , and LF is Weibull shape parameter. The Weibull distribution’s shape parameter, LF , represents the diversity in the market. For high values of LF , the standard with lowest payback gains highest market shares, while for low values of LF , market shares are relatively evenly distributed (compare also the above explanations of Equation 8.5 and see [Figure 8.4](#)).

We use the mean of the minimum payback distribution, obtained from the logit, and estimate therewith the rate of refurbishment in a PAC. The mean of the minimum payback distribution is given as [27,28]:

$$PB_{AV}(i) = \left[\sum_{st} PB(i, st)^{-LF} \right]^{-\frac{1}{LF}} \quad (8.6)$$

where is PB_{AV} is the mean payback of component i across standards st , PB is the intrinsic payback of the standard st , and LF is the Weibull distribution’s shape parameter.

The PAC is assumed to be a standard logistic function:

$$RR(i) = RR_{\max}(i) \cdot \left\{ 1 - \frac{1}{1 + e^{-(PB_{AV}(i) - PB)}} \right\} \quad (8.7)$$

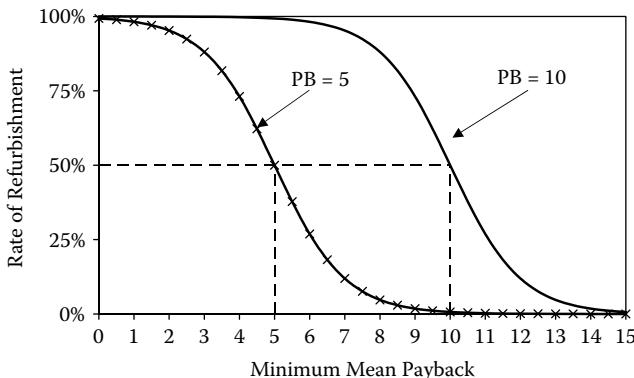


Figure 8.5 Sample payback acceptance curve.

where RR = rate of refurbishment of dwelling component i , RR_{\max} = technical refurbishment potential of component i , PB_{AV} = minimum mean payback, and PB = acceptable payback. For each component, the rate of refurbishment represents the technical potential multiplied by the share of customers that would accept the payback. The PAC is defined such that in 50% of the cases refurbishment will be undertaken, subject to the condition that the payback period equals the acceptable value (PB) (see Figure 8.5).

Finally, uncertainty about the rate of housing demolition (RET) is explored with five empirically established survival functions of Dutch dwelling stock, describing the demolition of dwellings as a function of their age [29–31]. These survival functions differ in the rate of dwelling demolition: for $RET = 0$ the demolition rate is the highest, and for $RET = 4$ the lowest.

8.4.1.4 δ Level

We test two subsidy arrangements to stimulate energy efficiency refurbishments in housing buildings. The first one is to employ a 20 € subsidy per ton CO₂ avoided for the whole period of 2000–2050 (i.e., SUB1 = SUB2 = 20). The second design is to provide a 20 € subsidy for the 2000–2025 period and a 50 €/tCO₂ for 2025–2050 (i.e., SUB1 = 20, SUB2 = 50). These subsidies directly influence the payback time of the respective refurbishment measures, i.e.:

$$PB(i, st) = \frac{\Delta I(i, st) - SUB \cdot \Delta CO_2(i, st)}{\Delta E(i, st) \cdot P_e} \quad (8.8)$$

where $PB(i, st)$ = simple payback time of refurbishment standard st for dwelling component i , SUB = subsidies, ΔCO_2 = CO₂ emissions avoided over the lifetime of the insulation standard st , ΔI = additional investment costs of the refurbishment standard st , ΔE = resulting energy savings, and P_e = energy price. As can be seen by inspection of Equation 8.6, increasing subsidy

decreases the payback time, which in turn leads to higher investments and hence to higher CO₂ reductions.

Furthermore, different regimes of building codes (CODE) are imposed on the construction of new dwellings. Tightening of building codes in the period 1996–2006 caused a reduction of energy use for space heating and hot water production for new dwellings of about 5% per year on average [32]. We assume a range of 2.5% to 5% of average annual reductions realized in new dwellings as result of different scenarios of future building codes. Finally, occupation density of dwellings is explored by the lifestyle variable household average size (HZ), and uncertainty about the future activity of the sector by a population growth multiplier (POP).

8.4.2 Computational experiments

Computational experiments are carried out across the uncertainty space defined in the previous section. We use CARs (Computer Assisted Reasoning System) Software to perform the experiments [33]. The software treats a model (in this case a spreadsheet model) as a black box and maps model inputs to outputs, creating a database of model runs. A set of inputs is created by Latin-Hypercube sampling (e.g., [34]). We take 50,000 samples of data sets across the uncertainty space over the 50-year period. Figure 8.6 illustrates the results of these model runs in terms of a “scenario funnel,” capturing the entire solution space generated. Additionally, some randomly chosen realization pathways are plotted.

We derive the pattern of behavior on which the policy adaptation is based, using a pattern recognition process called Classification and Regression Tree (CART) (e.g., [35]). It applies a nonparametric classification algorithm, which consists of a sequence of binary split mechanism, to the database of model runs. As a result, we obtain a classification tree of input variables with end

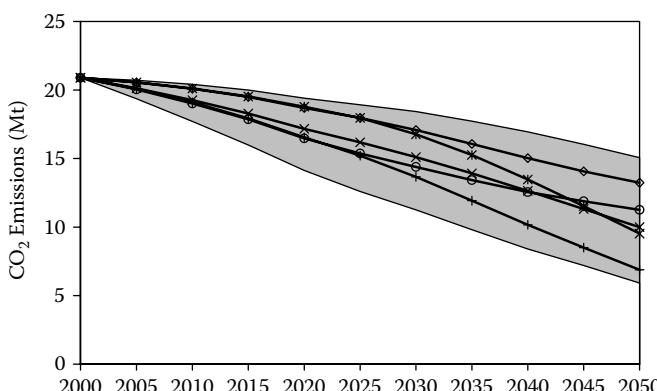


Figure 8.6 Scenario funnel of future residential CO₂ emissions from space heat and warm water production.

nodes of categories of model outputs (e.g., category of emission level). On the input variables, each level of split determines the importance of the variables in the classification process; the higher the hierarchy of the split, the more important the variable is in influencing the output variable.

8.5 Adaptive policy design

8.5.1 Trajectories of carbon emission reduction

Based on the policy target information in Section 8.3, we establish several trajectories for carbon emission reduction resulting from energy used for space heating and warm water production in the residential sector (see Figure 8.7). As a policy target, the carbon emission level should be reduced to around 15 Mton in 2025, which corresponds to a reduction of about 30% compared to 1990 levels. Furthermore, the emission should be brought down to around 10 Mton in 2050 (55% reduction). A policy can be considered as a success when the emission level in 2050 is less than or equal to 10 Mton.

In order to illustrate the adaptive policy approach, we run a hypothetical case that is divided into two parts. The first part covers the 2000–2025 period in which policies are implemented under uncertainties. The second part covers the 2025–2050 period in which some of the uncertainties are resolved and policies are adapted, attempting to meet the policy target.

8.5.2 Circumstances required to achieve 2025 target

For the first part, the amount of subsidy allocated to the 2000–2025 period is 20 €/ton CO₂ avoided. To be able to reach the target of around 15 Mton in 2025, certain circumstances must materialize. We put a query to the database

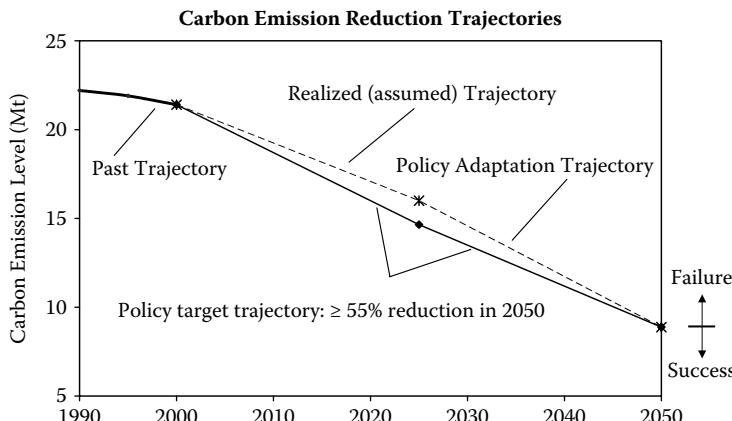


Figure 8.7 Assumed trajectories of carbon emission reduction.

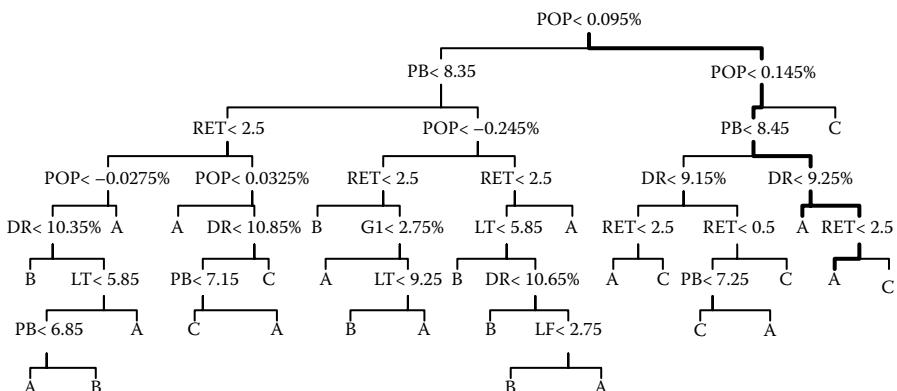


Figure 8.8 Circumstances required to achieve 15 Mton policy target in 2025. Cross-validation error = 44.7% (min. error = 39%). Legend: category A: emission target 2025; category B: < target; category C: > target; G1: growth rate gas price (2000–2025); LF: lambda refurbishment; PB: payback threshold; POP: population growth; LT: lambda technology; DR: discount rate; RET: demolition rate function.

of model runs to reveal such circumstances. The query for the 2025 emission level is set in the range of 14.8–15.2 Mton (category A), emission level greater than 15.2 (category B), and below 14.8 Mton (category C).

On the resulting (smaller) dataset, we employ a classification tree algorithm that reveals multiple sets of variables that meet the query criteria (see Figure 8.8). In CART diagram, the splitting variables are shown with their condition. When the condition is met, the path continues to take the left way, otherwise to the right of the tree. For example, taking the right-hand side paths (highlighted in bold lines), the emission target of 2025 can be achieved when the population growth (POP) is between 0.095% and 0.145%, payback threshold (PB) above 8.45, and discount rate (DR) lower than 9.25%. In case that DR is higher than 9.25%, then the demolition function (RET) should be the ones with faster rate (RET = 0, 1, and 2), implying that more old and less-energy-efficient buildings need to be demolished.

It is important to note that the application of CART on a large data set (50,000 in this case) produces a complex tree with many branches and terminal nodes (up to 4000 nodes). Such a complex tree is very good at classifying existing data but can be poor in classifying new data. So we employ a criterion of minimum cross-validation error to prune the tree ([36]). The result is a smaller tree that is used to model system behavior. For illustrative purpose, Figure 8.5 shows a tree with a classification error of 44.7%, meaning that the CART at this pruning level will misclassify almost 45% of the data (min. error is 39%). In this case, the precision is sacrificed to gain a broad understanding about system behavior under uncertainty.

8.5.3 Conditions and guidance for policy adaptation

Suppose, however, that the circumstances are such that the target of 15 Mton in 2025 cannot be met. Instead, it goes off the target at 16 Mton. Such a path, which coincides with a long-term scenario study recently carried out in the Netherlands [37], is depicted as the “realized (assumed) trajectory” in Figure 8.7. A query to the database of model runs is performed to reveal circumstances that lead to an emission level of 15.5–16.5 Mton. We also assume that, during the 2000–2025 period, learning and monitoring of the SoS variables takes place (beyond the scope of this chapter). Such learning enables one to know the realization of the variables, or to reduce the uncertainty (i.e., a smaller variable range).

The question at this point is, having missed the target of 2025, what conditions are required to bring the emission level to meet the target for 2050 (i.e., the “policy adaptation trajectory” in Figure 8.7)? To address this problem, one needs to deal with the path dependency within the SoS, since what can be achieved in 2050 is more or less constrained by the performance in the first 25 years (2000–2025).

To see the nature of such path dependency, two cases of subsidy arrangement are tested. Case1 is when the 20 €/ton CO₂ level of subsidy is maintained for the period 2025–2050; Case2 is when the subsidy level is increased to 50 €/ton CO₂. Using the resulting dataset of 18,596 samples for each case, we set the range of 2050 emissions level to 10 Mton and below (category S = policy success) and emissions level above 10 Mton (category F = policy failure). The CART algorithm is then employed to the data set. The results are shown in Figure 8.9 (Case1) and Figure 8.11 (Case2).

These resulting CART in the form of “if-then” rules become the basis to inform the design of adaptive policy. As a whole, CART shows the different

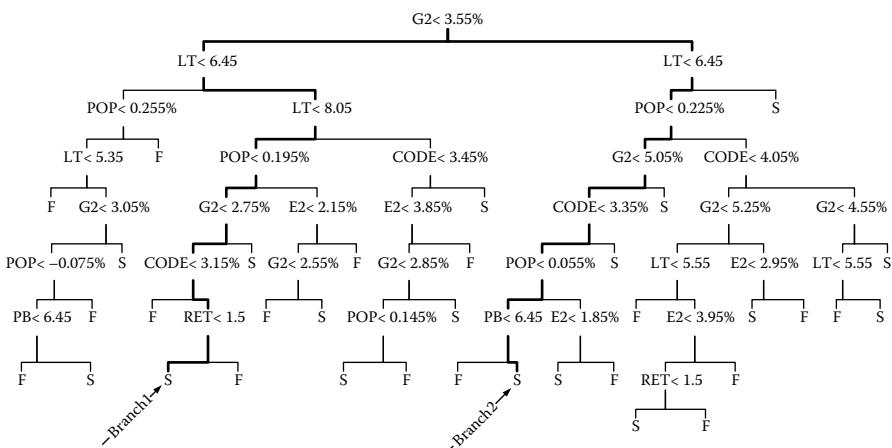


Figure 8.9 Case1 CART for policy adaptation. Minimum cross-validation error = 20.7%. Legend: category S: policy success; category F: policy failure.

pathways that lead to emission level category. Each branch describes one set of circumstances, which differs from other branches, and therefore may shed light on the trade-offs among variables. The trade-off analysis for the two cases is described below.

8.5.4 Analysis of Case1

The CART for Case1 in [Figure 8.9](#) shows different conditions that are required for policy adaptation. In total, there are about 12,800 realizations of SoS variables (almost 70% of that leads to policy success). Because of the reduced uncertainty, the misclassification error is around 20%. A subset of these successful scenarios is represented by each of the CART branches leading to category S. For illustration, two branches from Case1 CART (pointing arrows in [Figure 8.9](#)) will be further analyzed and compared. The idea is to illustrate the trade-off when the growth of gas price is below (Branch1) and above 3.55% (Branch2). In addition, Branch1 involves a distinct splitting variable demolition function (RET), while Branch2 involves payback threshold (PB).

Following the “if-then” rules given by each of the branches, the applicable ranges of all SoS variables are then reconstructed using box plots (see [Figure 8.10](#)). Invented by Tukey [38], a box plot provides a summary of statistical information about smallest observation, lower quartile (Q1), median, upper quartile (Q2), and largest observation. It indicates variance, skew, and outliers without any assumption of probability distribution.

At each SoS level, the system variables are given. The range in the box plot y-axis corresponds to the low and high values set in the specification of system variables specified in [Table 8.3](#). The bar shows the applicable range in which one system variable, in combination with the range of all others, should materialize in order to achieve the policy target.

At the α level, there seems to be little difference regarding the realizations of learning rates of the four technologies considered. In the two branches, Branch1 and Branch2, the lowest and the highest realizations are similar, covering almost all the plausible range defined in Table 8.3. In Branch2, however, the learning rate for micro-CHP (LR3), is allowed to be extended to a lower value, since the first quartile is now somewhere below 12%.

At the β level, the difference in requirements for the growth of gas price (G2) is apparent. As the first split of the CART indicates, in Branch1 the G2 occupies the value range below 3.55%. On the other hand, little impact is felt for the realizations of the growth of electricity price (E2).

At the γ level, one consequence of the fact that G2 realization is lower in Branch1 is that the value of LT (higher LT represents greater market share for cost-effective heating technologies) has to be higher compared to the one in Branch2. What also remarkable is, with low G2, the demolition rate (RET, given in percentage count of the dataset) in Branch1 has to be higher than that in Branch2. In the former, the demolition rate is restricted to only RET = 0 and 1 (faster rate), while in the latter it is not restricted.

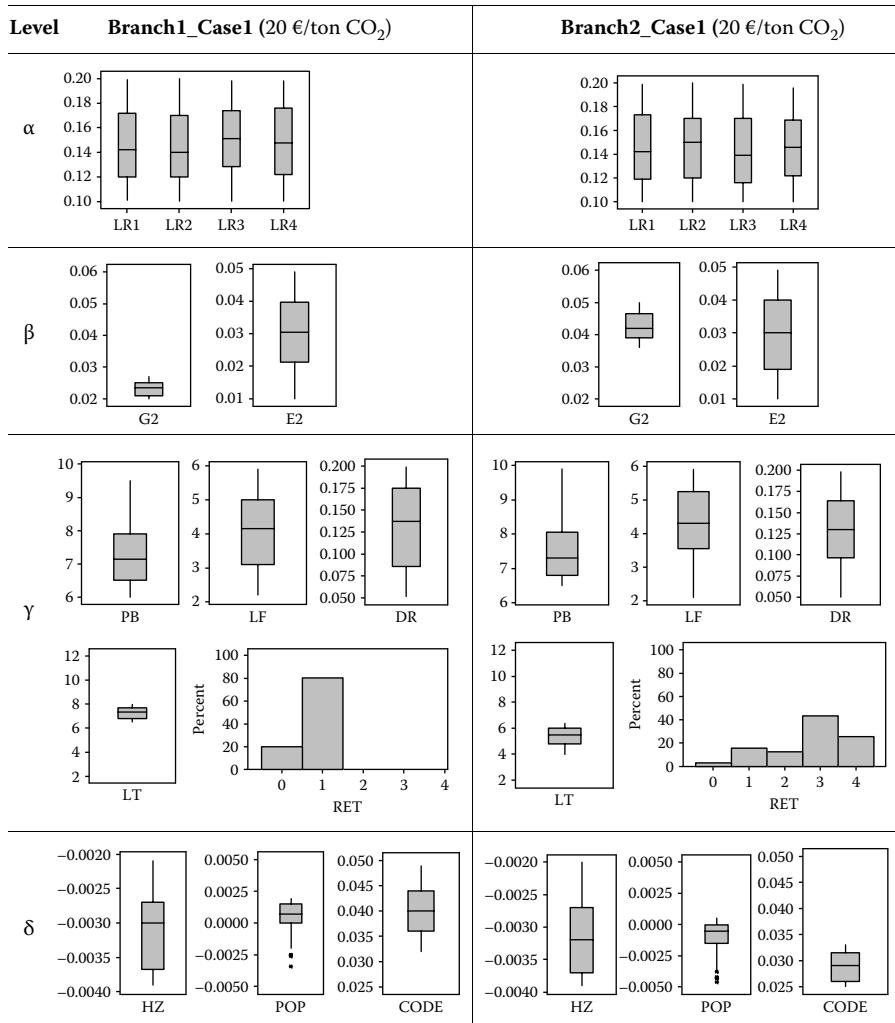


Figure 8.10 The applicable range of system variables for policy adaptation Case1.

At the δ level, there are two apparent trade-offs between population growth and building codes (CODE). In Branch1, the requirement for more stringent building codes allows the population to grow at positive level, to almost 0.25% at the maximum. In Branch2, on the other hand, while the building code is not so strict (lower range regime), the population is barely allowed to grow or even has to decline. A positive population growth will bring the emission level above the 2050 target (i.e., policy failure).

8.5.5 Analysis of Case2

We now analyze the case in which, in the 2025–2050 period, the subsidy is increased from 20 to 50 €/ton CO₂. First, as a result of this increase, the number of successful scenarios (dataset that lead to policy success) rises from 69% to 76.7%, almost an 8 percentage point increase. So, apparently an increased subsidy has some impact on the performance of the system.

The CART for Case2 is presented in Figure 8.11. As in Case1, we illustrate one branch of the “if-then” rule tree (Branch3). Here, we select a particular branch that has the electricity growth figure (E2) as an influencing variable. Branch3 is then reconstructed in box plots to identify the applicable range for each SoS variable (see Figure 8.12).

One particular performance pattern emerges here. At β level, the G2 is required to grow at a high rate range, whereas E2 at a low rate one. The combination of G2 and E2 realizations allows the population to grow at high rate (above 0.25%) at δ level. This pathway is enabled by relatively less stringent constraints on other variables. Some even have less demanding requirements, like a low value range for both LT at γ level and CODE at δ level.

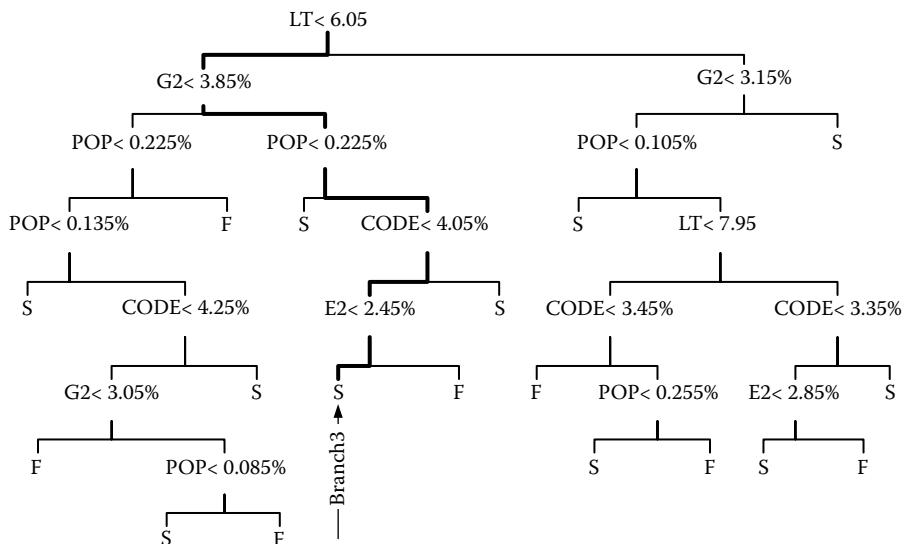


Figure 8.11 Case2 CART for policy adaptation. Minimum cross-validation error = 16.5%. Legend: category S: policy success; category F: policy failure.

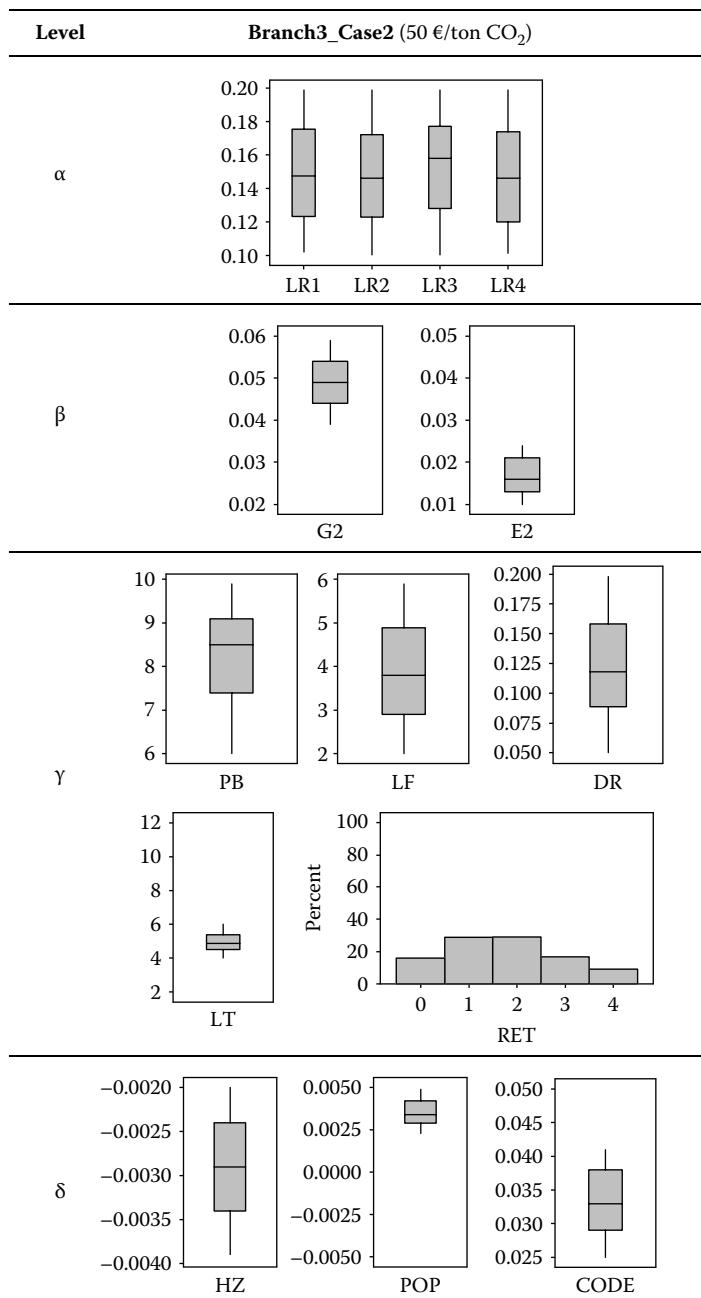


Figure 8.12 The applicable range of system variables for policy adaptation Case2.

8.5.6 Implications for policy design

The analyses above illustrate the different pathways for policy adaptation needed to bring the carbon emission level back to the 2050 target. They also highlight the trade-offs that need to be made regarding realizations of SoS variables.

To benefit from this insight, the evolution of SoS variables needs to be monitored in order to establish the state of the SoS and hence which pathway is relevant. For instance, when the population is growing at high rate, Branch3 is more relevant to inform the requirements than Branch1 or Branch2.

The implication for policy design is that policymakers should choose a policy set based on the feasibility to influence the system variables to remain in their applicable range. Obviously, there is a trade-off to be made. It might be the case that influencing the price of gas (β level) by tax, for instance, is more feasible than influencing the demolition rate (γ level) by regulation or influencing the building code (δ level). Other policy alternatives may have impact on achieving the policy target to reduce carbon emissions. They might include a fiscal policy to influence the investment behavior represented by the discount rate (γ level) or an awareness campaign to relax the hurdle for investment on energy efficiency technology represented by higher payback threshold (γ level). Controversially, one might envision demographic policies to influence population growth and household size (δ level), which have a large impact on the emission level. The choice of a policy set requires careful analysis, which is beyond the scope of this chapter. The point is that SoS provides a conceptual framework to structure the complexity of policymaking, taking into account the interactions and possible trade-offs among the SoS levels.

Lastly, in one of the findings, Branch2 suggests that a zero or negative population growth is required to achieve the policy target. When the population actually increases, the target will be out of reach, requiring further relaxations of system constraints (e.g., higher growth rate of gas price) or additional policies that promote the use of lower- or zero-carbon fuels. In this light, it is very relevant to question whether the policy target itself needs to be adjusted. The approach we propose can be used to test the achievability of the policy target.

8.6 Concluding remarks

We have demonstrated how the adaptive policy approach can be applied under conditions of uncertainty, enabling a policy to be adapted as some of the uncertainties are resolved. Integral to the approach is the exploratory modeling and analysis method which leads to insights that can be used to support the design of an adaptive policy.

Reductions in carbon emission levels from the Dutch residential sector provide the case study for the methodology demonstration. In particular, when emissions are off the target, multiple sets of conditions can be identified to bring the emissions back to the desired level. CART technique in this case provides insight into the boundaries between policy success and failure. Further analysis reveals the trade-off among system-of-system variables and possible pathways required for policy adaptation. Policy design should then focus on factors that policymakers can feasibly influence. The system-of-systems perspective, in this case, provides the problem definition commensurate with the complexities of the problem, allowing for identifying and specifying different kinds of policy systems at various levels to be influenced by policies.

Acknowledgments

This research is supported by The Next Generation Infrastructure Foundation and by NWO (Dutch Organization for Scientific Research) under the research program "Managing uncertainties in energy innovations and transition processes." We are grateful to the valuable comments from Wil Thissen, Warren Walker, and Andre Faaij.

References

1. Hammitt, J. K., R. J. Lempert, and M. E. Schlesinger. 1992. A sequential-decision strategy for abating climate change. *Nature* 357:315–318.
2. Morgan, M. G., M. Henrion, and M. G. Morgan. 1990. *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*. Cambridge University Press, New York.
3. Walker, W. E. 2000. Policy analysis; a systematic approach to supporting policymaking in the public sector. *Journal of Multi-Criteria Decision and Analysis* 9:11–27.
4. Lempert, R. et al. 2004. Characterizing climate-change uncertainties for decision-makers—an editorial essay. *Climatic Change* 65:1–9.
5. Lempert, R., S. Popper, and S. Bankes. 2003. *Shaping the Next One Hundred Years, New Methods for Quantitative Long-Term Policy Analysis*. The RAND Pardee Centre, Santa Monica.
6. DeLaurentis, D. and R. K. C. Callaway. 2004. A system-of-systems perspective for public policy decisions. *Review of Policy Research* 21(6):829–837.
7. Walker, W. E., S. A. Rahman, and J. Cave. 2001. Adaptive policies, policy analysis, and policy-making. *European Journal of Operational Research* 128(2):282–289.
8. Lempert, R. J., M. E. Schlesinger, and S. C. Bankes. 1996. When we don't know the costs or the benefits: adaptive strategies for abating climate change. *Climatic Change* 33:235–274.
9. Bankes, S. 1993. Exploratory modeling for policy analysis. *Operations Research* 41(3):435–449.
10. Hedges, J. S. 1991. Six (or so) things you can do with a bad model. *Operations Research* 39(3):355–365.

11. Sterman, J. D. 2002. All models are wrong: reflections on becoming a systems scientist. *System Dynamics Review* 18(4):501–531.
12. IEA. 2000. *Energy policies of IEA countries: The Netherlands*. IEA/OECD, Paris.
13. CBS. 2006. *Statline*. Statistics Netherlands.
14. Oosterhuis, F. and A. E. Nieuwlaar. 1999. Energy use for residential space heating in the Netherlands 1990–1995: An empirical analysis. Working paper, Utrecht University, Department of Science, Technology and Society, Utrecht.
15. BAK. 2000. *Investigation of natural gas use by private consumers (in Dutch)*. 1980–2000, Amsterdam: Centrum voor Marketing Analyses.
16. Klein Goldewijk, K. et al. 2005. Greenhouse Gas Emissions in the Netherlands 1990–2003 National Inventory Report 2005. Netherlands Environmental Assessment Agency (MNP), Bilthoven.
17. Boonekamp, P. G. 2005. Improved Methods to Evaluate Realised Energy Savings. Doctoral Dissertation, Utrecht University.
18. Wolbers, R. 1996. *Floor area of Netherlands' dwellings (in Dutch)*. Department of Science, Technology and Society (NW&S), Utrecht University.
19. Petersdorff, C., T. Boermans, and J. Harnisch. 2006. Mitigation of CO₂ emissions from the EU-15 building stock. Beyond the EU Directive on the Energy Performance of Buildings. *Environmental Science and Pollution Research* 13(5):350–358.
20. Treffers, D. J. et al. 2005. Exploring the possibilities for setting up sustainable energy systems for the long term: two visions for the Dutch energy system in 2050. *Energy Policy* 33(13):1723–1743.
21. Dutch Ministry of Economic Affairs. 2005. Now for Later; Energy Report 2005. Ministry of Economic Affairs, The Hague.
22. European Commission. 2004. Housing statistics in the European Union. European Commission, Brussels.
23. Agusdinata, D. B. 2006. Specification of system of systems for policymaking in the energy sector. In *IEEE SMC System of Systems Engineering Conference*, Los Angeles, CA.
24. Dittmar, L., A. P. C. Faaij, and W. C. Turkenburg. 2007. *DREM: The Dutch Residential Energy Model*. Utrecht University, Department of Science, Technology and Society.
25. Wene, C.-O. 2000. *Experience Curves for Energy Technology Policy*. IEA / OECD, Paris.
26. BCG. 1972. *Perspectives on Experience*. Boston Consulting Group, Boston, MA.
27. Boyd, D. W., R. L. Phillips, and S. G. Regulinski. 1982. A model of technology selection by cost minimizing producers. *Management Science* 28(4):418–424.
28. Clarke, J. F. and J. A. Edmonds. 1993. Modelling energy technologies in a competitive market. *Energy Economics* 15(2):123–129.
29. Johnstone, I. M. 2001. Energy and mass flows of housing a model and example. *Building and Environment* 36(1):27–41.
30. Elsinga, M. and C. Lamain. 2004. *Onttrekkings- en overlevingskansen van woningen*. OTB Research Institute for Housing, Urban and Mobility Studies, Delft.
31. Bekker, P. C. F. 1991. A Lifetime Distribution Model of Depreciable and Reproducible Capital Assets. Ph.D. thesis, VU University Press, Amsterdam.
32. Blok, K. 2004. Improving energy efficiency by five percent and more per year? *Journal of Industrial Ecology* 8(4):87–99.
33. Evolving Logic. 2005. www.evolvinglogic.com.
34. Helton, J. C. and F. J. Davis. 2000. Sampling based methods. In *Sensitivity Analysis*, eds. A. Saltelli, K. Chan, and E. M. Scott. Willey, Chichester.

35. Mishra, S., N. E. Deeds, and B. S. RamaRao. 2003. Application of classification trees in the sensitivity analysis of probabilistic model results. *Reliability Engineering and System Safety* 79(2):123–129.
36. Breiman, L. et al. 1984. *Classification and Regression Trees*. Wadsworth, Monterey, CA.
37. Janssen, L. H. J. M., V. R. Okker, and J. Schuur. 2006. Welfare, prosperity and quality of the living environment: A scenario study for the Netherlands in 2040 (in Dutch). Background document. Den Haag, Bilthoven, Den Haag: Centraal Planbureau, Natuurplanbureau, Ruimtelijk Planbureau, Netherlands.
38. Tukey, J. W. 1977. *Exploratory Data Analysis*. Addison-Wesley, Reading, MA.

chapter nine

Medical and health management system of systems

Yutaka Hata, Syoji Kobashi, and Hiroshi Nakajima

Contents

9.1	Systems of systems in medical ultrasonics	233
9.1.1	Ultrasonic surgery support system.....	233
9.1.1.1	Segmentation system.....	236
9.1.1.2	Registration system	238
9.1.1.3	Location system.....	239
9.1.2	Summary	239
9.2	System of systems in medical imaging	240
9.2.1	Image segmentation in cooperation with expert knowledge system.....	240
9.2.1.1	Growing criteria of general RG.....	240
9.2.1.2	Growing criteria of RG with expert system	241
9.2.2	Image registration in cooperation with expert knowledge system.....	243
9.3	System of systems in health management	244
9.3.1	Evolution of science, technology, and society	245
9.3.2	The world and problems requiring solutions	245
9.3.3	Health management technology based on causality	245
9.3.4	Application study	247
9.3.5	Summary and discussion.....	248
References	249	

9.1 Systems of systems in medical ultrasonics

9.1.1 Ultrasonic surgery support system

Ultrasonic techniques are widely applied in medicine. The most popular usage is to image the inside of the human body. Clinical ultrasonic treatment

is also essential to disrupt objects such as gallstones. All systems consist of hardware and software. Current medical ultrasonic systems require system of systems engineering (SoSE) techniques comprising the hardware systems of an ultrasonic probe, a pulser and receiver, an A/D converter that can rapidly process large amounts of data, and the software systems of data synthesis, analysis, and image rendering. In this section, we introduce an ultrasonic SoS for clinical orthopedic surgery.

An implant is used to reinforce broken bones in orthopedic treatment. An intramedullary titanium nail that supposedly has no deleterious effects on the body functions as an implant is inserted into broken bones (Figure 9.1). During the first step of this procedure, surgeons drive a nail into the femur.

In the second step, the region is opened to drill holes for screws. In the third step, two screws are positioned into the holes of the intramedullary nail in the bone, and then the region is closed. Figure 9.2 (a,b) shows typical X-ray images before and after surgery, respectively. In the second step, surgeons cannot visualize screw holes in nails hidden in the bone using the naked eye. Therefore, an X-ray device is often used to locate the holes. Several studies [1,2]



Figure 9.1 Titanium nails.

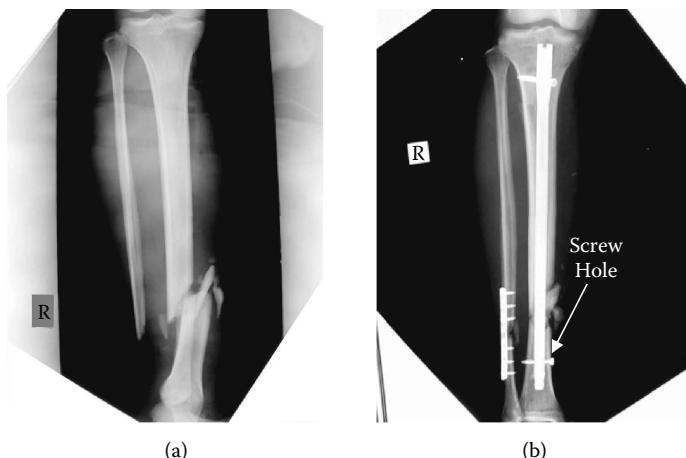


Figure 9.2 X-ray images of broken bones. (a) Before surgery. (b) After surgery.

have examined ways to technically support surgeons and decrease the surgical duration. However, radiography has the disadvantage of X-ray exposure. To solve this problem, we developed an ultrasonographic system that can locate screw holes in intramedullary nails. This system is practical for clinical applications, because it is sufficiently compact for use on the operating table, requires no mechanical systems with complex alignments, and the surgical duration is short. System of systems engineering was required to develop this system, because it requires several hardware and software systems.

The hardware system consists of a 32-element array probe (ISL Inc., ISL1938; Figure 9.3(a)) and a single-axis freehand scanner (Figure 9.3(b)). The probe performs linear scanning and transmits the data to a personal computer. The scanner transmits x-axis information of all scanning points to the same computer, which provides the coordinates of the screw holes to surgeons. This hardware SoS is shown in Figure 9.4.

The computer system has a software SoS to locate screw holes of intramedullary nails. Our software system consists of the three systems shown in [Figure 9.5](#). The segmentation system extracts the regions of the screw holes. The

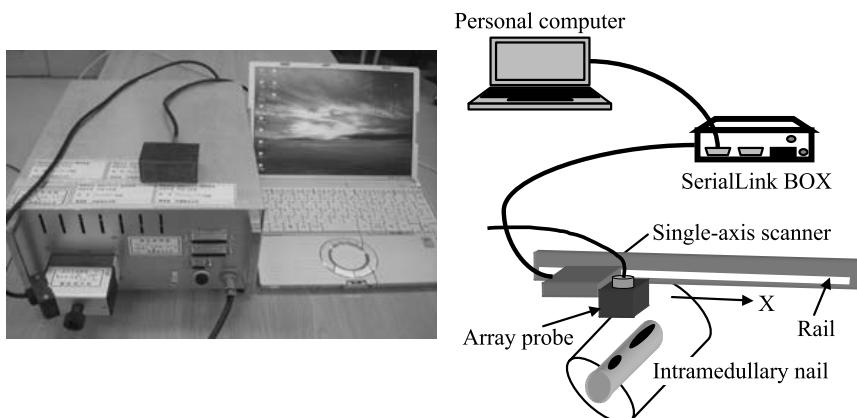


Figure 9.3 Hardware systems.

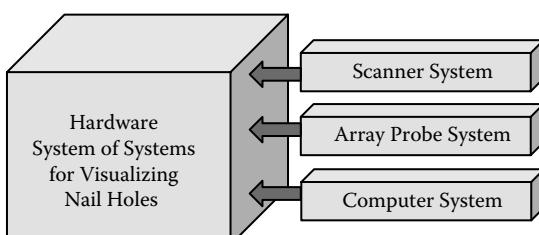


Figure 9.4 Hardware system of systems.

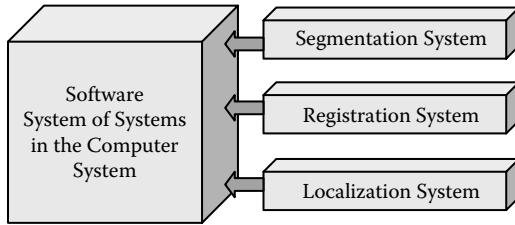


Figure 9.5 Software system of systems.

registration system performs the registration between the obtained image and the true image that is obtained by a digital camera. The location system identifies screw hole positions using a Euclidean translation matrix obtained from the registration system. The details of the process are as follows.

9.1.1.1 Segmentation system

The array probe system outputs ultrasonic waves. The array probe is scanned at a distance of 30 mm at 2.0-mm intervals guided by unidirectional freehand scanning. The 2.0-mm interval is experimentally determined. The sampling interval is 20 ns. The peak-to-peak value (P-P value) is calculated by the difference between the minimum and maximum amplitude of every ultrasonic wave. A feature value map is defined as an image $I(x, y)$ ($0 \leq x \leq M, 0 \leq y \leq N$) consisting of the P-P values with 8-bit intensities from 0 to 255 [3,4]. [Figure 9.6\(a\)](#) shows a feature value map in which the brightness indicates larger echoes from the nail, and the nail surface has higher intensity, whereas the screw holes have lower intensity. We inserted a 2.0-mm interval line into the map because our scanning interval is 2.0 mm.

First, we determine a gradient g of the long axis as the direction of the intramedullary nail using Equation (9.1).

$$g = \frac{\sum_{x=1}^M \sum_{y=1}^N (x - \frac{\sum_{x=1}^M \sum_{y=1}^N x \times f(x, y)}{\sum_{x=1}^M \sum_{y=1}^N f(x, y)}) \times (y - \frac{\sum_{x=1}^M \sum_{y=1}^N y \times f(x, y)}{\sum_{x=1}^M \sum_{y=1}^N f(x, y)}) \times f(x, y)}{\sum_{x=1}^M \sum_{y=1}^N (x - \frac{\sum_{x=1}^M \sum_{y=1}^N x \times f(x, y)}{\sum_{x=1}^M \sum_{y=1}^N f(x, y)})^2 \times f(x, y)} \quad (9.1)$$

where $f(x, y) = 1$ if $I(x, y) > th$; $= 0$, otherwise. The threshold, th , is experimentally determined from the intensity in each feature value map. The knowledge required to segment screw holes of an intramedullary nail is as follows [3,4].

Knowledge 1: The average of the intensity is low in the screw holes.

Knowledge 2: The variance of the axis is high on the center line of the intramedullary nail.

Two fuzzy if-then rules are derived from Knowledge 1 and Knowledge 2.

Fuzzy Rule 1: IF the average, $ave(x, y)$, is LOW, THEN the degree μ_{ave} is HIGH.

Fuzzy Rule 2: IF the variance of line, $var(x, y)$, is HIGH, THEN the degree μ_{var} is HIGH.

The notation $ave(x, y)$ indicates the average of eight neighborhood pixels of (x, y) in the feature value map. The notation $var(x, y)$ indicates the variance of the intensity of the line with point (x, y) and gradient g . The fuzzy membership functions are defined in Figure 9.7. In Figure 9.7, max_{ave} and max_{var} are calculated as the maximum values of $ave(x, y)$ and $var(x, y)$, respectively, in the map. We calculate fuzzy degrees $\mu_{ave}(x, y)$ and $\mu_{var}(x, y)$ from Equation (9.2) and (9.3), respectively.

$$\mu_{ave}(x, y) = \min(S_{ave(x, y)}(a), \text{LOW}) \quad (9.2)$$

$$\mu_{var}(x, y) = \min(S_{var(x, y)}(v), \text{HIGH}) \quad (9.3)$$

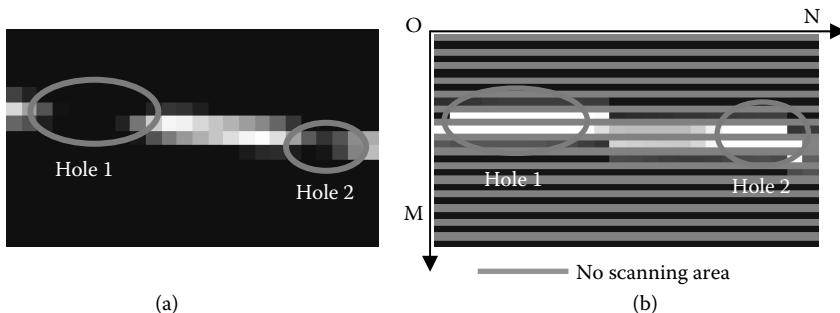


Figure 9.6 Feature value map and $\mu_{hole}(x, y)$.

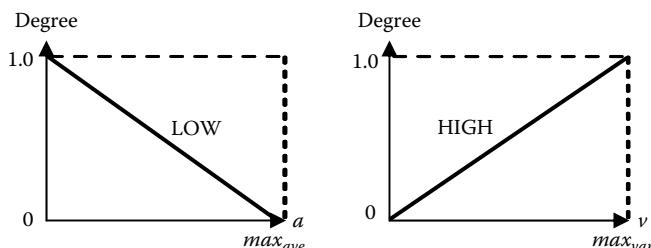


Figure 9.7 Membership functions.

Fuzzy singleton function, $S_{ave(x,y)}(a)$, is defined as $S_{ave(x,y)}(a) = 1$ if $a = ave(x, y); = 0$, otherwise, and $S_{var(x,y)}(v)$ is defined as $S_{var(x,y)}(v) = 1$, if $v = var(x, y); = 0$, otherwise. A fuzzy degree, $\mu_{hole}(x, y)$, is calculated from Equation (9.4).

$$\mu_{hole}(x, y) = \mu_{ave}(x, y) \times \mu_{var}(x, y) \quad (9.4)$$

The degree $\mu_{hole}(x, y)$ represents the degree of screw holes in the nail; that is, the hole regions are segmented. Figure 9.6(b) shows the image of $\mu_{hole}(x, y)$ by the intensities of $255 \times \mu_{hole}(x, y)$, in which the gray line indicates the absence of a scanning line.

9.1.1.2 Registration system

The registration process matches both images and is used to locate screw holes. A true image (Figure 9.8(a)) is obtained using a digital camera. We first segment the hole regions from the true image using a sobel filter and region growing. The resultant image is called a segmented image (Figure 9.8(b)). We then standardize the resolution of the two images and perform a Euclidean translation of the image in Figure 9.8(b) to match $\mu_{hole}(x, y)$ that appears in Figure 9.6(b); that is, registration between Figure 9.8(b) and Figure 9.6(b). The concept is shown in Figure 9.9. We calculate the square sum of the difference, $SumD$, between the two images using Equation (9.5), where the notation $T(x, y)$ denotes the intensity of the segmented image.

$$SumD = \sum_{x=1}^M \sum_{y=1}^N (255\mu_{hole}(x, y) - T(x, y))^2 \quad (9.5)$$

We determine the translation matrix with the minimal $SumD$ and then translate the image into the desired position.

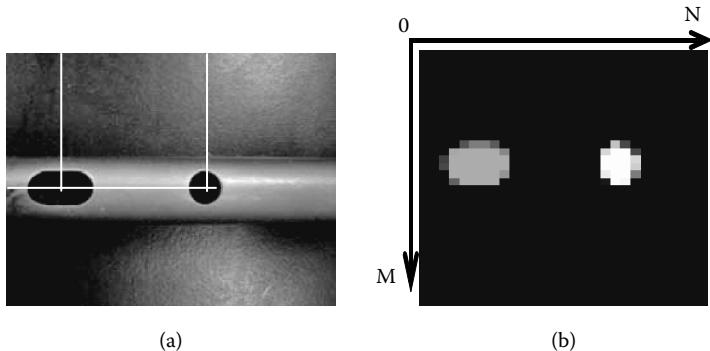


Figure 9.8 True image and segmented holes.

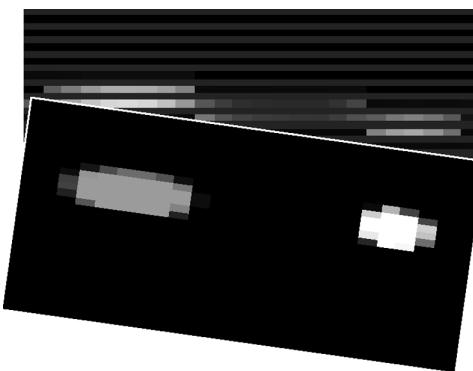


Figure 9.9 Registration.

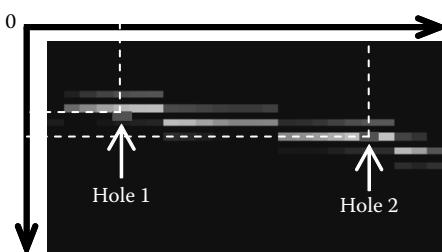


Figure 9.10 Resultant image.

9.1.1.3 Location system

The positions of the screw holes in the $\mu_{hole}(x, y)$ image are calculated from the Euclidean translation matrix, because we know the screw hole positions in the true image. The location system provides the necessary coordinates of the screw hole centers, because we know the start position of the scan. The resultant image is shown in Figure 9.10. Thus, the surgeons know where the center points of the holes are for precise screw insertion.

9.1.2 Summary

We introduced a SoSE technique for a support system comprising a clinical ultrasonic support system for locating screw holes during orthopedic surgery. The system is constructed of hardware and software SoSE techniques. The system complements the surgical technique of inserting screws into holes in a nail positioned within the human body. The results of our experiments showed that this system provides precise information (1.0 mm margin of error) about screw holes within about 5 minutes. This level of accuracy

and processing time are sufficient for clinical applications. As this example shows, SoSE will be widely applied to future surgical support systems. In other words, such systems can only be established using SoSE techniques.

9.2 *System of systems in medical imaging*

Medical image processing of magnetic resonance (MR) imaging, X-ray computed tomography (CT), and radiography play fundamental roles in computer-aided diagnosis (CAD) systems that produce volume visualization within the body. Although many image processing techniques have been investigated, only simple techniques such as thresholding and edge detection have been applied to medical image processing. The principal difference between medical image processing and general image processing is the need for specialized knowledge. Although expert knowledge has been used to provide image processing parameters, to understand and fine-tune these parameters is difficult and can obstruct the processing of medical images.

This chapter introduces new approaches to construct SoS in medical image processing by integrating the expert knowledge systems into image processing systems. In particular, this chapter focuses on the segmentation and registration of medical images. Fuzzy logic can be applied to represent the expert knowledge and thus integrate the systems. Some applications are described to illustrate the effectiveness of these approaches.

9.2.1 *Image segmentation in cooperation with expert knowledge system*

Image segmentation is a way to define a region of interest (ROI) from medical images. For example, ROIs include soft tissues (brain, viscera, etc.), hard tissues (bone), and tumors. In general, image segmentation methods have no *a priori* knowledge of ROI. Therefore, system designers must adjust the image processing parameters (usually called magic numbers) by trial and error. In the present SoS, the expert knowledge system includes ROI features such as location, shape, and intensity, and the image processing system segments ROIs by evaluating the appropriate region using the expert system.

For example, consider a region-growing (RG) algorithm, which segments a ROI by recursively including neighboring voxels (or pixels in 2-D images) that satisfy the growing criteria into the growing regions. Examples of growing criteria are as follows.

9.2.1.1 *Growing criteria of general RG*

When a neighboring voxel satisfies all following conditions, the voxel is included in the growing region.

$$[\text{Condition } \#1] I_C \geq 150 \quad [\text{Condition } \#2] I'_C < 10 \quad [\text{Condition } \#3] x_C < 128$$

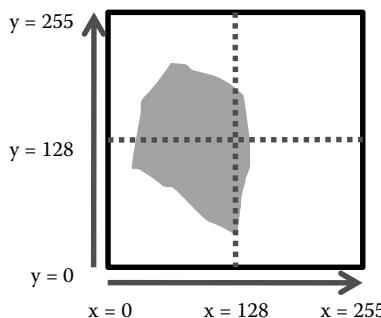


Figure 9.11 Example of undersegmentation by general RG. The gray region is the target region to be segmented.

I_C and I'_C are intensity and differential value of the voxel, respectively, and x_C is the x -coordinate value of the voxel.

This growing criterion is designed so that the RG algorithm segments a region in which intensities of pixels are high, differential values inside the region are low, and location is left. However, this growing criterion will undersegment ROI from the image shown in Figure 9.11, because part of the target region is over the line of $x_C = 128$.

9.2.1.2 Growing criteria of RG with expert system

If the neighboring voxel is that of a ROI, the voxel is included in the growing region. Whether the neighboring voxel is indeed a voxel of a ROI is evaluated by the expert system. When the expert system is implemented with fuzzy if-then rules, the rules can be given as follows.

IF I_C is HIGH and I'_C is SMALL and x_C is LEFT
THEN the degree of belonging to the ROI is HIGH.

HIGH, SMALL, and LEFT are fuzzy linguistic values defined by the fuzzy membership functions given by Figure 9.12. This fuzzy if-then rule can be evaluated by Mamdani's MIN-MAX implementation [5], and a resultant fuzzy degree belonging to the ROI will be obtained. Then, when the resultant fuzzy degree exceeds a threshold (for example, 0.5), the voxel is included in the growing region.

This growing criterion based on the fuzzy expert system will correctly segment the target region from the image shown in Figure 9.11, because the undersegmented area with the general RG has enough of the features of a ROI. In addition, because fuzzy if-then rules and fuzzy membership functions are easy to understand, users can design the rules according to their expert knowledge.

Figure 9.13 shows applications of a SoS, which comprises image segmentation and expert knowledge systems. Figure 9.13(a) shows segmented MR images of the human brain that are helpful in understanding cerebral atrophy.

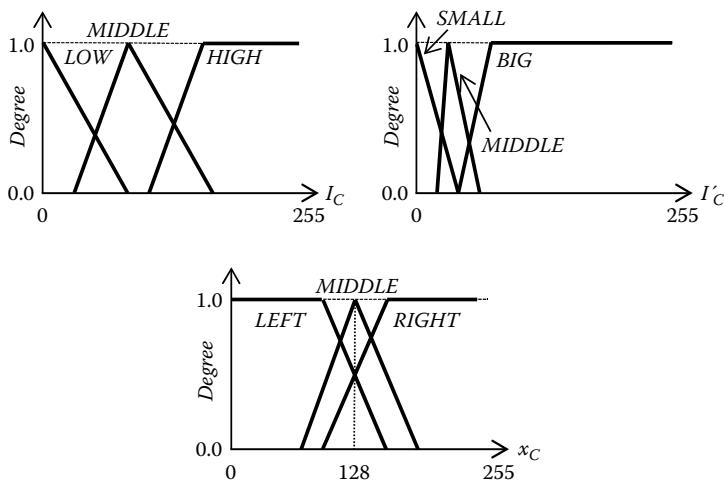


Figure 9.12 Fuzzy membership functions. Left, degree of intensity; middle, degree of difference; right, degree of location.

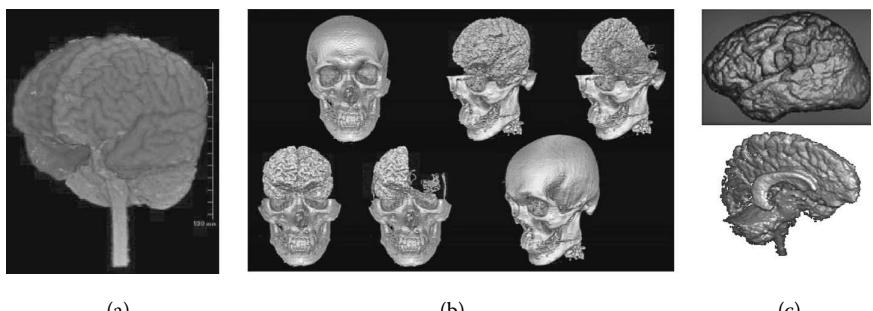


Figure 9.13 (a) Segmented brain parts from MR images. (b) Integration of three regions segmented from MR, MR angiography, and CT images. (c) Segmented cerebral lobes and lateral ventricles from MR images.

To reconstruct the images, the brain region was segmented by 3-D RG using an automated threshold finding method, and the segmented region was decomposed into brain parts by evaluating the position, intensity, distance, etc. The threshold finding method and the evaluation system were constructed by integrating an expert knowledge system [6]. Figure 9.13(b) shows the anatomical structure of the human head that can be used to plan neurosurgery. These images were reconstructed from three images produced by three systems: the first segmented the brain region from MR images, the second segmented the cerebral artery region from MR angiography images [7], and the third segmented the skull region from CT images, because each imaging modality can describe regions with high contrast. Figure 9.13(c) shows the cerebral lobes segmented by an active surface model in which a fuzzy expert system evaluates the

deforming model [8], and the lateral ventricles are segmented by a watershed segmentation algorithm in which a fuzzy expert system provides knowledge of the shape and location [9].

9.2.2 Image registration in cooperation with expert knowledge system

Image registration is a method of finding appropriate parameters of affine transformation (such as translation and rotation) of one image so that the transformed image matches with the other image. It is achieved by searching parameters with maximum matching scores. For example, the matching score between image I_A and image I_B can be defined by:

$$\mu = \sum_{x=0}^M \sum_{y=0}^N \{ \kappa_I G(x, y) H(x, y) + \kappa_D J(x, y) K(x, y) \} \quad (9.6)$$

where M and N are the width and height of the image, respectively, $G(x, y)$ and $H(x, y)$ are the intensity values of images I_A and I_B , respectively, $J(x, y)$ and $K(x, y)$ are differential values, and κ_I and κ_D are weighting parameters.

Equation (9.6) shows that the matching score equivalently evaluates all pixels in the given image. However, in the human sense, pose/position can be understood by observing specific areas without actually being able to see the entire area. For example, consider the silhouette shown in Figure 9.14 (a). Assume that this image is that of a cup with a handle, and that part of the cup is covered by an obstacle. That the cup is located with the pose/position shown in Figure 9.14(b) can be established by observing the protrusion of the silhouette.

By simulating the human manner of pose/position recognition, the matching score defined by Equation (9.6) can be rewritten as:

$$\mu = \frac{\sum_{x=0}^M \sum_{y=0}^N \mu_{ROI}(x, y) \{ \kappa_I G(x, y) H(x, y) + \kappa_D J(x, y) K(x, y) \}}{\sum_{x=0}^M \sum_{y=0}^N \mu_{ROI}(x, y)} \quad (9.7)$$

where $\mu_{ROI}(x, y)$ is a fuzzy degree belonging to area of concern, which takes a value between 0 and 1. Zero degree means that the pixel is completely ignored, whereas one degree means that the pixel is completely concerned. For example, by giving areas that must be concerned and areas that must NOT be concerned by users according to their expert knowledge, the fuzzy expert system can calculate and provide the fuzzy degree shown in Figure 9.14(c) to the image registration system.

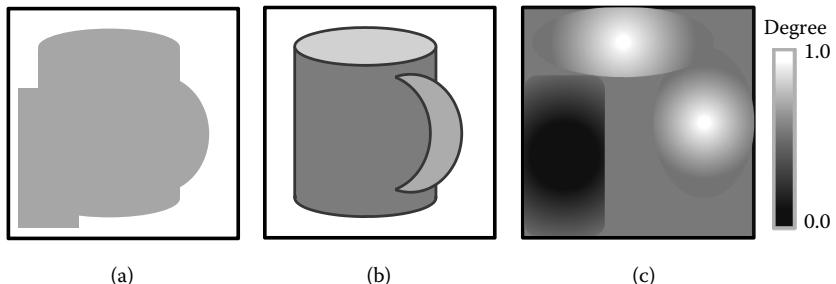


Figure 9.14 Image registration with expert knowledge system. (a) Silhouette of a cup. (b) Pose/position of the cup. (c) Degree of belonging to area of concern.

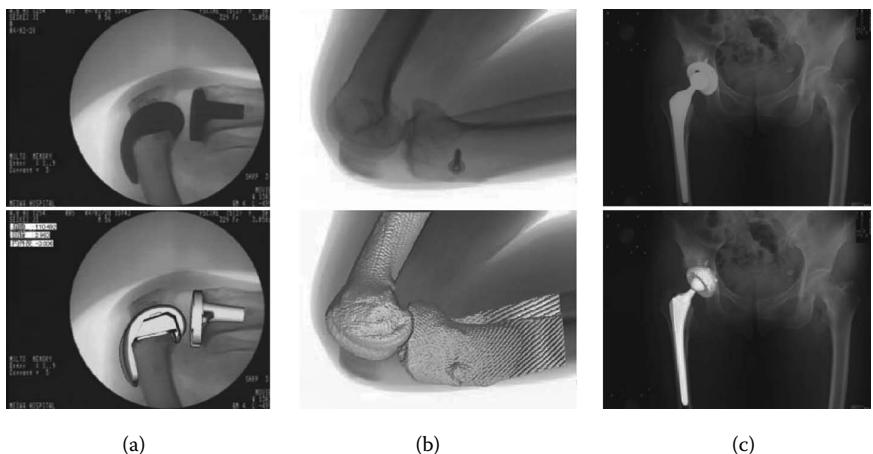


Figure 9.15 Three-dimensional pose/position estimation of TKA implant from 2-D X-ray fluoroscopic image (a) of femoral and tibial bones from 2-D digital radiograph (b), and THA implant from 2-D X-ray fluoroscopy image. (c) Upper and lower panels: raw images and 3-D images reconstructed by the proposed systems, respectively.

Figure 9.15(a) shows an application of the SoS to 2-D/3-D image registration between a 2-D X-ray image and a 3-D geometric model of a total knee arthroplasty (TKA) implant [10]. Figure 9.15(b) shows the results of 2-D/3-D image registration between the 2-D digital radiography (DR) image and 3-D multi-detector-row CT (MDCT) images [11]. Figure 9.15(c) shows the results of 2-D/3-D image registration between a 2-D DR image and a 3-D geometric model of a total hip arthroplasty (THA) implant. These findings are useful for investigating the kinematics of knee/hip joints in orthopedic research.

9.3 System of systems in health management

In this section, the notion of health management technology is introduced and discussed from the viewpoint of SoS. First, the evolution of science, technology,

and society is briefly discussed to define the problems to be solved by observing the world as an organization of human beings, artifacts, and nature. As one important solution to problems, the notion of the health management technology centered on causality is proposed and discussed, especially with application studies. Finally, the health management technology for humans, artifacts and nature is assumed to address the essential issues of SoS and illustrate its important functionality for problem-solving.

9.3.1 Evolution of science, technology, and society

Society, technology, and science have evolved together from ancient eras, whereas technology has been influenced by scientific innovation and by the impetus of society. Especially after the Industrial Revolution of the late eighteenth and nineteenth centuries, society, technology, and science have rapidly evolved, mainly in developed nations. Mass production and consumption led by competitive societies has led to an ever-increasing accumulation of problems. Among them are crises of worldwide social security, as well as of human and environmental health. Unless these problems are resolved in the near future, the social requirements of safety, security, and health, together with individual happiness and well-being, will never be realized. The highly complex nature of target systems such as the environment, humanity, and society obviously reflects that of SoS.

9.3.2 The world and problems requiring solutions

An essential and simple observation of the world reveals that it can be considered as comprising humans, artifacts, and nature. Even though the value of each entity is determined by humans, they are completely different. Examples of such values are comfort and safety for humans, efficiency and effects for artifacts, and environmental enhancement for nature. The problem is that these values come into conflict with each other. Consider a factory and its environment as an example for understanding conflict values. Productivity related to efficiency and effect is the most important value in manufacturing lines. However, focus only on productivity will increase the emission of carbon dioxide and other contaminants that will negatively influence the safety and comfort of human operators in the manufacturing line. Thus, the conflicts among values of the three entities cause serious problems in important areas such as the environment, agriculture and food, security and safety, and human health. Through the discussions above, the harmonization of values among the entities seems to be in direct relationship to the good health of each entity.

9.3.3 Health management technology based on causality

Figure 9.16 shows a general idea of the health management technology based on the causality model. Because the target system continuously changes and

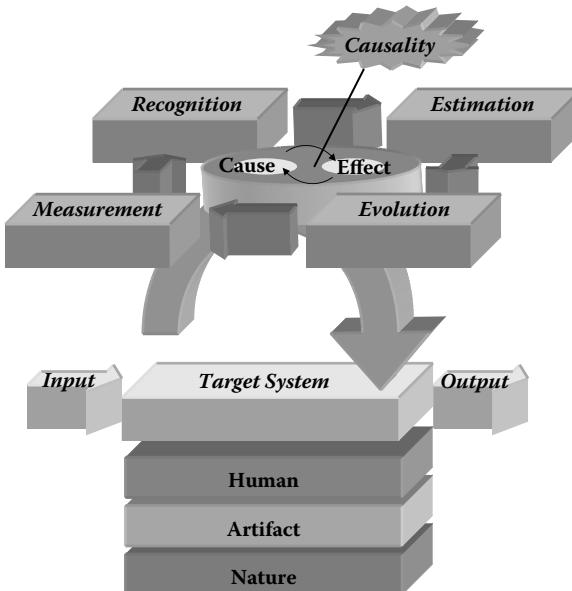


Figure 9.16 Framework of health management technology.

management must therefore adapt, causality must evolve cyclically and continuously. Four kinds of functions are defined for evolving this model. *Measurement* is the quantification of phenomena to arrive at a value from analyzing signals. *Recognition* is to identify the condition of the target system using the measured value. *Estimation* is to predict the past and future status of the system based on simulation using causality. *Evolution* is to improve causality by the discovery of new events and make changes in the target system. Based on causality, *measurement*, *recognition*, *estimation*, and *evolution* are cyclically and continuously executed to improve causality for better health.

Interest in human health improvement has increased in addition to preventing psychological and physiological pathologies. To realize both issues, important continuous and heterogeneous measurements can be obtained using a pedometer, scales, blood pressure meters, etc. 24 hours each day including measurements during sleep, while eating meals, and during exercise. The parameters measured by sensors should lead to the discovery of causalities that can be applied to prevent diseases and improve health. The causality shown in Figure 9.17 is useful for understanding human health status. Blood pressure is used as a reference index of health, because it is closely associated with cardiovascular events such as brain infarction, stroke, myocardial infarction, and heart failure. Active mass, weight, visceral fat, and behavior could be measured continuously using a pedometer, a weight scale, and a sensor bed from the aspects of exercise, meals, and sleep. By analyzing time series data obtained from measuring equipment, the causality shown in Figure 9.17 among active mass, weight, visceral fat, and behavior will be

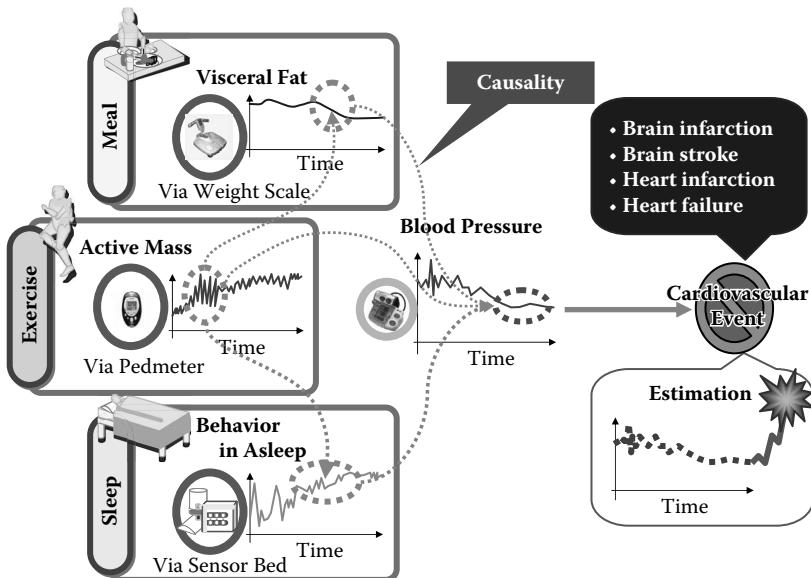


Figure 9.17 Example of causality among meals, exercise, sleep, and blood pressure.

revealed. The causality should reflect individual characteristics, because the optimum quality and quantity of exercise, meal, and sleep depends on individual life styles. Because causality can adapt to changes in individual health status, the solution could be realized in details and optimized for disease prevention and health improvement adapted to specific individuals. The solution could advise the user about the quality and quantity of exercise, for example, "You need to add 2,000 more steps to your commute to lose 1 kg within one month according to your recent monthly data." This kind of message should be very persuasive and motivating for users, because quantitative effectiveness is predicted, and the message is well-organized considering individualization and personalization.

9.3.4 Application study

This chapter introduces an application to the human health management technology from the SoS perspective. The main function provided by the application is behavior estimation of a person lying on a bed. This is the first step toward realizing disease prevention and to improving health based on whole-day measurements as described above.

Figure 9.18 shows the system architecture. The two kinds of sensors used to measure human behavior on the bed were developed under consideration of unconsciousness and noninvasiveness. These features are important for realizing long-term sensing and avoiding influence from conscious and unconscious human reactions. One is an ultrasonic oscillosensor

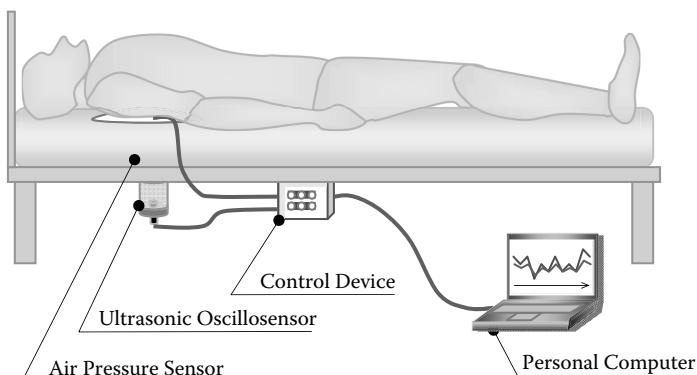


Figure 9.18 System architecture of sensor bed.

(UOS), and the other is an air pressure sensor (APS). Each provides different functionality, because they have different mechanisms. The UOS can measure low-frequency vibration ≤ 10 Hz using ultrasonic reflection from a liquid surface [12]. It can obtain data about almost all phenomena that occur in the entire bed because it is installed just below the bed frame (Figure 9.18), and vibrations can travel through the solid frame. On the other hand, the APS obtains data about phenomena that occur in a specific location, because it is installed underneath the back of a supine person. Data generated by the different types of sensors lead to the classification and prediction of behaviors such as rest, turning over, and getting out of bed [13]. Based on the functions of classification and prediction, an individual can be prevented from falling out of bed. This can be important to the elderly, who can develop fractures from such accidents and then become bedridden. The function can also realize an index of quality and quantity of sleep for health assessments. Besides behavioral estimations, indexes related to physiological status can be measured such as heart beat and heart rate [14]. Such indices could also be nodes of causality for realizing disease prevention and health improvement. Although the system has a simple configuration, the different sensors provide different types of information that can provide powerful solutions.

9.3.5 Summary and discussion

Systems of systems in general health management technology for humans, artifacts, and nature are argued considering a vision of the future of society. Because the most important part of health management is for humans, the general idea of human health management was introduced here. Even though the architecture of the described application is simple, it can lead to extremely powerful strategies for preventing disease and improving health.

Returning to the original idea of health management technology, the entities of humans, artifacts, and nature should be taken care of to realize good

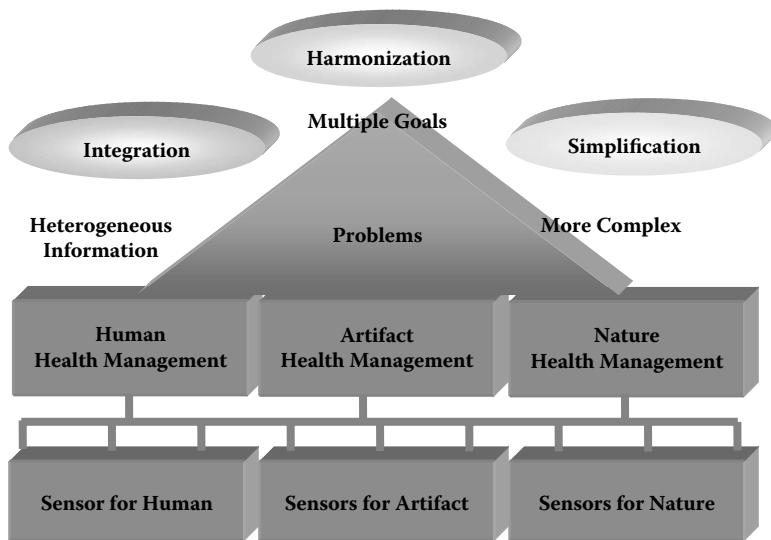


Figure 9.19 System of systems aspects of health management technology.

health status. To formulate the causalities used in the health management technology of huge and complex systems such as manufacturing, society, and the environment, the quantity and quality of data gathered via sensor networks are incredibly complex and complicated (Figure 9.19). According to the characteristics of the systems, important heterogeneous information, as well as more complex, multiple goals are extracted. The many types of sensors gather data with heterogeneous features such as numerical and time series, images, and text. The amount of sensed data is vast and complex because many sensors are continuously used to obtain data over long periods. As mentioned previously, multiple entities have different goals. *Integration*, *simplification*, and *harmonization* can be thought as useful functionalities against the issues: integration against heterogeneous information, simplification against increasing complexity, and harmonization against multiple goals. The functionalities are useful solutions for developing vast and complex systems considering the issues arising from the nature of SoS.

References

1. Taylor, R. H. 1999. Robotics in orthopedic surgery. In *Computer Assisted Orthopedic Surgery*, ed. L. P. Nolte and R. Ganz, pp. 35–41, Hogrefe and Huber Publishers, Seattle, WA.
2. Slomczykowski, R., M. Hofstetter, Y. Strauss, M. Sati Bourquin, and L. P. Nolte. 1999. Fluoroscopy-based surgical navigation-concept and possible clinical applications. In *Computer Assisted Orthopedic Surgery*, ed. L. P. Nolte and R. Ganz, pp. 206–217, Hogrefe and Huber Publishers, Seattle, WA.

3. Endo, M., K. Nagamune, N. Shibanuma, S. Kobashi, K. Kondo, and Y. Hata. 2007. An ultrasonography system aided by fuzzy logic for identifying implant position in bone. *IEICE Trans. Inf. Syst.* E90-D(12):1990–1997.
4. Ikeda, Y., S. Kobashi, K. Kondo and Y. Hata. 2007. Fuzzy Ultrasonic array system for locating screw holes of intramedullary nail. In *Proc. 2006 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3428–3442.
5. Mamdani, E. H. and S. Assilian. 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Man-Machine Studies* 7(1):1–13.
6. Hata, Y., S. Kobashi, et al. 2000. Automated segmentation of human brain MR images aided by fuzzy information granulation and fuzzy inference. *IEEE TSMC-C* 30(3):381–395.
7. Kobashi, S., N. Kamiura, et al. 2001. Volume quantization based neural network approach to 3D MR angiography image segmentation. *Image and Vision Computing* 19(4):185–193.
8. Kobashi, S., Y. Fujiki, M. Matsui, N. Inoue, K. Kondo, Y. Hata and T. Sawada. 2006. Interactive segmentation of the cerebral lobes with fuzzy inference in 3T MR images. *IEEE TSMC-B* 36(1):74–86.
9. Kobashi, S., K. Kondo, and Y. Hata. 2006. Fully automated segmentation of cerebral ventricles from 3-D SPGR MR images using fuzzy representative line. *Soft Computing* 10(2):1181–1191.
10. Kobashi, S., T. Tomosada, et al. 2005. Fuzzy image matching for pose recognition of occluded knee implants using fluoroscopy images. *J. of Advanced Computational Intelligence and Intelligent Informatics*, 9(2):181–195.
11. Kubo, D., S. Kobashi, et al. 2007. Fuzzy ROI based 2-D/3-D registration for kinetic analysis after anterior cruciate ligament reconstruction. *Proc. of North American Fuzzy Information Processing*, pp. 266–270.
12. Kamozaki, Y., T. Sawayama, K. Taniguchi, S. Kobashi, K. Kondo, and Y. Hata. 2007. A new ultrasonic oscillosensor and its application in biological information measurement system aided by fuzzy theory. *IEICE Trans. Inf. Syst.* E90-D(11):1864–1872.
13. Yamaguchi, H., H. Nakajima, K. Taniguchi, S. Kobashi, K. Kondo, and Y. Hata. 2007. *Fuzzy detection system of behavior before getting out of bed by air pressure and ultrasonic sensors*. In *Proceedings of IEEE International Conference on Granular Computing 2007*, pp.114–119, Fremont, TX.
14. Hata, Y., Y. Kamozaki, T. Sawayama, K. Taniguchi, and H. Nakajima. 2007. A heart pulse monitoring system by air pressure and ultrasonic sensor systems. In *Proceedings of IEEE International Conference on System of Systems Engineering*, paper no. 122 in CD-ROM, San Antonio, TX.

chapter ten

The microgrid as a system of systems

Laurence R. Phillips

Contents

10.1	What is a microgrid?	252
10.1.1	Motivation: a straightforward microgrid use case	254
10.1.2	Microgrid problems and issues	255
10.2	The microgrid decisionmaking agency and its functions.....	256
10.2.1	Essential decisionmaking agency capabilities	257
10.2.2	Decisionmaking agency roles.....	257
10.2.2.1	User	258
10.2.2.2	System point of contact	258
10.2.2.3	Architectural advisor	258
10.2.2.4	Strategic planner	258
10.2.2.5	Strategic monitor.....	258
10.2.2.6	Tactical monitor and controller.....	258
10.2.2.7	Device protector	258
10.3	Microgrid organizing principles.....	259
10.3.1	Organization of the microgrid in the context of a primary power grid	259
10.3.2	Organizations made up of microgrids.....	259
10.3.3	Organizing principles for power systems made up of microgrids	260
10.4	Behavior of the organizational elements	261
10.5	Policy for microgrid operation	262
10.6	A DMA policy outline	264
10.7	The subsystems that make up a microgrid	265
10.7.1	Electric power subsystems	265
10.7.1.1	Production.....	266
10.7.1.2	Storage	267
10.7.1.3	Distribution.....	267
10.7.2	Management, operation, and control.....	268
10.7.3	Communication	268

10.7.4	Cyber security	269
10.7.5	Decisionmaking and optimization.....	269
10.7.6	Modeling and prediction.....	271
10.7.7	Planning and monitoring.....	271
10.7.8	Computation.....	272
10.8	Microgrid use cases.....	272
10.8.1	Operation under normal conditions.....	273
10.8.2	Operation under anomalous, unexpected, and failure conditions	273
10.8.3	Operation at system capacity	273
10.8.4	Operation connected to the primary grid	273
10.8.4.1	Central control when connected to the main grid.....	273
10.8.4.2	Distributed control when connected to the main grid.....	274
10.8.5	Operation disconnected from the primary grid.....	275
10.8.5.1	Central control when disconnected from the main grid.....	275
10.8.5.2	Distributed control when disconnected from the main grid.....	276
10.8.6	Transition from the connected to the disconnected state.....	276
10.9	Conclusions and future work	277
	References	277

10.1 What is a microgrid?

For the purposes of this chapter, a microgrid is a collection of small, non-collocated electric power sources, storage devices, and power conditioners interconnected and operated to meet the power requirements of a designated community. [Figure 10.1](#) illustrates the essential differences between common power system arrangements. As this is being written, there are many small power sources in operation; the Energy Information Administration reports 28,744 dispersed and distributed* generators with a total capacity of 14,532 MW of installed combined heat and power sources in operation in the United States in 2005 [1]. The mean size of these generators is slightly more than 500 kW. A generator that size can hardly be considered “small” in an absolute sense; a 500-kW diesel generator occupies 200 cubic feet and weighs 12 tons and, if run around the clock, could produce enough electricity to supply 600 families.[†] Conversely, the average generator in the United States is 60 MW, 120 times the size of 500-kW generators, which makes them relatively very small.

* Dispersed generators are not connected to the grid; distributed generators are.

† Although this is probably not the appropriate power plant to meet that particular need.

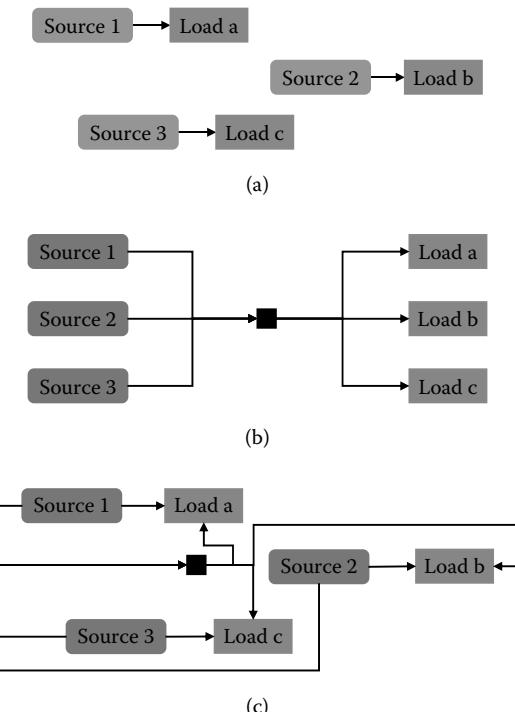


Figure 10.1 The essential difference between microgrids and other configurations is the cooperative operation of the sources despite their physical separation. (a) In an *independent* or *backup* setting, power is delivered to each load from a local source along a dedicated path. Loads are normally unsatisfied when sources fail. (b) In a *powerplant* setting, power is delivered to distributed loads from a set of collocated sources via a power distribution system. Power is normally transmitted over long distances. (c) In a microgrid, primary power is delivered to distributed loads from local sources as in (a), but an ancillary distribution and energy management system delivers power efficiently in low load situations and supplies important loads when their local source fails.

Power sources operated at physically separated sites are almost always operated independently of one another to meet local power needs as determined by local decisionmaking processes based on local information, and otherwise shut down. Most dispersed and distributed generators are operated either as ancillary power sources, for peak-shaving and emergency power, or as “mini power plants,” several colocated identical sources connected to one bus and operated by a single cadre of engineers. Neither of these situations can be considered a microgrid: multiple independently operated, cooperating small sources that are *not* colocated.

It is the need for engineers that argues against powering the nation with microgrids. Replacing one percent of the nation’s operational generators

with microsources would require 48,000 new installed 200-kW generators, which would quadrupling the total number of generating units. This implies that widespread deployment can occur only if some managing/operating mechanism can perform day-to-day operations and respond to at least simple contingencies.

10.1.1 *Motivation: a straightforward microgrid use case*

Here we examine an operation that we expect the microgrid to perform on a regular basis. To keep the analysis straightforward, we assume no failures anywhere. Our intent is to motivate a discussion of the systems that make up a microgrid with an eye toward determining what systems are necessary and what they should do. We look at the case where several generators are producing power at an optimum level, the rest are idle, and load is slowly increasing. How is the decision to commit (start up) a generator made?

To make a commitment decision, the agency* in charge of managing the microgrid must have three capabilities: (1) the ability to commit at least one source that is not currently producing—in both the sense of being able to physically start it and the sense of being authorized to do so; (2) a means of determining which of its available generators to commit, and (3) the ability to recognize that the situation implies this would be the right thing to do. All three need to be accomplished in real time, i.e., while they still matter, and each requires additional information.

The first capability requires that the decisionmaker have access to an information structure that contains some “permanent” knowledge (e.g., which generators the decision-maker commands) and information that is continuously updated (e.g., which ones are presently producing power and how much). The content of this information structure allows the decisionmaker to determine whether any of the sources that “belong” to it are capable of being committed.

The main source of additional power in this situation will be idle generators that can be started (although see [Section 10.7.1.2](#), “Storage,” for discussion of another way to provide power). The decision comes down to choosing the most economical power source based on cost of operation, distance from sources, and start-up cost, under any given situation. In [1], we used relatively simple linear programming to decide among production alternatives. A more relevant approach, because we’re likely to be working with a distributed computing plant, would be to use a distributed technique such as particle swarm optimization [3] or a hybrid genetic-Lagrangian search algorithm [4] to determine which unit to commit.

* More than one agency could be involved, but if so, they must act as if they were one agency to achieve the desired result. The ability of multiple agencies to cooperate with the intent of achieving a goal implies a set of ancillary skills discussed further in [1].

The question of *when* to commit a generator is somewhat complex. Several subsidiary pieces of information are needed. These are all collected from the power network or derived from data collected from the power network. The primary information needed to answer this question is the rate at which load is increasing. Given this, it is simple division to compute how long it will be before the capacity of the current set of operating generators is reached: remaining capacity in kW divided by load increase rate in kW/second. Arriving at a useful estimate of rate increase is not straightforward, because load is unlikely to increase uniformly; we need to describe the load time series to estimate the rate of load increase. Given some estimation of the way the waveform varies, we can estimate the time error in our “generator start time” result and start the required generator early by that amount.

In a small system, discontinuous jumps in load will cause instability in the power system, possibly to the extent of causing sources to self-protectively trip. Furthermore, the sudden appearance of a large load may outstrip the capacity of the active generators too quickly for the system to respond, again causing frequency droop and possible generator trip. In this case, we can take any of several courses of action, although some require recognition of this possibility in the design phase:

1. Design loads to limit sudden load increases.
2. Electrically isolate sudden large loads to preserve operational integrity.
3. Include appropriately dispatchable storage to compensate for expected load increases.
4. Use knowledge of the microgrid’s loads and load history to recognize that spinning reserve should be available at certain times.
5. Keep a generator running as spinning reserve, i.e., under no load, at all times.

These are approximately in order of overall operational economic appeal, although capital costs will affect the design as well. The first three options are considered good microgrid design practice for other reasons (see Section 10.7.1.2, “[Storage](#),” in particular).

10.1.2 Microgrid problems and issues

It is easy to adopt a straightforward stance toward managing a microgrid—“do the correct things in the proper order”—but this is difficult if the system is at all complicated (as even small power systems tend to be), if “normal” operations are compromised by unusual events (a relatively common occurrence), or if operators do not have good situation awareness or information about the likely future system state. Operators will be further hampered by the general inability of any group of people—even if they are all in one room*—to agree

* It is tempting to say *especially* if they are all in one room.

on a goal state they will cooperate to achieve (given that an unexpected event has occurred, what should be the state of the power system?) or to construct a series of steps likely to achieve it. In the general case of trying to solve a power system problem, there will be a time constraint—not only do decisions need to be correct, they need to be made quickly. This becomes nearly impossible for human operators communicating by telephone. As a case in point, it is generally agreed that poor situation awareness and slow, uncoordinated response to known faults were instrumental in the August 2003 Northeast Blackout [5].

From first principles, distributed resources are difficult to operate because it is difficult to determine global requirements, to get coherent global status information, to determine what local actions need to be taken with the intent of satisfying global requirements, and to coordinate local actions to meet global requirements. These are required if we are to claim we are running things as a system. This is because operational decisions for the controllable elements of a distributed system must refer to the other components; by definition, a system consists of “interrelated interacting artifacts designed to work as a coherent entity [WordNet].” So, what does it mean to operate noncollocated generators as elements of a system?

In the case of the primary power grid, decisions are based primarily on a complex decisionmaking process that incorporates predicted and actual power needs, market conditions, the state and condition of the power system, the state of the economy, the availability of fuel, the weather, and other influences that are difficult to enumerate, but which everyone agrees affect the system, such as whether the lead facility operator for the power plant is at work when the coolant pump for generator #3 begins vibrating even though it was inspected only last week.

The power principles inherent in operating a microgrid are not particularly different from the principle of operating a large grid. For instance, in [6] we find the following advice: “*Connect moderate loads in each step.* As the frequency derivative is proportional to the load step a connection of a moderate load will result in an acceptable frequency deviation.” This is good advice for any power system, differing for a microgrid only in the meaning of the word *moderate*: the microgrid power waveform is much more delicate than the highly resilient waveform of the primary grid because the capacitance of the microgrid is comparatively small.

10.2 *The microgrid decisionmaking agency and its functions*

Many decisions needed to operate a microgrid are relatively simple and could be made by an automated decisionmaking system, but in any case must be made by some agency. Some decisions, for example abandoning large loads to prevent an impending blackout, are more complex, and given

today's technology, it is difficult to imagine such decisions being made with no humans in the loop. These human decision makers constitute agencies as well. In the remainder of this chapter we refer often to the *decisionmaking agency*, abbreviated DMA. In some cases this may be an automated computational process and in others will require a human, but at this point we are more interested in the decisions themselves and rather less in the exact nature of the entity making them. Figuring out which decisions will be made by what sort of agency depends to a large extent on what technology is available and must wait for implementation.

10.2.1 Essential decisionmaking agency capabilities

A microgrid is made up of a power system (PS) and a management, operation, and control system (MOCS). The PS and MOCS together are "the system."

Microgrid operation, as in any power system, centers on power production and distribution.* Overall, it is easy to say what we want: The system should supply the loads for which it is responsible. In practice, however, the best we can hope for is that the system will *act* to supply its loads. This will often be sufficient because the PS will usually respond as the MOCS expects. In this section we focus on the duties of the DMAs that make up the MOCS.

Firestone and Marnay state in [7] that "Microgrids require control... to make dispatch decisions that achieve system objectives such as cost minimization, reliability, efficiency and emissions requirements, while abiding by system constraints and regulatory rules." Lasseter et al. advise in [8] using an *energy manager* to determine source power and voltage set points based on a desire to "insure that the necessary heat and electrical loads are met by the microsources; insure that the MicroGrid satisfies operational contracts with the bulk power provider; minimize emissions and/or system losses; [and] maximize the operational efficiency of the microsources." See the "Management, operation, and control" section ([Section 10.7.2](#)) for a discussion of the system that would accomplish these functions.

10.2.2 Decisionmaking agency roles

One or more DMAs must accomplish the roles listed in this section for every power system. The more complicated of these have traditionally been accomplished by humans, who over time have come to rely on complex computational processes to make the necessary decisions. Without being specific, it is safe to say that in the future additional functions will be automated.

* Because microgrids are relatively compact, we do not differentiate between power transmission and power distribution.

10.2.2.1 User

Users add and remove devices, propose configuration changes and scenarios, and view information about actual or predicted system state based on real or hypothetical states. Users can also enable and disable individual sources, based on the premise that a user can have information for which the system has no sensors, for example, knowledge that an active generator is in a burning building.

10.2.2.2 System point of contact

A DMA must act as the system point of contact (POC) to inform a user about the current system state through a user interface, to receive requests from the users to perform certain actions (setting maintenance schedules, enabling and disabling devices, etc.), and to act as a conduit to pass user commands into the system.

10.2.2.3 Architectural advisor

A DMA in the architectural advisor role deliberates among alternative planning policies and potential sites to add sources, loads, lines, and sensors to the system. In this capacity the architectural advisor is in a position to assist users in decision making and in establishing official system policies that constrain planned activities.

10.2.2.4 Strategic planner

A DMA in the strategic planner role proposes configurations to the other DMAs in the MOCS, regarding where power and heat is generated, transmitted, and used, and proposes contingency plans.

10.2.2.5 Strategic monitor

A DMA in the strategic monitor role tracks system-wide status and identifies when the power system is and is not operating according to the agreed-upon configuration.

10.2.2.6 Tactical monitor and controller

A DMA in the tactical monitor and controller role is responsible for making tactical control decisions in support of the strategic plan and is responsible for giving advance notice of actions to the strategic monitors.

10.2.2.7 Device protector

A DMA in the device protector role monitors the state of a device and acts to prevent damage to the device, including configuring the device to act properly under contingencies. This includes taking it online and offline, negotiating down times, etc.

10.3 Microgrid organizing principles

Discussions about microgrids generally center on two points: How does an individual microgrid function, and how does a microgrid behave when connected to a much larger power system? A third issue is often ignored: How should microgrids relate to one another? This section addresses the behavior of microgrids connected to other microgrids. We discuss decision making, organization, and operating concerns.

As long as microgrids are operated as ancillary power systems, addressing the question of how they should behave in the context of the primary grid is essential if microgrids are to be a viable way to provide power. Conversely, if microgrids succeed, it is possible that at some point there will be no primary grid, but only other microgrids. Although it is difficult to imagine from today's vantage point, we will certainly find ourselves, as more microgrids reach operational status, somewhere between having only isolated microgrids and having nothing but microgrids. The relationship among microgrids will need to be addressed.

10.3.1 Organization of the microgrid in the context of a primary power grid

Organizational concerns of the microgrid in the context of the primary grid center on two states addressed elsewhere in this chapter in Section 10.8.5, "Operation disconnected from the primary grid," and Section 10.8.4, "Operation connected to the primary grid." To sum up, the organizational concerns in the context of a primary grid are subsumed entirely in how the microgrid is arranged internally in each of these conditions. The microgrid, as is so clearly put in [8], "can be thought of as a controlled cell of the power system within which heat and power are generated for local customers, and generation and load are controlled."

10.3.2 Organizations made up of microgrids

As implied in the previous section, how microgrids communicate with one another is not particularly relevant in the earlier stages of the evolutionary process that ends in a power system consisting entirely of microgrids. Early on, inter-microgrid communication could be handled as necessary on a situational basis. Farther along, however, general principles would be needed to design new microgrids that will need to communicate with existing microgrids at the moment they are activated.

The remainder of this section discusses concerns raised when microgrids need to interact with other microgrids.

10.3.3 *Organizing principles for power systems made up of microgrids*

Cells, globs, and cooperatives are the organizing and scaling concepts for DMAs operating a distributed power system. A *cell* is a set of sources, loads, switches, branches, and buses managed by a single independent DMA. The idea behind a cell is to attach an ontological element—the *cell*—to the smallest unit of DMA responsibility. This places a lower limit on the functionality required of a DMA. Cell extent is limited so that operating a cell under ordinary conditions can be presumed to be straightforward, consisting essentially of maintaining adequate power for the loads that “belong” to the cell.

A cell should be conceptually simple and arranged so a single “tactical” decisionmaker could manage it easily; e.g., its sources should be physically collocated and all on one bus and its loads switched *en masse* as either “critical” or “noncritical.” For instance, a cell might consist of a building, the three generators in its basement, the photovoltaic array and storage batteries on its roof, and an additional building sharing power but with no generating capacity.

When two or more cells are electrically connected to one another, the possibility of trading power arises. Cell interaction policy is the locus for the economic rules that govern power transactions among cells and determines the nature of the organizations the cells can form. A group of electrically connected cells can be organized in two ways: a glob or a co-op. If generators are physically separated from one another to the extent that they cannot readily be operated by a single human technician, and they are not on the same bus, meaning their power is distributed via a network of switches, breakers, buses, and lines and can be routed or islanded, those generators are better characterized as a group of separate cells, either a glob or a co-op, depending on the role of policy in their operation. Globs and co-ops look alike, but the DMAs that manage the components base their behavior on different kinds of policies.

A *glob* is a network of cells in which the member cells have agreed to trade power. A glob differs from an unconstrained collection of connected cells in that the cells in a glob need to be able to execute policy elements that relate to power trading (kilowatt-hour volumes, tariffs, payments, prices, contracts, etc.). DMAs without these elements could not technically form a glob, because they cannot trade power; they lack the knowledge to conduct power business. The nominal behavior of a glob-capable cell is to produce or acquire adequate power for cell loads. Basic standalone cell policy would be supplanted by liveness and safety conditions specifying how and when to trade (“buy power if it is cheaper than it costs you to produce it yourself,” “buy power if you cannot make enough,” etc.). Glob membership is attractive to a glob-capable cell DMA because it can import available power from other cells if in-cell loads cannot be satisfied (or cannot be economically satisfied) by in-cell sources. A cell DMA that is part of a glob would “prefer”—we are

using quotes because the preference could be dictated by policy—to purchase power rather than not satisfying its loads and to spend less on purchased power than on power it makes with its own sources.

A *co-op* is a glob in which the cells obey a common policy governing transactions among the constituents. In particular, policy may dictate that a cell give preferential treatment to loads not its own. Co-op cells would need all the capabilities of glob-capable cells. The primary distinction between a glob and a co-op is that co-op policy is designed to affect power over the entire group of cells—to maximize the probability that critical co-op loads will be satisfied, for example—while there is no such overarching design in a glob. A cell in a glob satisfies its own loads before considering other cells, whereas a cell in a co-op *may*, based on policy, satisfy loads in other cells before it satisfies its own. A glob becomes a co-op when its cells begin to obey a policy that overrides individual cell policies; e.g., “a cell in a co-op shall shed its own noncritical loads when necessary to serve another cell’s critical loads.”

10.4 Behavior of the organizational elements

The default behavior of a DMA managing a cell is *greedy*: it acts to supply in-cell loads. If the sources in a cell can generate more power than required by in-cell loads, a cell DMA may export power if it is part of a glob. For the moment, we assume price conditions are always met; i.e., there might not be enough power from in-cell sources to satisfy in-cell load, but this would not be because of economic reasons. This raises the prospect of a cell/co-op-based power economy that admits auctions, etc., which we leave for future consideration. Note this has no bearing over supply decisions within a cell, because power is always applied preferentially to in-cell loads in the canonical cell. This would not prevent the appropriate tariffs from being collected nor free the DMA from having to know about them.

The default behavior of cells in a glob is unspecified and depends primarily on the commitments, if any, of its constituent cells. Interconnections between cells from two different globs just makes a bigger glob; there is no notion of two globs of cells interacting as globs,* because globs do not have an identity, nor is there any overarching mechanism to differentiate a cell as belonging to a particular glob. Emergent or unstable conditions might result from cells responding to loads in other cells as specified by their individual contracts.

The glob organizing principle exists primarily to describe “life beyond the cell” so that co-ops can be discussed. Co-ops make sense when some loads are more important than others; greedy cells want to satisfy their loads, and if all loads are alike, it does not matter which ones get satisfied first. But in a glob, even if some loads are more important, unimportant loads might be satisfied while important loads nearby remain unsatisfied (because, say, generators have failed in that cell) because of default greedy cell behavior.

* This does not limit interaction among individual cells from different globs in any way.

The default behavior of cells in a co-op is defined by the policy of the co-op and can be made to appear altruistic; for example, policy might dictate that a hospital without power be supplied by its neighboring cells, even if they must shed advertising and entertainment loads to do so. Exactly what loads are shed under what conditions is determined by co-op policy and enforced by the DMAs.

We suggest that a co-op should be, without loss of generality, either an actual multicomponent co-op or a primitive cell following co-op policy. This is conceptually appealing because, if a cell and a co-op can act alike, a group of co-ops can form a larger co-op, so that cooperation among co-ops would be much like—the appropriate abstraction barrier would allow us to say *exactly* like—cooperation among cells.

This resolves, at least conceptually, the *scaling* question: It is all well and good to conduct a laboratory proof-of-principle experiment involving a few dozen DMAs, but how should several thousand DMAs be organized to manage a large distributed power grid? We respond that they should be organized as co-ops whose members are themselves co-ops.

This requirement implies new features. A DMA representing a cell would provide an abstracted public picture of the cell. It would not be necessary to distinguish* a cell trying to join a co-op from a genuine co-op acting as if it were a cell trying to join a co-op. Fundamentally, cell capabilities and co-op capabilities need to be separated from one another and made available as separate packages. This means that a co-op could then take on the cell-level capabilities needed to be a member of a co-op, which would enable it to participate in co-op operations as a member element, i.e., as if it were a cell.

It also means that co-ops and cells could be members of the same co-op. Connections out of a co-op would be managed as if they were connections out of a cell, and co-ops could negotiate with one another and with cells in a larger glob or a larger co-op. Co-op policy would constrain the interaction.

Although a cell DMA would have detailed information about cell contents and a co-op would generally be a much larger entity than a cell, this would all be hidden by the abstract interface. A large power system consisting of many thousands of microgrids would be organizationally fractal, in the sense that each of the pieces of the organization would be very similar to the organization itself; co-ops would be made up of smaller co-ops, etc., with the most primitive elements being single cells following co-op rules.

10.5 Policy for microgrid operation

Control and *management* are two distinct sets of functionality, and both are necessary to operate distributed infrastructures. Many control tasks can be

* To be fair, an entity with whom one was interacting would likely identify itself as a cell or co-op, but to an implementor, the protocols, messages, and content information would be the same.

performed by wholly automated mechanisms based on local conditions, but management requires knowledgeable oversight by autonomous, situationally aware entities that can communicate with one another and act quickly and with assurance.

There are three operation time scales of interest in the electric power domain:

1. Near-instantaneous (a second or less)—This is the time frame in which the system has to handle minor or routine load fluctuations, presumably by having generators preconfigured to respond in a certain way to local load changes. In emergency scenarios this is the approximate time frame in which breakers would need to be thrown; for example, to cut off a noncritical load in order to mitigate the problem of a local generator failure. The DMAs are not expected to respond in this time frame, due to the need to communicate with one another, but can observe its effects. We assume nearly all actions taken in this regime will be accomplished completely automatically.
2. Short-term (seconds, up to a minute)—This is the time frame in which conditions that require no more than reporting and invocation of existing group plans can be addressed. In practice, an event is observed and reported to tactical peers, and the DMAs to whom the event has been reported take predetermined steps. This is the notional process for responding quickly when a generator fails: a contingency plan for the scenario will have been distributed *a priori*, and the failure event alone will trigger the designated response. System designers must accommodate short-time-scale events by building in automatic protection.
3. Long-term (minutes to years)—This is the time frame in which plan generation and distribution can occur, enabling changes to the power system to be handled according to policy, assuming the system has been stabilized in the near-instantaneous or short-term time frames. Short-term responses should already be in place for specific contingencies, and these might buy time for the planning process. Replanning, agreeing upon the final plan, distributing this, and putting the plan into action is expected to take several seconds, if not minutes.

We would like the system DMAs, regardless of their locations, to rapidly and automatically (i.e., without the benefit of explicit real-time instructions) make decisions based on policy, in other words, to *manage* the power system. The decisions to be made are, essentially, which resources to use to satisfy load requirements under a given set of conditions, what actions to take under various conditions of rapid change, and with whom and in what manner to interact in order to make these decisions with the appropriate authority.

DMA-enforced policy orientation gives human policymakers the capacity to define an explicit policy that the DMAs will endeavor to enforce in the face of contingencies, threats, and unpredicted behavior by human operators.

A DMA of this sort acts either in direct response to an authorized entity or because the data on which it is basing its decisions is consistent with premises that taking the indicated action is consistent with policy and will increase the likelihood that the future state will be a desirable one.

10.6 A DMA policy outline

1. Human interaction

- a. When an entity with the appropriate authority issues a command that the MOCS is able to obey, the MOCS should obey the command and report that it has done so.
- b. When an entity with the appropriate authority issues a command that the MOCS cannot obey, the MOCS should report that it cannot obey the command and say why.

2. Source control

- a. When total load is in excess of the maximum that the MOCS can supply in its current configuration, transition to a new configuration, if any, that can supply adequate power.
- b. Check first for stored configurations designated as capable for equivalent loads.
- c. If there are no appropriate stored configurations, search for some.
- d. When choosing a configuration to supply power, prefer configurations that:
 - i. Generate equivalent power at lower cost
 - ii. Differ less from the preceding configuration
 - iii. Have a lower system-wide average fraction of rated power being carried by all lines
 - iv. Can supply a thermal load that occurs within the appropriate time interval
- e. If it appears that total load will at some future time exceed the maximum that the system can supply in its current configuration, search for other configurations in which the projected load can be satisfied. Record each such configuration in conjunction with associated load information and other information needed to select among configurations. Denote the configuration as capable of satisfying its associated load.

3. Load control

- a. Maintain service to all loads.
- b. When load must be shed, shed noncritical loads before critical loads.
- c. Prefer supplying critical loads to shutting down sources for maintenance.
- d. Prefer shutting down sources for maintenance to supplying non-critical loads.

- e. If it appears that projected load will soon be greater than the system can supply, determine the order in which to shed loads and which loads should be shed.
4. Maintenance scheduling
 - a. Take components offline as required by their maintenance schedules.
 - b. As a component nears 90% of its MTBF, assign it high priority for being taken offline.
 - c. Take any component that exceeds 90% of its MTBF offline for maintenance (may be overridden by 3c).
5. Distribution path
 - a. When a distribution path fails, compute the power flow for the remaining network and determine whether any of the remaining lines will be forced to carry more power than their rated capacities.
 - b. If a line is carrying more than its rated capacity, and there exists some other line not in service whose placement into service will allow a new load distribution where no lines are overloaded, place that line into service. If more than one such line exists, choose the line for which the system-wide average fraction of rated power being carried by all lines is lowest when the system is placed in the suggested configuration.
 - c. When the steady-state power flow through any generation, transmission, or distribution element exceeds its maximum steady-state rating, transition to a new configuration that will bring all system power flows into specification. Such reconfiguration may include a prioritized shedding of heat loads or those that can be served by other heat sources.

10.7 The subsystems that make up a microgrid

A microgrid is made up of a power system (PS) and a management, operation, and control system (MOCS). The PS and MOCS together are “the system.” The power system is made up of systems for production, storage, and distribution. The MOCS incorporates systems for communication, cyber security, decisionmaking and optimization, modeling and prediction, planning and monitoring, and computation.

10.7.1 Electric power subsystems

In order for a series of noncollocated power sources to be operated as a microgrid, they must be in the same circuit, since sources in different circuits can have no effect on one another, and there would be no point in considering them concurrently. Any load served by any source in the microgrid is thus also in the circuit with all the sources while it is being served, as are all the other electrical sensing, control, and management devices. In general, it will

be possible to separate any of these elements from the rest by switches or breakable connections.

10.7.1.1 *Production*

Microgrid sources are generally assumed to be small relative to primary power generators, in the range of 200 kW or less. A paper mill or semiconductor plant consuming several tens of megawatts would need hundreds of 200-kW generators. The interconnection hardware, control systems, and human staff needed to manage such a large number of sources would straightforwardly render the power enterprise uneconomical compared to relatively simple connection to the main power grid, especially since the main grid already exists. This is one reason there are not more microgrids: they are obviated by easily available power from the primary grid. Conversely, if we were designing a power system from scratch we probably would not build the power system we have today, which was built during a time of high regulation and cheap fuel.

To be fair, a microgrid for, say, a 75-MW paper plant would probably not be constructed using 200-kW generators. One of the primary economic benefits of a microgrid is that the sources can be operated at optimum levels. A 75-MW paper plant would be operated around the clock with a minimum power load of a few tens of megawatts. There would be no real point in satisfying that load with a hundred 200-kW generators. The appropriate number of generators for a paper mill is around half a dozen, plus or minus a few. A 2004 study of actual and potential microgrid sites [9] reports almost a hundred paper mills operating generators in the United States. Of these, 76 are operating three to six generators, with only 16 operating seven or more. This allows the individual power sources to be operated optimally (for the most part; see [Section 10.7.5](#), “Decisionmaking and optimization”) and shut down for maintenance while providing flexibility for variation in power requirements. Nominally we would rotate duty so that each generator would have a duty cycle less than 100% to allow for maintenance. We also need to know something about the statistics of the 75-megawatt load. Among other things, is 75 MW the peak load or the average load? We do not want to curtail operations too often because the power system does not have sufficient capacity, but we must be cognizant of the fact that satisfying 100% of the load 100% of the time may be very expensive.

Power production needs to be dispatchable, so that we can start production when we need power and stop when we do not. The energy inputs to solar and wind generation are free, which makes them economically attractive even though the capital cost per watt is high. Unfortunately, however, they are not dispatchable; whether they are making power at any given moment has little relationship to load. It is advised that, if renewable energy sources are being considered, site load profiles be compared to the site’s renewable energy profile to determine whether renewable sources are feasible. It is also worthwhile to consider that, since wind and solar power, in particular, may

not be available at any given moment, they cannot be relied upon to provide power at any given moment, and their capacity must be, in essence, duplicated by dispatchable sources, especially for high-priority loads. This means that they can supplant energy sources but cannot displace capital plant expenditures.

10.7.1.2 Storage

Storage is used to compensate for the inability of system sources to provide adequate power for some period of time. In a system that incorporates non-dispatchable sources such as solar photovoltaics or windmills, either dispatchable sources or storage can be used to compensate for variation in the essentially uncontrollable production. In a small system, however, the inability of system sources to respond in a timely manner virtually dictates some storage to compensate for source response lag.

In general, the smaller the overall system, the greater the requirement for storage. This is discussed at some length in [8]. Essentially, the inertia of a smaller power system is less, all other things being comparable. The transient effects of switching a source or load of a given size in or out will be proportionally larger for a small power system and consequently are more likely to cause self-protective shutdown of other system devices. In a large power system, the capacitance of the system as a whole can absorb and compensate for variation caused by the behavior of individual elements. This is also true for a smaller system, but for phenomena of a much smaller magnitude.

Additionally, components of the system need time to respond to variation. A rotating-mass source can require several minutes at startup before it can provide power. Since such a startup would be in response to a power requirement, total system production may very well be less than total system load during this startup period. In a small power system, this power deficit can cause frequency and voltage droop to the extent that, again, individual devices will self-protectively shut themselves down. This will in turn cause further instability and power deficit and can lead to cascading shutdown that will cause the entire power system to black out. It is important to note that the behavior of individual devices is the same whether part of a small or a large power system, but any effects are proportionally greater in the smaller system and therefore more likely to be problematic.

The primary storage decisions that the MOCS needs to make are when to store power in the storage subsystem and when to take stored power out of the storage system.

10.7.1.3 Distribution

One of the benefits of serving loads with a microgrid is that power need not be transmitted over the long distances assumed for regional power grids, and transmission losses are therefore lower. In general, the distances over which power needs to be moved in a microgrid are consistent with distances at the distribution level of a regional power grid.

It would be helpful in terms of overall reliability and ease of operation if we could arrange our microgrid distribution system so that any generator could supply any load. We could then literally start up any generator when we needed power. This will not always be the case because of congestion, i.e., the distribution system may not be of sufficient capacity to allow any producible amount of power to flow anywhere in the microgrid. In a distributed power system the DMAs should cooperatively compute the optimum power flow using a distributed technique like ant colony optimization, as in [11], or particle swarm optimization, as in [12].

10.7.2 Management, operation, and control

The overarching task of managing a microgrid is to operate the generators so that the loads that “belong” to the microgrid are satisfied (loads are not normally associated with sources, but they are in this formulation so that a responsible entity can be named). This can be complicated for even a simple power system, as discussed in [1], especially one without a high-inertia reference power source with which the sources can synchronize. It is particularly important to balance production and load at any given moment in a small power system (discussed further below in Section 10.8.5, “Operation disconnected from the primary grid”), because relatively small differences between production and load can cause the sources to self-protectively halt.

The complex of elements that accomplishes the management, operation, and control functions for a microgrid should be thought of as a system, because its mission is to operate the microgrid in a unified way. The MOCS is made up of DMAs and the communication network, computers, sensors, relays, etc. needed to supply data to, execute the required functionality of, and carry out the commands of the DMAs.

10.7.3 Communication

Microgrid communication occurs in two senses: electrical communication and information communication. In the electrical domain, power system events have system-wide effects that are communicated throughout the electrical system, with the system itself as the medium. In the information domain, all information that is needed to operate the power system with the intent of achieving or maintaining a desirable nonlocal state, and which is not already present in the power system, must be communicated explicitly.

The failure of a generator in Seattle can be detected in San Diego as a slight reduction in the frequency of the alternating current. The frequency drops because the burden of serving the system’s loads falls to the generators remaining in service, most of which are of the rotating mass type. This increases the load at each generator, which is then slowed slightly, which reduces the frequency. The governing control system of each generator

responds independently by accelerating the generator to bring the system frequency back up to the nominal level.

Control of the system frequency, voltage, and phase thus relies in large part on the system frequency, voltage, and phase. In essence, the system provides its own reference. Since the system is never completely shut down, the reference is always available. This is one reason it takes weeks to return a large blacked-out power system to service: The reference is gone and must be carefully preserved as sources and loads are returned to function. Similarly, the microgrid has no readily available reference, but must somehow provide one.

This brings us to information communication: The reference standard will not necessarily be available via the power system, but may need to be provided through a networked digital communication system, along with all the other information needed by the system that cannot be gained by observing the power waveform. In particular the power system itself cannot generate information about which nonoperating generator should be committed to service to provide power for an upcoming power need.

The communication system is needed to carry two kinds of information: *commands* and *power system condition information*, whether directly from sensors or operational devices (a relay, for example, might send a digital signal that it had complied with a command) or from other DMAs (who could use their information storage and computational resources, for example, to compute the mean power output of a source during a period of time).

It is becoming increasingly common to transmit digital control system information via the Internet, for essentially the same reason that an electric power user connects to the primary power grid: the service is available, cheap, and meets the demand; all that is needed is connection. Bandwidth may become a problem, but the Internet as it stands today is not bandwidth-limited for the relatively low data rates needed for intrasystem communication. On the other hand, cyber security is an issue.

10.7.4 Cyber security

The cyber security of the communication system used by the MOCS is of increasing concern. It is worth pointing out that the primary security concern in this regime is that many operators do not have a security policy and do not follow best practices. The most effective advice for producing an effective security plan is to have one. Significant and plentiful information is available on securing infrastructure control systems. We recommend [13–15] for advice on effective control system security practice.

10.7.5 Decisionmaking and optimization

The decisionmaking facility is the primary way a microgrid is distinguished from a set of independently operating sources. This implies the DMA needs a continuous supply of relevant, timely information on which to base its decisions.

For example, consider the policy statement: "When load is expected to exceed the amount of power that can be produced by the operating sources at their current setpoints, and the setpoints of at least some sources are not at their maxima, raise the setpoints of the appropriate sources to the appropriate levels in an economically optimum manner." This implies that the MOCS needs to maintain data structures containing up-to-date information about:

1. Which sources are operating and which are not, their electronic addresses, their current setpoints, their current production levels, their maximum production levels, and their production increase rates
2. Current demand, its rate of increase, and its expected value at future points
3. The algorithms to use to determine which sources will have their setpoints increased and the new levels
4. The commands to be executed to enable these setpoint increases
5. The connectivity and interfaces needed to execute these commands within the PS

Each policy statement to be upheld by the system requires similar analysis. The MOCS will require a data structure containing the results of this policy analysis.

The commands that can be given to the devices that make up the power system are simple, consisting primarily of "on" and "off." A switch or relay is "off" when open and "on" when closed, and a transmission line can be treated similarly, although in most cases the line will simply be "on"—able to transmit power up to its capacity and regarded as "off" only under failure conditions. It is reasonable to allow a complex subsystem consisting of a transmission line, power conditioning equipment, protective relays and command-operated switches to be regarded as a single element that is usually "on" but can be "off" under certain failure or control conditions.

A generator is somewhat more complicated, but only a little; in an efficiently operating microgrid, most of the generators producing power at any given moment will be operating at their most efficient settings and can be regarded as "on."

In all but the most idealized conditions, since power production and consumption must never be allowed to differ by more than a small amount,* the output of a few—ideally only one—must be allowed to fluctuate in response to changes in load. We want to minimize the number of generators operated in this fashion because they are, by definition, operating inefficiently.

The selection of which generator(s) are allowed to fluctuate is of particular interest, since it depends on the statistical properties of the system loads (fast, large load fluctuations will require more generators in "float" mode).

* Small standalone power systems, and some large ones, often incorporate storage to "soak up" and emit power to accommodate source response time.

This requirement points out the need to incorporate the statistics of the power system into the microgrid decisionmaking process. Each set of statistics requires sensors, network bandwidth, storage, and computation. In this case we would relate the number of generators allowed to float to the stochastic parameters of the load time series.

In any case, the MOCS needs to be able to issue four commands to a source: *stop*, *start* (“begin producing power according to your setpoint”), *adopt a setpoint* (“produce power at the specified level”), and *float* (“vary the amount of power you produce according to changes in load”). Both of the latter two are subject to preemption based on observed conditions, in the sense that we want to retain self-protective local control for all sources, just as in any other power system, so they will not be damaged by power surges, rapid frequency variation, or any other power system behavior that would cause a standalone generator to trip off.

10.7.6 Modeling and prediction

Throughout we have spoken of whether what is going to happen is what the MOCS expects. One essential implication of this sort of statement is that the MOCS have expectations. The MOCS will attend to its expectations in two different ways: First, if the expected state is undesirable, the MOCS should recognize this and act to keep the power system from entering that state. Second, the MOCS should be able to sense whether its expectations are consistent with reality and respond accordingly. Note the MOCS knows about reality only what its sensors tell it.

The MOCS should have a model of the power system at its disposal in order to be able to predict its future state. The time scale of interest is the shorter end of the long-term scale—tens of minutes—since the MOCS cannot react to faster events.

10.7.7 Planning and monitoring

The MOCS needs to determine its actions based on the difference between what it thinks the future state will be and what it would like the future state to be. Under desirable conditions, this should be based on efficiency. Achieving a given state requires planning, which means essentially determining a series of steps likely to achieve a desired state given an initial state. The general planning cycle is:

1. Determine an achievable goal state.
2. Determine a sequence of steps that should result in the goal state given the initial state.
3. Predict the sequence of intermediate states that should result from applying the steps in the plan.
4. Begin executing the steps.

5. Compare the actual state after each step with the appropriate predicted post-step state.
6. Replan when significant discrepancies arise.

10.7.8 Computation

Computation can be regarded as a service in the sense that the choice of computational engine is essentially an implementation detail. Using grid services to accomplish computation is discussed in [16]. We can, however, say something about the computational requirements of the microgrid system. Two kinds of computation are needed to operate the microgrid: mathematical and symbolic. Mathematical computation is needed to answer questions of the sort: “Is the load increasing or decreasing? At what rate? Is it about to exceed the amount of power that can be produced by the generators that are currently running?” Symbolic computation is needed to answer questions of the sort: “Is a human operator attempting to override a policy-derived parameter setting? If so, under what conditions is the override allowed? Are those conditions currently in force?” Mathematical computation and symbolic computation can appear identical, depending on viewpoint and level. The distinction between “mathematical” and “logical” resides in the interpretation of the results of the actions needed to specify the behavior of the system.

Execution of the various distributed algorithms, encryption schemes, communication protocols, and user interaction will require a certain level of around-the-clock dedicated CPU time.

10.8 Microgrid use cases

It is safe to say that, in almost every case where electric power is needed, the user will choose to connect to the primary grid if it is possible. This is because it is cheap, safe, convenient, reliable, and does not require power engineers on staff. On the other hand, primary grids have been known to fail and can be damaged by disasters and adversarial attacks. Microgrids are inherently resistant to widespread or cascading failures because they can operate standalone. This enables microgrids to *island*—isolate themselves from the primary grid.

A microgrid will be operated either standalone or connected to a large power system, normally the primary power grid. It is not hard to imagine a microgrid meant to serve an isolated set of loads that will never be connected to the primary grid. This kind of microgrid is relevant to military uses (shipboard power systems are microgrids), space exploration, and small communities “off the grid.”

The desirable situation, however, seems to be a collection of power sources organized into microgrids which are in turn organized into cooperatives of some kind.

10.8.1 Operation under normal conditions

Operation under normal conditions consists in the most general way of countering influences that would push the system out of a desirable state. The MOCS should recognize when the power system is in a desirable state and act to preserve it. Desirable system conditions are called *liveness conditions*. The MOCS also needs to recognize when changes in the power system indicate that the system will leave the desirable state if trends continue.

10.8.2 Operation under anomalous, unexpected, and failure conditions

The MOCS should recognize that the system is in an undesirable state and act to correct this. Undesirable system conditions are called *safety conditions*. As in the simple use case presented at the beginning of the chapter in “Motivation: a straightforward microgrid use case” (Section 10.1.1), there are two essential elements to dealing with unexpected events: first, realizing that something is not right; second, deciding what to do about it.

10.8.3 Operation at system capacity

Primarily, the decisionmaker needs to know whether any of its generators are not yet producing power, and if so, one or more would be started to accommodate the predicted increase in load. If not, steps must be taken to curtail the load, which cannot exceed production. This is simple arithmetic and counting, as long as which generators are producing power—and which are not—is known.

Sources operating can produce some additional power, although at the point where the capacity of an operating generator is about to be exceeded, this is not much and is used only rarely. A generator operating at its optimal point and whose power output is controllable can generally be made to produce more power, but they are seldom operated this way because it wears them out quickly, and the regime beyond the optimal point is uneconomical at best. Producing power in this regime would be used only under unusual circumstances, if ever.

Data giving the optimum power production level of each generator, the amount by which this can be exceeded, and the effect this would have on maintenance would be stored associated with the generator.

10.8.4 Operation connected to the primary grid

10.8.4.1 Central control when connected to the main grid

When a set of microsources is run with a central controller and tied into the main power infrastructure, it is like a big power plant in every way but size. Although outwardly comparable to a microgrid, this kind of installation is

not operated in a distributed fashion and is, conceptually speaking, only a little more complex to operate than a single generator.

Control is centralized either in the sense of having all sensor and control channels connected into a common control site, or in the sense of having the generators all reporting to and obeying a single controller over a network. There are technical issues with ensuring that the generation of power is coordinated—having suitably configured droop rates, deciding when to turn generators on or off to follow load, establishing a policy to determine the generation in response to local load, etc.—but the agency controlling the phase and frequency of our local system can use the bulk power supply phase and frequency as reference, and the detail that the power is being generated by an array of microturbines or other such devices can be largely abstracted away, since only minor variation is caused by the slight differences in transmission path between the individual generators and the loads.

This satisfies the general desire of the utility operator that a power resource be simple to operate. A study by the Consortium for Electric Reliability Technology Solutions (CERTS) [8] states: “the [electric power] utility does not want to be burdened with additional control issues. . . . The first goal is for the microgrid system to appear to the utility only as a controllable load. Local electric utilities are justifiably concerned about uncontrolled voltage regulation.”

Such centrally controlled microplants can be used to ensure continuity of power when access to bulk power is interrupted, or can serve as local supply for peak-shaving, load reduction, or power conditioning (in which case access to bulk power becomes the reserve, rather than the local microsources). Typically such plants are run as fixed, homogenous installations; often they are engineered to meet specific requirements associated with a known load’s characteristics, and new generation capacity is either not added, or added to match expansion of the load. Because the plant and load are most likely owned by the same organization, expansion and reduction of the facility probably does not have to be coordinated between very many parties.

10.8.4.2 Distributed control when connected to the main grid

This applies to a collection of microsources that are independent of one another, not collocated, operating autonomously, and connected independently to the bulk power network, which acts as a phase and frequency reference and provides virtually unlimited capacitance. The overt question is how much power each of the generators must produce under the assumption that excesses or deficits will be accommodated by the bulk power network. In general these microsystems are not all owned by the same owner, which means generation and compensation have to be negotiated. When the bulk power system is the national infrastructure (relatively few producers; many consumers), this may be effectively brokered by the bulk power network, but for systems that are distributed with the intent of supplying a certain set of distributed loads, and for systems that must maintain certain conditions (for security, safeguards, etc.), collective internal negotiation will be a greater

issue. This case, and the corresponding disconnected case, will be of interest to facilities that want the benefits of a locally maintained microgrid but want the system to handle changing topological configurations (the connection and disconnection of portions of the facility from one another, for example) gracefully and do not want the liability of a central point of failure in the power supply.

10.8.5 Operation disconnected from the primary grid

The smaller the power system, the more important it is to balance power production and power use at any given moment, because relatively smaller—and therefore more common—fluctuations in either production or load can cause the sources to self-protectively halt, or “trip.” Fluctuations in voltage or frequency that would be damped out in a large power system, because of the inertia inherent in the large rotating mass generators ordinarily employed, can cause tripping and cascading failure in a microgrid. Figure 10.2 illustrates the effect on the three-phase voltage of a simulated power system when an operating 35-kW source is connected to the system and begins providing power. Although stability quickly returns, the effect is dramatic. In short, the stability of the microgrid requires somewhat closer attention, because common events cause effects that are proportionally much larger in the microgrid than they would be in a large power system.

10.8.5.1 Central control when disconnected from the main grid

When a centralized microgrid has its connection to bulk power severed, or if it is constructed with no connection to a bulk power network, it must operate in such a way that the load is met within predefined tolerances (or selectively shed), and the generators must maintain phase and frequency in synch with one another and within the tolerances required by the load. Because command is central, an “official” reference phase and frequency can be maintained. For homogenous plants the difficulties associated with centrally controlled power systems have been addressed, but unplanned addition or

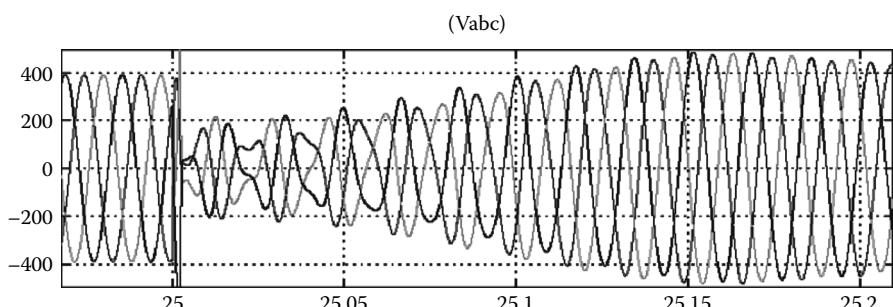


Figure 10.2 Momentary three-phase voltage instability caused by generator cut-in.

removal of capacity remains difficult. This is referred to as “plug and play” (often shortened to “p&p”), and Lasseter [18] has shown that, given certain sensing and logic capability, a collection of plug-and-play sources* can be operated as a microgrid with only casual oversight.

The nominal p&p source makes decisions based on the condition of the power waveform at the point where the source is connected to the power grid. These independent sources cannot, however, make decisions about ensemble operation. In essence, with additional power conditioning hardware, sensors, and logic, a source can be given the capability of matching its phase and frequency to the values of the primary circuit, but cannot decide when to allow another source precedence. Although information about the entire power system is present in the waveform, the state of the system that produced the waveform cannot be reconstructed.

In small to mid-scale cases, the valid “island” when bulk power is severed will be determined in advance based on the centrally owned or managed facility, the centrally controlled plant, the load the plant is designed to support, and the requirements of how the load must be supported in this condition. Islanding portions of the national power grid remains a far more difficult problem, not primarily because of the technical challenges of coordinating the significant alterations in load and generation as the network undergoes substantial topological changes, but because of the legislative, procedural, and financial issues that come into play for a collaboratively managed system that is run at every moment under the auspices of prearranged contracts.

10.8.5.2 Distributed control when disconnected from the main grid

This is the most difficult case. Phase and frequency have to be controlled collaboratively, since there is neither a main grid nor a central authority available to supply frequency or phase reference, and power generation has to be negotiated without the benefit of a large inertial bus to absorb or supply the necessary excess. These systems could be flat or hierarchically organized, with local power subsystems ganging together to form semiautonomous agencies that interact with one another at an amortized level, leaving the internal decisionmaking entities to negotiate how the details will be accomplished in support of the higher-level contracts made with other organizations. The multi-DMA system discussed in Reference 2 was designed to accommodate this case.

10.8.6 Transition from the connected to the disconnected state

When primary grid instability and/or power condition threaten to cause protective response in the microgrid, the MOCS may wish to disconnect from

* Note the general power system is already considered plug and play with respect to *loads*, in that proper operation is mostly about managing sources in the face of load variation.

its primary grid. Before disconnecting, the MOCS must determine the disparity between the power being generated within the island and the power being used and decide on a response to the predicted disparity. If too much power is being generated, the MOCS needs to shut down and/or lower the setpoints of some generators; if not enough, the MOCS needs to shed some load. If the predicted primary grid disturbance will occur too soon to have time for this, it may be necessary to disconnect without proper condition, allow the protective response of the microgrid, then restore or blackstart, as discussed in [19].

The problem of how to split an operating grid into islands to best serve current loads with operating sources is addressed in [20–22], but each determines where to split the grid to best serve demand. We have the inverse problem. We know where to split the grid (i.e., at the microgrid boundary), and we need to know how to shed load or increase production so that the split will not cause the microgrid to go black. We must approach the problem as in [23] and decide when to split in the only place the split can occur and minimize the impact of the resulting disturbance.

10.9 Conclusions and future work

Penetration of microgrids requires automation of their day-to-day operation. Advances in power control technology have enabled the essential connect-supply-shutdown-disconnect operations without significant higher-level concern. What is needed is a unified logic process to apply the appropriate algorithms for deciding which sources should be providing power, how the power system should be reconfigured to isolate faults, what steps to take to recover from upsets, how to restore the system to operation after primary failures, what to do to halt or slow cascading failure, and when to separate from the primary grid when blackout threatens. In addition, the system must respond to requests from humans. The means to accomplish these goals is a distributed set of automated decisionmaking agencies (DMAs) organized into management, operating, and control systems (MOCS) with the necessary information to make the appropriate operational decisions and the sensing and effecting mechanisms needed to carry them out. The DMA MOCS should be built with reference to a policy that it is equipped to in the context of a set of operational cells organized into cooperatives.

References

1. Energy Information Administration. 2006. Total Capacity of Dispersed and Distributed Generators by Technology Type, 2005 Electric Power Annual, Energy Information Administration. <http://www.eia.doe.gov/cneaf/electricity/epa/epat2p7c.html>.
2. Phillips, L. R., Link, H. E., Smith, R. B., and Weiland, L. A. 2006 *Agent-Based Control of Distributed Infrastructure Resources*. Sandia Technical Report SAND2005-7937.

3. Ting, T., Rao, M., and Loo, C. 2006. A novel approach for unit commitment problem via an effective hybrid particle swarm optimization. *IEEE Trans. on Power Systems* 21:411–418.
4. Valenzuela, J., and Smith, A. E. 2002. A seeded memetic algorithm for large unit commitment problems. *J. Heuristics* 8:173–195.
5. U.S.-Canada Power System Outage Task Force. 2004. *Final Report on the August 14, 2003 Blackout in the United States and Canada: Causes and Recommendations*. U.S.-Canada Power System Outage Task Force, North American Electric Reliability Corporation. <http://www.nerc.com/~filez/blackout.html>.
6. Agneholm, E. 1996. The restoration process following a major breakdown in a power system. Tech. Rep. 230L, Chalmers University of Technology, Göteborg, Sweden.
7. Firestone, R. and Marnay, C. 2005. Energy manager design for microgrids. Tech. Rep. LBNL-54447, Lawrence Berkeley National Laboratory.
8. Lasseter, R., Akhil, A., Marnay, C., Stephens, J., Dagle, J., Guttromson, R., Meliopoulos, A., Yinger, R., and Eto, J. 2002. Integration of distributed energy resources: The CERTS microgrid concept. Tech. Rep. LBNL-50829, Consortium for Electric Reliability Technology Solutions.
9. Stevens, J. 2005. Characterization of microgrids in the United States. Technical report, Resource Dynamics Corporation. http://www.electricdistribution.ctc.com/pdfs/RDC_Microgrid_Whitepaper_1-7-05.pdf.
10. Boyes, J. and Menicucci, D. 2007. Energy storage: the emerging nucleus of America's energy surety future. *Distributed Energy*, Vol. 5, January/February 2007. http://www.erosioncontrol.com/de_0701_energy.html.
11. Bouktir, T. and Slimani, L. 2005. Optimal power flow of the Algerian electrical network using an ant colony optimization method. *Leonardo Journal of Sciences* 7:43–57, Amici Publishing House.
12. Wang, C.-R., Yuan, H.-J., Huang, Z.-Q., Zhang, J.-W., and Sun, C.-J. 2005. A modified particle swarm optimization algorithm and its application in optimal power flow problem. *Proc. Int. Conf. on Machine Learning and Cybernetics* 5:2885–2889.
13. Berg, M. and Stamp, J. 2005. A Reference Model for Control and Automation Systems in Electric Power. Sandia National Laboratories Report SAND 2005-1000C. http://www.sandia.gov/scada/documents/sand_2005_1000C.pdf.
14. Kilman, D. and Stamp, J. 2005. Framework for SCADA Security Policy. Sandia National Laboratories Report SAND 2005-1002C. http://www.sandia.gov/scada/documents/sand_2005_1002C.pdf.
15. Phillips, L., Baca, M., Hills, J., Margulies, J., Tejani, B., Richardson, B., and Weiland, L. 2005. Analysis of Operations and Cyber Security Policies for a System of Cooperating Flexible Alternating Current Transmission System (FACTS) Devices. Sandia National Laboratories Technical Report SAND2005-7301. http://www.sandia.gov/scada/documents/sand_2005_7301.pdf.
16. Phillips, L. R. 2007. Managing microgrids using grid services. In *Proc. IEEE Int. Conf. on System of Systems Engineering*, April 2007, pp. 1–5.
17. Kueck, J. D. et al. 2003. *Microgrid Energy Management System*; ORNL/TM-2002/242, Oak Ridge National Laboratory. <http://www.ornl.gov/sci/btc/apps/%20Restructuring/ORNLTMM2002242rev.pdf>.
18. Lasseter, R. H. and Piagi, P. 2004. Microgrid: a conceptual solution. In *Proc. Power Electronics Specialists Conference*.
19. Lopes, J. A. P., Moreira, C. C. L., and Resende, F. O. 2005. Microgrids black start and islanded operation. In *Proc. 15th Power Systems Computation Conf.*

20. Liu, W., Cartes, D. A., and Venayagamoorthy, G. K. 2006. Application of particle swarm optimization to split power grids into islands. In *Proc. IEEE Int. Conf. on System of Systems Engineering*.
21. Sun, K., Zheng, D.-Z., and Lu, Q. 2006. Searching for feasible splitting strategies of controlled system islanding. In *IEE Proc. on Generation, Transmission, and Distribution* 153:89–98.
22. Zhao, Q., Sun, K., Zheng, D.-Z., Ma, J., and Lu, Q. 2003. A study of system splitting strategies for island operation of power system: a two-phase method based on OBDSS. In *IEEE Trans. on Power Systems* 18:1556–1565.
23. Sodzawiczny, G., and Sowa, P. 1999. Multicriterial adaptive load shedding algorithm. In *Int. Conf. on Electric Power Engineering, PowerTech '99*, p. 192.

chapter eleven

An integrated intelligent decision support system based on sensor and computer networks

*Qishi Wu, Mengxia Zhu, Nageswara S. V. Rao,
S. Sitharama Iyengar, Richard R. Brooks, and Min Meng*

Contents

11.1	Introduction.....	282
11.2	Related work.....	283
11.3	System framework.....	285
11.4	Technical solutions	287
11.4.1	Sensor deployment.....	289
11.4.1.1	Sensor deployment problem formulation	289
11.4.1.2	Probabilistic sensor detection model	290
11.4.1.3	An approximate solution using genetic algorithm.....	291
11.4.1.4	Performance evaluation	293
11.4.2	Sensor data routing	294
11.4.2.1	Adaptive and energy-efficient sensor data routing based on spin glass theory	294
11.4.2.2	Data routing in mobile agent-based distributed sensor networks	297
11.4.2.3	Data routing in multi-sink sensor networks.....	300
11.4.3	Network mapping for optimal computing pipeline configuration	301
11.4.3.1	Cost models of pipeline and network components	302
11.4.3.2	Mapping problem formulation	302
11.4.3.3	Optimal linear pipeline configuration (OLPC)	304
11.4.4	Sensor data fusion	306
11.4.4.1	Problem formulation	306
11.4.4.2	Threshold-OR fusion method	307
11.4.4.3	Simulation results	311

11.5 Conclusion	313
Acknowledgments	314
References	314

11.1 Introduction

Wireless sensor networks are becoming increasingly pervasive in many military, civil, agricultural, and industrial applications ranging from mission-critical homeland security defense, battlefield assessment, health care improvement, environment surveillance, climate research, disaster recovery, ecological forestry and wildlife monitoring, to manufacturing process control. The large amount of data produced in sensor networks combined with complex mathematical models in various applications is far beyond the processing and computing capabilities of a sensor node, or even a high-end PC workstation, and therefore must be transmitted to computer networks such as the Internet for distributed processing where abundant system resources such as computing power, storage space, and network bandwidth are widely deployed. In this chapter, we present a general framework of IDSS-SC, an intelligent decision support system that integrates both sensing and computing subsystems.

The development of the IDSS-SC framework is driven by the needs for capturing, storing, transmitting, sharing, processing, fusing, and analyzing large-scale observational sensor data in an integrated environment. Rapid advances in sensor and network technology have enabled distributed sensor networks to evolve from small clusters of large sensors to large swarms of micro-sensors, from fixed sensor nodes to mobile sensor nodes, from wired communications to wireless communications, from static network topology to dynamically changing topology, and from homogeneous sensor networks to heterogeneous sensor networks. These advances in various technological areas have also brought new challenges of managing a colossal amount of multidimensional sensor data of high redundancy in a bandwidth-limited, power-constraint, unstable, and dynamic sensor network environment. The success of large-scale computation-intensive sensor network applications requires the development of an integrated network and system architecture where data processing is moved from a single base station to a set of powerful computing nodes distributed in wide-area computer networks.

To make the IDSS-SC more concrete, we consider a real-life application where a dirty bomb is reported to have been dropped in a residential area with dense population. In response to this terrorist incident, a number of sensor nodes of different types are deployed either randomly or strategically within the affected region to create a wireless sensor network and collect various measurements including the level of chemical pollution or radiation, acoustic and seismic signals, temperature, atmospheric pressure, wind power, and other environmental parameters. A complex mathematical model composed of air absorption, ground attenuation, wind effect, and many

other factors is constructed to simulate the explosion process and determine the pollution diffusion pattern, which is used to guide the evacuation of residents in the region. The complexity of the diffusion model and the enormous volume of real-time sensor readings render a single computer inadequate to perform data processing and explosion simulation in a timely manner. In the IDSS-SC framework, the sensor data collected using energy-efficient routing methods is transmitted to wide-area computer networks composed of supercomputers, PC clusters, and many other powerful computing engines. The computation-intensive data fusion and explosion simulation tasks are partitioned and distributed to these high-performance computing facilities, which generate and send the simulated pollution dispersion map to the command center to assist in the evacuation of residents.

Terrorist detection and monitoring in homeland security defense is another example using such an integrated IDSS-SC. A number of cameras or other imaging sensors are installed at a custom clearance gate to capture and record from different angles the physical features of every passenger entering the custom. The captured images are continuously transferred to a set of strategically selected computing nodes that perform a sequence of computing tasks including image augmentation, feature extraction and detection, facial reconstruction, pattern recognition, data mining, and identity matching. The individual detection results based on a set of images collected from various angles are fused to reach a final decision to assist in the investigative actions.

The structure of IDSS-SC is divided into three logical components: (1) sensing field covered by wireless sensor networks, (2) cyberspace based on computer networks, and (3) command control center for intelligent decision making. The general framework of IDSS-SC incorporates various subsystems for sensor deployment, data routing, distributed computing, and information fusion. The integrated system is deployed in a distributed environment composed of both wireless sensor networks for data collection and wired computer networks for data processing. For these subsystems, we formulate the analytical problems and develop approximate or exact solutions. These subsystems are implemented and evaluated through either experiments or simulations in various application scenarios. The extensive results demonstrate that these component solutions imbue the integrated system with the desirable and useful quality of intelligence.

The rest of the chapter is organized as follows. We introduce the related work in Section 11.2. The detailed system framework of IDSS-SC is described in Section 11.3. The technical solution to each subsystem is presented in Section 11.4. We conclude our work in Section 11.5.

11.2 Related work

We conduct a survey on the related work of the subsystems of IDSS-SC in the areas of sensor deployment, data routing, network mapping for distributed computing, and information fusion.

Sensor deployment problems with practical considerations have been studied in depth for decades in a variety of scenarios. In the adaptive beacon placement, the strategy is to place a large number of sensors and then turn off some of them based on their localization information. In this context, Bulusu et al. [3] presented an adaptive algorithm based on measurements by considering the evaluations for spatial localization using radio frequency-proximity. In a related area, Guibas et al. [19] presented a unique solution to the visibility-based pursuit evasion problem in robotics applications. In wireless sensor networks with the global knowledge of node positions, Meguerdichian et al. [27] used a Voronoi diagram to compute the maximal breach path for the worst-case coverage and Delaunay triangulation to compute the maximal support paths for the best-case coverage. Voronoi diagrams were also used by Wang et al. to discover coverage holes in [39], where several sensor deployment protocols were designed to provide high coverage by moving sensors from densely deployed areas to sparsely deployed areas. Both static and mobile sensor deployment schemes were considered to optimize sensing coverage and secure connectivity. Many research efforts were devoted to the investigation of sensor deployment strategies that provide sufficient coverage for distributed detection. Martinez and Bullo [26] studied optimal sensor placement and motion coordination strategies for mobile sensor networks in a target tracking scenario. To improve the integrity of sensed data and minimize the energy consumption for data communications, Ganesan et al. [17] tackled the combined optimization problem of sensor placement and transmission structure for data gathering.

Data routing is another focus area attracting increasing attention of researchers in sensor networks. Many routing protocols have been proposed for sensor networks, such as directed diffusion [21], two-tier data dissemination (TTDD) [44], mesh [43], and low energy adaptive clustering hierarchy (LEACH) [20]. Several recent works [7,8,12–16,22,25,28,40,45,47] addressed security problems in sensor networks. The existing routing protocols fall into two categories, namely, table-driven (proactive) routing and source-initiated on-demand (reactive) routing. Proactive sensor data routing maintains up-to-date routing tables, which provide paths from each node to every other node in the network. Any topology change is propagated throughout the network to update the routing table of each node. The storage requirement of routing tables and the transmission overhead of topology changes are the main drawbacks of this category of protocols. Reactive methods run on the demand of a source node to a destination node; thus, the overhead for control packet transmission is tremendously reduced compared with the table-driven routing. This reactive routing process starts with the route discovery procedure, followed by the route maintenance procedure until either the destination becomes inaccessible or the route is no longer desired [32]. Most protocols use the shortest path as the only performance metric in data routing without accounting for other considerations, including power consumption. For ad hoc sensor networks, routing protocols must deal with

some unique constraints such as limited power, low bandwidth, high error rate and dynamic topology, which motivate us to explore routing protocols that are energy efficient, self-adaptive, and error tolerant.

A considerable amount of research efforts have been devoted to the design of scheduling algorithms in various disciplines to achieve the best resource utilization by carefully mapping computing modules onto network or processor nodes [1,9,10,18,23,34]. Task scheduling and network mapping continue to be the focus of attention in distributed computing due to their theoretical significance and practical importance, especially as the grid computing technology prevails [4–6]. A grid scheduling algorithm, called *Streamline* [2], is developed for placing a coarse-grain dataflow graph on available grid resources. This scheduling heuristic is specifically designed to improve the performance of streaming applications with various demands in grid environments. Kwok et al. proposed a Dynamic Critical-Path (DCP) scheduling algorithm [24] to map task graphs with arbitrary computation and communication costs to a distributed network environment consisting of fully connected identical nodes. Chen et al. proposed and evaluated a run-time algorithm for supporting adaptive execution of distributed data mining on streaming data [11]. However, few previous works have addressed the global optimization of the decomposition and mapping of an application computing pipeline under varying network conditions in a distributed environment. In addition, existing approaches that map computing modules to network nodes are mostly empirical and require manual configuration. Their implementations are typically limited to a traditional client and server mode with no intermediate nodes considered.

Information fusion is critical to the performance of sensor network applications. Many existing non-model-based or model-based fusion methodologies are derived from some variants of decision rules such as Voting, Bayes Criterion, Maximum a Posterior Criterion (MAP), and Neyman-Pearson [30,31,33,35–38]. Data fusion is in general categorized as low-, intermediate-, or high-level fusion, depending on the stage where the actual fusion process takes place.

The general framework of IDSS-SC we proposed integrates novel solutions in the aforementioned areas such as sensor deployment, data routing, network mapping, and information fusion for intelligent decision support. Through IDSS-SC, we not only provide the decision maker the best guidance for action, but also ensure the accuracy of each final decision fused from individual observations.

11.3 System framework

The application domains of IDSS-SC span from military operations, homeland and global security defense, to environment monitoring. The success of these large-scale sensor network applications requires transmitting large

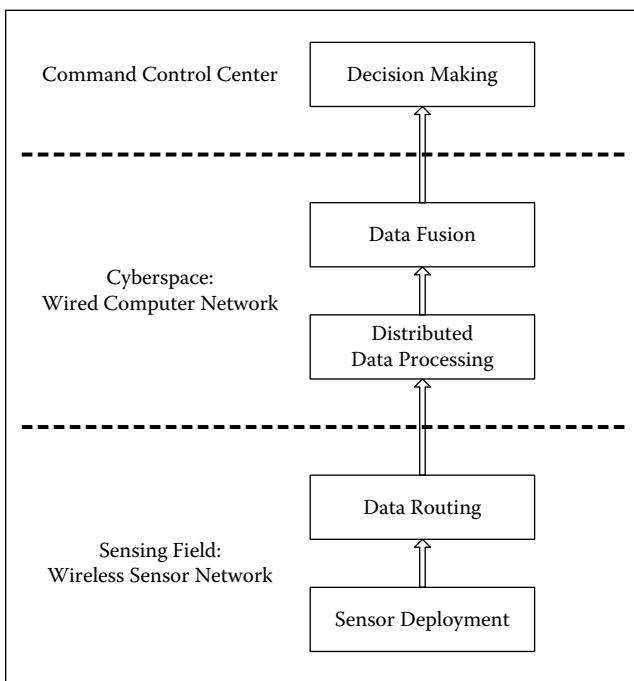


Figure 11.1 Intelligent decision support system integrating sensing and computing subsystems: system components and design process.

amounts of sensor data to computer systems, accessing remote databases, and generating a quick response at the time of critical incidents.

The component subsystems and design process of the proposed IDSS-SC framework are illustrated in Figure 11.1. From a spatial and temporal perspective, the data flow in IDSS-SC runs through the following three major areas:

- Sensing field covered by wireless sensor networks
A large number of sensor nodes on the order of hundreds, thousands, or even millions may be deployed in vast geographical areas to collect multimodal environmental measurements. Appropriate sensor deployment schemes must be determined in different applications for various deployment purposes such as target detection, localization, and tracking. The large amount of sensor data also requires energy-efficient, low-latency, and fault-tolerant routing algorithms to gather individual sensor readings at a processing element or a cluster head, where the data is preprocessed and transmitted to wide-area computer networks for further processing.
- Cyberspace based on computer networks
Many computing-intensive sensor network applications feature a so-called computing pipeline that consists of a sequence of computing

- modules.* The large amount of data generated by wireless sensor networks is fed into this computing pipeline to produce required results. Due to the disparity and heterogeneity of system resources distributed in computer networks, it is critical to find an optimal pipeline partitioning and network mapping scheme that achieves the best system performance and resource utilization. A reliable and fault-tolerant data fusion algorithm then integrates the computed local results from the data collected by individual sensors or separated sectors of sensors to reach a global optimal solution for final decision support.
- Command control center for intelligent decision making
Critical situations in military operations or homeland security require real-time decision making in the presence of imminent threat. The command control center makes intelligent decisions based on the final results from sensor data processing and information fusion.

The integrated sensing and computing system is deployed in a distributed environment composed of both wireless sensor networks for data collection and wired computer networks for data processing. The IDSS-SC system architecture with a general computing pipeline is illustrated in [Figure 11.2](#). There are four virtual component nodes in IDSS-SC, i.e., client or user at the command center, central management (CM), dynamic data source (DS) collected by and transmitted from sensor networks, and computing service (CS) distributed in computer networks, which are connected together by network links.

In the sensing field, the sensors deployed in the surveillance region form a network of certain architecture to collect environmental measurements, which are transmitted via energy-efficient routing mechanisms to wired computer networks for data processing and analysis. On the other hand, in the cyberspace, a client may initiate a particular surveillance, detection, or tracking task by sending a request containing environmental attributes, predicate constraints, sampling interval, computational models, and other control parameters to a designated CM node. CM determines the best system configuration based on the global knowledge of data sources and accessible distributed resources such as computing node capabilities and current network link bandwidths. The pipeline of the selected computing method is strategically partitioned into groups and mapped onto an appropriate set of CS nodes to execute the computing modules. A final decision fused from local results based on individual sensor data is sent to the command control center for final intelligent decision making.

11.4 Technical solutions

In this section, we discuss the technical solutions in IDSS-SC: (1) sensor deployment strategy based on a two-dimensional genetic algorithm to

* Computing modules are also referred to as subtasks or stages in certain contexts.

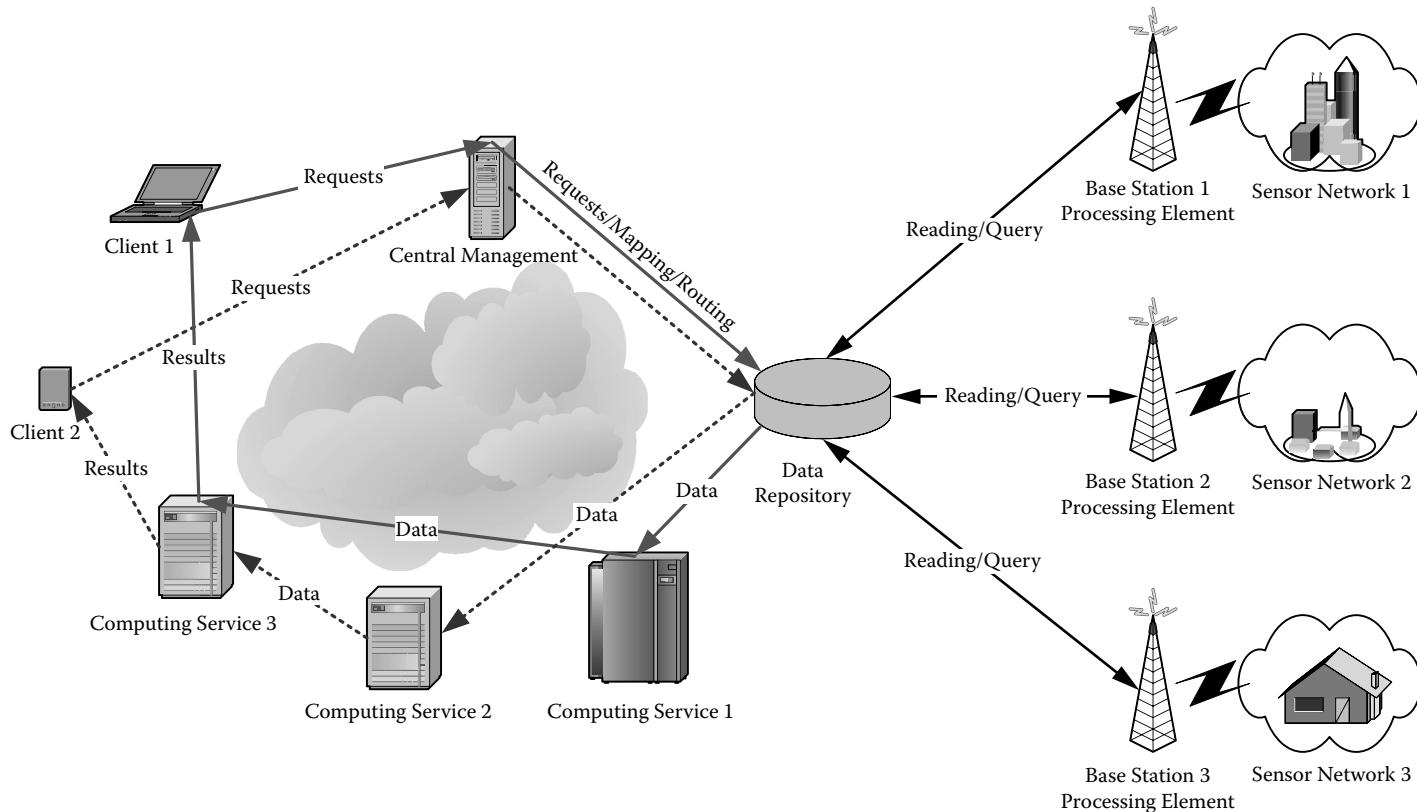


Figure 11.2 IDSS-SC system architecture.

achieve maximum coverage with cost constraints; (2) data routing scheme to achieve maximum signal strength with minimum path loss, energy efficiency, and fault tolerance; (3) network mapping method based on dynamic programming to assign computing modules to network nodes for sensor data processing with minimum end-to-end delay; and (4) binary decision fusion rule that derives threshold bounds to improve system hit rate and false alarm rate. These subsystems are implemented and evaluated through either experiments or simulations in various application scenarios. The extensive results demonstrate that these component solutions imbue the integrated system with the desirable and useful quality of intelligence.

11.4.1 Sensor deployment

One practical goal of sensor deployment in the design of distributed sensor systems is to achieve an optimal monitoring and surveillance of a target region. The optimality of a sensor deployment scheme is a trade-off between implementation cost and coverage quality levels. We consider a probabilistic sensing model that provides different sensing capabilities in terms of coverage range and detection quality with different costs. A sensor deployment problem for a planar grid region is formulated as a combinatorial optimization problem with the objective of maximizing the overall detection probability within a given deployment cost. This problem has been shown to be NP-complete in [42], and we present here an approximate solution based on a two-dimensional genetic algorithm. The solution is obtained by the specific choices of genetic encoding, fitness function, and genetic operators such as crossover, mutation, and translocation for this problem. Simulation results of various problem sizes are presented to show the benefits of this method as well as its comparative performance with a greedy sensor placement method.

11.4.1.1 Sensor deployment problem formulation

A planar surveillance region R is to be monitored by a set of sensors to detect a target T if located somewhere in the region (our overall method is applicable to the three-dimensional space). The planar surveillance region is divided into a number of uniform contiguous rectangular cells with identical dimensions. A circular coverage area can be approximated by a set of cells within a certain maximum detection distance of the sensor. The main reason we discretize the 2-D space is to facilitate an efficient approximation of the sensor's sensing behavior and the region's coverage probability in the later computation of the genetic algorithm-based solution. When the ratio of the sensor detection range to the cell dimension is very large, the sensor coverage area made up of many tiny rectangular cells will approach a circle.

Assume there are q types of sensors and a sensor of the k -th type is denoted by S_k for $k \in \{1, 2, \dots, q\}$. There are N_k sensors of type k . A sensor S can be deployed in the middle of cell $C(i, j)$ to cover the discretized circular area $A_S(i, j)$. A sensor S_k deployed at cell $C(i, j)$ detects the target $T \in A_{S_k}(i, j)$.

according to the probability distribution $P\{S_k | T \in A_{S_k}(i, j)\}$ while incurring the cost $w(k)$. A *sensor deployment* is a function \mathfrak{R} from the cells of R to $\{\epsilon, 1, 2, \dots, q\}$ such that $\mathfrak{R}(i, j)$ is the type of sensor deployed at the cell $C(i, j)$; $\mathfrak{R}(i, j) = \epsilon$ indicates no sensor is deployed, i.e., $w(\epsilon) = 0$. The *cost of a sensor deployment* \mathfrak{R} is the sum of cost of all sensors deployed in region R , which is given by:

$$Cost(\mathfrak{R}) = \sum_{C(i,j) \in R} w(\mathfrak{R}(i, j)) \quad (11.1)$$

The *detection probability* $P\{\mathfrak{R}|T \in R\}$ of deployment \mathfrak{R} is the probability that a target T located somewhere in region R will be detected by at least one deployed sensor. The sensor deployment problem (SDP) is formally formulated as follows:

Given a surveillance region R , cost budget Q , q types of sensors, and N_k sensors of type k , find a sensor deployment \mathfrak{R} to maximize detection probability $P\{\mathfrak{R}|T \in R\}$ under the constraint $Cost(\mathfrak{R}) \leq Q$.

Informally, we are required to locate the sensors of various types on the grid points to achieve a maximum detection probability while keeping the deployment cost under a specified budget. The *decision version* of the SDP asks for a deployment with detection probability at least A under the same cost condition, i.e., $P\{R|T \in R\} \geq A$ and $Cost(\mathfrak{R}) \leq Q$.

11.4.1.2 Probabilistic sensor detection model

We consider that each sensor type is specified by its local detection probability of detecting a target at a point within its detection region. With regard to a sensor, detection is more likely as a target approaches the sensor. The cumulative detection probability of a sensor for a region is computed by integrating its local detection probability for detecting a target as the target gets close to the sensor, passes near the sensor, and then leaves it behind.

We construct a sensor performance model specified by a Gaussian cumulative detection probability. Given the detection probability density function $p_{S_k}(x)$ for a sensor of type k , the detection probability $P\{S_k | T \in C(i, j)\}$ for cell $C(i, j)$ is given by:

$$P\{S_k | T \in C(i, j)\} = \int_{x \in C(i, j)} p_{S_k}(x) dx \quad (11.2)$$

After obtaining the individual detection probabilities for all the cells covered by sensor S_k , we employ Gaussian function to compute the cumulative

detection probability. The Gaussian cumulative detection probability approximating a real sensor detection performance is defined by:

$$P(S_k, \tau, \alpha_{S_k}) = P\{S_k, \tau, \alpha_{S_k} \mid T \in A_{S_k, \tau}\} = e^{-\frac{\tau^2}{2\alpha_{S_k}^2}}, \tau \in [0, d_{S_k}] \quad (11.3)$$

where τ is the distance from the target to the sensor. The sensor detection quality coefficient α_{S_k} determines the shape of the detection probability curve. Distance τ is in the range between 0 and the maximum detection distance d_{S_k} .

11.4.1.3 An approximate solution using genetic algorithm

Genetic algorithm is a computational model that simulates the process of genetic selection and natural elimination in biological evolution. It has been widely used to solve the combinatorial and nonlinear optimization problems with complex constraints or nondifferentiable objective functions. The computation of genetic algorithm is an iterative process toward achieving the global optimality. During the iterations, candidate solutions are retained and ranked according to their quality. A fitness value is used to screen out unqualified solutions. Genetic operations of crossover, mutation, translocation, inversion, addition, and deletion are then performed on those qualified solutions to create new candidate solutions of the next generation. The above process is carried out repeatedly until a certain stopping or convergence condition is met. For simplicity, a maximum number of iterations can be chosen to be the stopping condition. The variation difference of the fitness values between two adjacent generations may also serve as a good indication for convergence. To utilize the genetic algorithm method, various parts of the SDP must be mapped to the components of the genetic algorithm.

11.4.1.3.1 Genetic encoding for sensor deployment. Since a candidate solution to the SDP requires a two-dimensional sensor ID matrix, we adopt a two-dimensional numeric encoding scheme to make up the chromosomes instead of the conventional linear sequence. We construct a sensor ID matrix for a possible sensor deployment scheme, where each element in the matrix corresponds to a cell within a surveillance region. An empty value ϵ in the matrix indicates that its corresponding cell has no sensor deployed in and should be covered by the sensors deployed in its neighborhood area. Furthermore, we arrange q types of available sensors in the following order:

$$d_{S_1} / w(s_1) \geq d_{S_2} / w(s_2) \geq \cdots \geq d_{S_k} / w(s_k) \geq \cdots \geq d_{S_q} / w(s_q)$$

Note that d_{S_k} and $w(s_k)$ are the maximum detection distance and cost of sensor of type k , respectively. Besides, each sensor type has a sensor coverage coefficient that determines the variation of its detection capability along the

target-sensor distance. The rank of ratio is used to decide the probability of sensor type selected during the population initialization as well as the addition operation.

11.4.1.3.2 Fitness function. We construct the fitness function from the objective function as:

$$f(\mathfrak{R}) = P\{\mathfrak{R} \mid T \in R\} + g \quad (11.4)$$

where g is the penalty function for overrunning the constraint, which is defined by:

$$g = \begin{cases} 0, & Cost(\mathfrak{R}) \leq Q \\ \delta * E_m * (Q - Cost(\mathfrak{R})) / Q & Cost(\mathfrak{R}) > Q \end{cases} \quad (11.5)$$

where δ is a proper penalty coefficient and is set to 100, and

$$E_m = \max_k \left\{ d_{S_k} / w(S_k) \right\}.$$

11.4.1.3.3 Selection of candidates. The selection operation, also called reproduction operation, retains good candidates and eliminates others from the population based on the individual fitness values. It aims to inherit good individuals either directly from the last generation or indirectly from the new individuals produced by mating the old individuals. The frequently used selection mechanisms include fitness proportional model, rank-based model, expected value model, and elitist model.

In our implementation, the survival probability B_i for each individual (solution) \mathfrak{R}_i is computed based on the following fitness proportional model:

$$B_i = f(\mathfrak{R}_i) / \sum_{j=1}^M f(\mathfrak{R}_j) \quad (11.6)$$

where M is the population size. The hybridization individuals are produced according to the selection rule so that the individual with bigger B_i has a higher probability to survive.

11.4.1.3.4 Implementation of genetic operators. The solution set of each new generation after initial population is generated as follows. Randomly select two hybridization individuals \mathfrak{R}_u and \mathfrak{R}_v , and combine them to get two other individuals \mathfrak{R}_u' and \mathfrak{R}_v' of new generation by using combinatorial rules of crossover, mutation, inversion, translocation, addition, and deletion. Some of these genetic operators are carried out on a two-dimensional

basis. Except for crossover, all the other operators operate on only one parent solution. This process continues until all M individual solutions of the new generation are created.

11.4.1.4 Performance evaluation

We present simulation results of the approximation solution based on genetic algorithm (GA) and compare its performances with those of a greedy solution based on uniform placement (UP) of sensors. The UP method employs two greedy algorithms in terms of selecting the sensor that provides the maximum ratio of detection range to unit price and deploying them in the optimal positions. Specifically, we distribute a maximum set of "best" sensors under a given budget limit in the surveillance region in an optimal way such that the overlapping areas between adjacent sensors are minimized.

We first consider a small surveillance region of 50×50 cells with five different types of sensors. The investment limit is set to be 1800 unit expense, and the maximum generation number is set to be 200. Upon the completion of optimization process, the GA achieves a suboptimal deployment scheme with detection probability of 94.52% for the surveillance region within the investment budget. A 3-D display of the local coverage probabilities of 50×50 cells is plotted in Figure 11.3.

A number of experiments with larger surveillance regions and more sensor types with different parameters are conducted to compare the performance of genetic algorithm (GA) and uniform placement (UP). The average

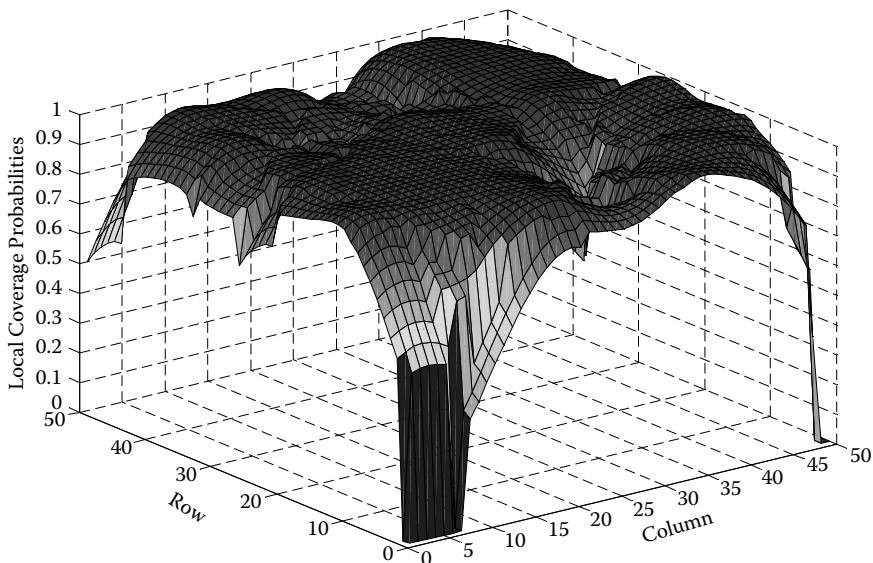


Figure 11.3 3-D display of the local coverage probabilities for a surveillance region with 50×50 cells based on genetic algorithm.

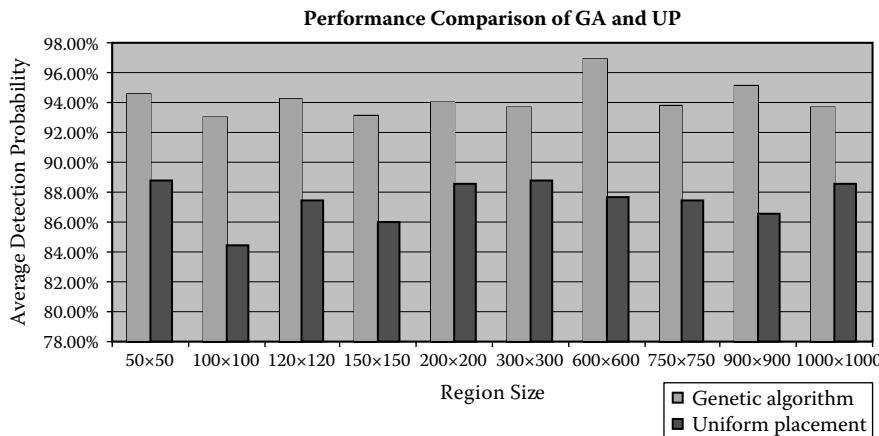


Figure 11.4 Performance comparison of GA and UP for various region sizes.

detection probabilities computed by both GA and UP for each region size are plotted as clustered columns in Figure 11.4. In all the simulation cases we studied, the GA achieved significantly higher probability of detection than the UP under the given cost bound. Also, these experimental results show that our GA-based approach is able to scale well with the region size, number of sensor types, and various constraints on the investment budget.

11.4.2 Sensor data routing

Once deployed, the sensors self-organize to form a wireless sensor network (WSN) and collect a large amount of data, including enormous redundancy and outliers. We present several routing mechanisms used to transmit the collected data to a gateway or base station that is connected to wired computer networks. The dominant problem in WSN is the limited battery power of each sensor node, and the most energy-consuming task in WSN is the transmission of sensor data. We propose energy-efficient sensor data routing methods that reduce data transmission energy to prolong the network lifetime.

11.4.2.1 Adaptive and energy-efficient sensor data routing based on spin glass theory

Distributed sensor networks deployed in unstructured environments that do not support wireless communication infrastructures always feature unpredictable ad hoc network topology, which brings new challenges to traditional best-effort routing for sensor data integration. We present an adaptive and energy-efficient routing protocol based on spin glass theory that exploits global optimization by simple local rules as instantiated by the phenomenon of ferromagnetism in physics. The Ising system is one of the earliest cellular

automata models under investigation. Individuals in the Ising system are spins, which can be viewed as little magnets. Spins occupy fixed positions in a one-, two-, or higher dimensional settings, and the polarity is the only variable feature. The probability for a spin to take each of the possible directions is defined by the Boltzmann probability distribution function.

This proposed routing algorithm aims to find dynamic routes from sensor nodes to a data sink with minimized energy consumption and fair usage of sensor nodes, achieving scalability and efficiency through localized communications and distributed computations. We employ a cellular automata modeling tool in the implementation of the routing algorithm and present the simulation results for an urban warfare scenario of 256 nodes. Results show that our protocol can support quick route discovery in chaotic urban region and is highly efficient in terms of energy consumption and load sharing.

In the spin glass model, a potential field characterizing minimal transmission energy cost to data sink is established by propagating energy computation from the data sink to its neighboring cells, which in turn serve as energy computation components for their neighboring cells, and so on. The same process is repeated until the entire area is fully resolved for energy value. Generally, there is only one data sink whose energy value is set to be zero, while all the rest of the cells have infinite initial energy values. Once the energy value computation starts, every node simultaneously looks around its adjacent nodes and adds the neighbor's energy value to the corresponding weight (minimal transmission energy needed for that link), which can be dynamically updated. The lowest one is chosen as its energy value representing the lowest energy cost to data sink. This step is similar to the classical Lee algorithm, which was used to find a shortest path between two terminals if such a path exists. However, in real applications, the potential field as well as the weights is dynamic, and the potential field is subject to frequent change due to topological disturbance. Furthermore, a kinetic factor is introduced in our implementation in order to simulate the high error rate in sensor networks and to better inhibit oscillation. Based on the current potential field, kinetic factor, node failure probabilities, and any possible topological disturbance, spin direction of each node can be determined as the next hop along the route to a data sink.

The Cantor tool is a simulator based on the generic automata with interacting agents (GAIA) model. It is capable of modeling dynamic and discrete (in both time and space) event systems consisting of a large number of interacting individuals. In our simulation, we assume that each cell is occupied by a sensor node. Neighborhood radius is one by default and is of Moore type. Initial weights of nodes are set to be identical for simplicity. Primary battery power is user-specified, and battery is subject to detritions based on the volume of communication, computation, and transmission as evolution proceeds.

An example of a 16×16 grid is used to investigate our spin glass theory. As shown in [Figure 11.5](#), black blocks stand for wall, white blocks stand for sensor nodes, green blocks stand for open doors, yellow blocks stand for

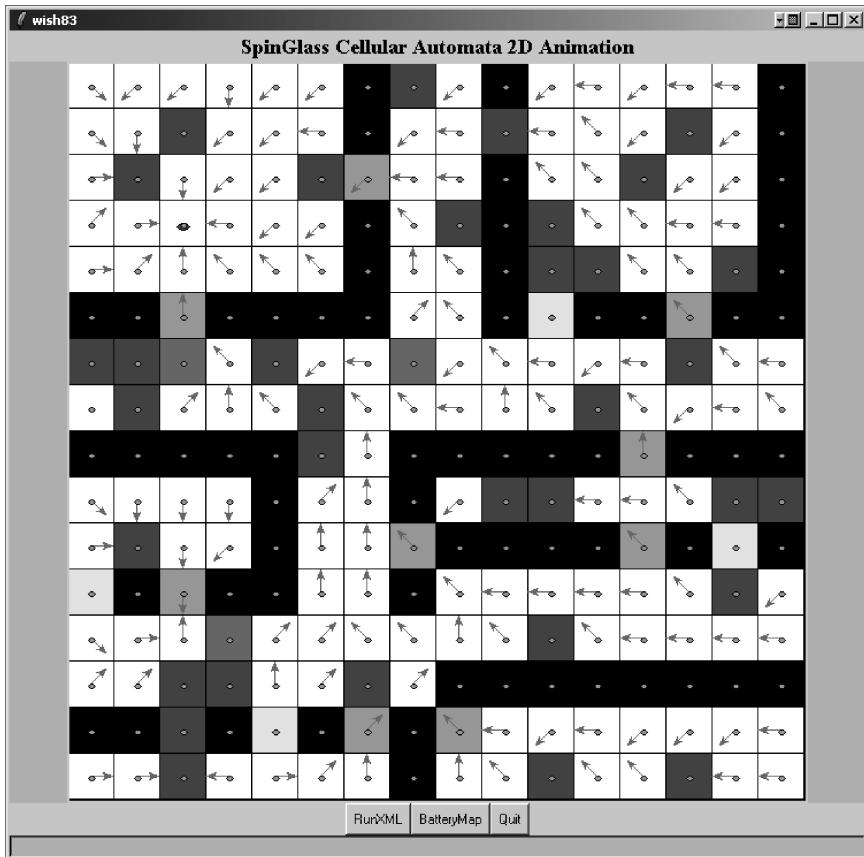


Figure 11.5 A spin glass routing example on 16×16 2-D grid.

closed doors, red blocks stand for obstacles, and blue blocks stand for sleep nodes. Environmental disturbances include opening a door, closing a door, placing new obstacles, and removing obstacles.

In the spin glass-based routing mechanism, we take into consideration minimal power consumption and democratic policy besides path length to make the routing protocol more power aware. The temperature variable not only controls the macro configuration of the system, but also tunes the system's ability to adapt. Low-temperature systems adapt slowly and dampen oscillations, while high-temperature systems adapt quickly but can become unstable. Simulation results with cellular automata modeling tool also show that this routing protocol is distributed in computation, localized in communication, and highly adaptive to environmental disturbance. Furthermore, this routing protocol can be easily modified to adapt to a real ad hoc sensor network scenario. In addition to a flat structure, a hierarchical structure is also applicable to our model with necessary extension.

11.4.2.2 Data routing in mobile agent-based distributed sensor networks

In conventional sensor fusion architectures, all the sensor data is sent to a central location where it is fused. But the transmission of noncritical sensor data in military distributed sensor network (DSN) deployments increases the risk of being detected, in addition to consuming the scarce resources such as battery power and network bandwidth. To meet these new challenges, the concept of *mobile agent-based distributed sensor networks* (MADSNs) was proposed by Qi et al. in [29], wherein a mobile agent selectively visits the sensors and incrementally fuses the appropriate measurement data. We consider the problem of computing a route for a mobile agent in a distributed sensor network. The order of nodes visited along the route has a significant impact on the quality and cost of fused data, which, in turn, impacts the main objective of the sensor network, such as target classification or tracking. We present a simplified analytical model for a distributed sensor network and formulate the route computation problem in terms of maximizing an objective function, which is directly proportional to the received signal strength and inversely proportional to the path loss and energy consumption. This problem has been shown to be NP-complete [41], and we present here a genetic algorithm to compute an approximate solution by suitably employing a two-level encoding scheme and genetic operators tailored to the objective function. We present simulation results for networks with different node sizes and sensor distributions, which demonstrate the superior performance of our algorithm over two existing heuristics, namely, local closest first (LCF) and global closest first (GCF) methods.

Mobile agent routing algorithms can be classified as dynamic or static routing according to the place where routing decisions are made. A dynamic method determines the route locally on the fly at each hop of the migration of a mobile agent among sensor nodes, while a static method uses centralized routing, which computes the route at the processing element (PE) node in advance of mobile agent migration. For a sensor network application at hand, one has to judiciously choose between the dynamic and static methods. For example, it might be sufficient to use a static routing for target classification, but a target tracking task might require a dynamic routing to follow a moving target.

The routing objective is to find a path for a mobile agent that satisfies the desired detection accuracy, while minimizing the energy consumption and path loss. The objective function for the mobile agent routing problem is based on three aspects of a routing path P : energy consumption $EC(P)$, path loss $PL(P)$, and detected signal energy $SE(P)$ as follows:

$$O(P) = SE(P) \left(\frac{1}{EC(P)} + \frac{1}{PL(P)} \right)$$

wherein the terms $SE(P)$, $EC(P)$, and $PL(P)$ are assumed to be “normalized” to appropriately reflect the contribution by various loss terms. To facilitate the design of a genetic algorithm, we define a *fitness function* based on the above objective function as follows:

$$f(P) = O(P) + g$$

where g is the penalty function for overrunning the constraint and is defined by:

$$g = \begin{cases} 0, & SE(P) \geq E \\ \delta \cdot (SE(P) - E) / E, & SE(P) < E \end{cases}$$

where E is the desired detection accuracy or signal energy level, and δ is a properly selected penalty coefficient.

We propose an event-driven adaptive method to implement a semidynamic routing strategy based on a two-level genetic algorithm. We convert the problem parameters into the individuals made up of genes in the genetic domain and employ a two-level encoding to adapt the genetic algorithm to the mobile agent routing problem in MADS. The first level is a numerical encoding of the sensor ID label sequence L in the order of sensor nodes being visited by mobile agent. This sequence consists of a complete set of sensor labels because it takes part in the production of a new generation of solutions through the genetic operations. The second level is a binary encoding of the visit status sequence V in the same visiting order, where ‘1’ indicates ‘visited’ and ‘0’ indicates ‘unvisited.’ If a sensor is inactive, its corresponding bit remains ‘0’ until it is reactivated and visited. Masking the first level of numerical sensor label sequence L by the second level of binary visit status sequence V results in a candidate path P for mobile agent. These two levels of sequences are arranged in the same visiting order for the purpose of convenient manipulations of visited/unvisited and active/inactive status in the implementation of the genetic algorithm. The number of hops H in a path P can be easily calculated from the second level of binary sequence as follows:

$$H = \sum_{i=0}^N V[i], \quad \begin{cases} V[i] = 1, & \text{sensor } S_i \text{ is active and visited} \\ V[i] = 0, & \text{sensor } S_i \text{ is inactive or unvisited} \end{cases}$$

The genetic operators are similar to those used in the conventional solution to the *Traveling Salesman Problem* with certain modifications made on the selection, crossover, mutation, inversion operators to adapt the algorithm to the routing problem under study.

The search result visualized in [Figure 11.6](#) shows that the GA has successfully discovered a clean route that goes through every node within the

sensing zone of the target while avoiding the nodes in sleep state. The global random optimization strategy makes GA capable of exploring any potential sensing zones and visiting nodes in the target neighborhoods with the highest priority for an optimal detection performance in addition to preserving an energy-saving route. This salient feature of the GA-based routing method is evident in the search results for all experimental sensor networks we tested.

In order to make a straightforward performance comparison between GA, LCF, and GCF, we plot the simulation results of node sizes vs. objective values in Figure 11.7, which shows that the GA has a superior overall performance

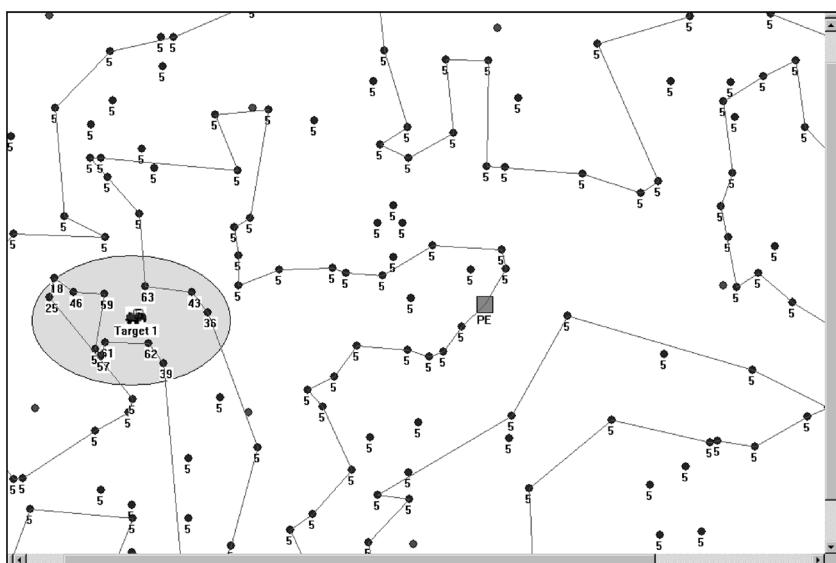


Figure 11.6 Visualization of the search results computed by GA for an MADS with 200 nodes.

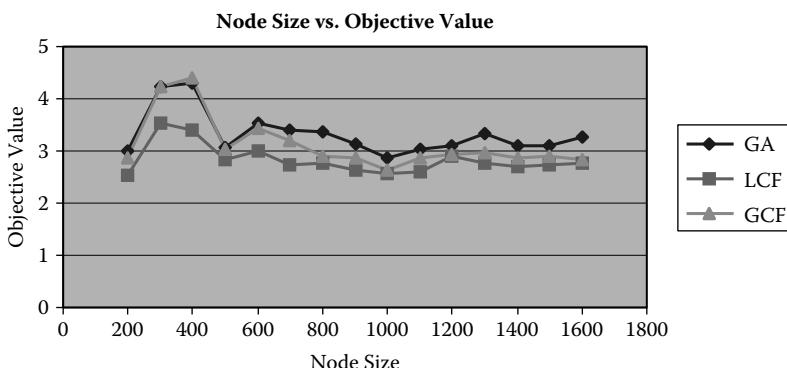


Figure 11.7 Node sizes vs. objective values for GA, LCF, and GCF.

over the other two algorithms in terms of the objective function defined in the implementation.

11.4.2.3 Data routing in multi-sink sensor networks

In traditional single-sink sensor networks where the routing algorithms mainly focus on the nearest path or minimum hops, sensors near the sink or on the critical paths consume energy much faster than others, causing unbalanced energy consumption. The failure of a single-sink node due to energy depletion may cause the breakdown of the entire sensor network. Through the analysis of the disadvantages of single-sink sensor networks, we propose the system architecture of multi-sink sensor networks and a new routing algorithm, priority-based routing (PBR), to balance the energy consumption of the sensor nodes in multi-sink sensor networks. Experiment results show PBR has better performances than traditional methods and prolongs the lifetime of sensor networks.

11.4.2.3.1 Minimum energy cost routing algorithm. Suppose that the sensor node v_i is the data source and there are n sink nodes deployed in the sensor network. The total energy consumption from v_i to sink v_k is $cost(v_i, v_k)$. The minimum communication cost routing algorithm is described as

$$\min_k (cost(v_i, v_k))$$

11.4.2.3.2 Energy level-based routing algorithm (ELBR). ELBR is a routing algorithm based on the energy levels of sensor nodes. We first calculate the energy level of the path and then choose the path with the maximum energy level to transmit data. Here, energy level is defined as the number of times a sensor node can transmit data to its neighbors under the current remaining energy.

$$L_{i,m} = E_{residual} / (k * d_{(i,j)}^a + \tau)$$

where $d_{(i,j)}$ is the distance between two neighboring sensor nodes v_i and v_j , $L_{i,m}$ is the energy level between v_i and v_m , m represents the sink node, $E_{residual}$ is the residual energy of a sensor node, k is power dissipation parameter of transmission circuit, τ is the total energy consumption for sampling, computation and receiving of sensor nodes, and a is a power dissipation exponent, whose value varies according to the environment.

According to the energy levels of sensor nodes, we calculate the energy level of the path and choose the path with the maximum energy level to transmit data, i.e., $\max_n(S_{i,k}) = \max_n(\min_j(L_{j,k}))$, where S is the energy level of a routing path.

11.4.2.3.3 Priority-based routing algorithm (PBR). PBR takes both the energy level and the energy cost of a routing path into consideration, so the energy consumption is more balanced, resulting in a prolonged network lifetime. Assume that node i is the data source, and there are k sink nodes in the sensor network. The energy consumption of data transmission from i to sink j is $\text{cost}(v_i, v_j)$; the energy level of the path is $s_{(i,j)}$. The maximum energy consumption from node i to all the sink nodes is $\max_n(\text{cost}(v_i, v_k))$. After generalization, we have $\text{cost}(v_i, v_j) / \max_n(\text{cost}(v_i, v_k))$. The PBR can be presented as $\max_n(p_{i,k})$, $p_{i,j} = (\max_n(\text{cost}(v_i, v_k)) / \text{cost}(v_i, v_j))^\alpha \times (s_{i,j})^\beta$, where $j \in v_{\text{sink}}$, and α and β are influential exponents. When $\alpha = 1, \beta = 0$, $p_{i,j} = \max_k(\text{cost}(v_i, v_k)) / \text{cost}(v_i, v_j)$, and the value of $\max_k(\text{cost}(v_i, v_k))$ is invariant in the same sensor network, PBR is essentially the minimum energy cost routing algorithm. When $\alpha = 0, \beta = 1$, $p_{i,j} = s_{i,j}$, PBR is essentially the energy level-based routing algorithm.

11.4.3 Network mapping for optimal computing pipeline configuration

A number of large-scale computation-intensive sensor network applications require efficient executions of computing tasks that consist of a sequence of linearly arranged modules, also referred to as subtasks or stages in certain contexts. These modules form a so-called computing pipeline between the data source collected in sensor networks and an end user, as shown in Figure 11.8.

Due to the disparate nature of data sources and the intrinsic heterogeneity of network nodes, communication links, and application computing tasks, deploying component modules on different sets of computing nodes can result in substantial performance variations. The pipeline mapping problems considered in our work are to find an optimal mapping scheme that maps the computing modules onto a set of strategically selected nodes to (1) minimize end-to-end delay for interactive applications where a single dataset is processed sequentially along a computing pipeline and (2) maximize frame rate for streaming applications where multiple datasets are fed into a computing pipeline in a batch-processing mode to sustain continuous data flow.

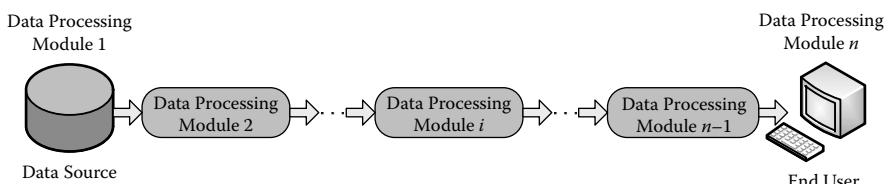


Figure 11.8 A general application computing pipeline consisting of n modules.

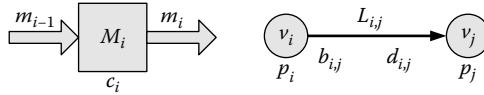


Figure 11.9 Cost models for computing modules, network nodes, and communication links.

11.4.3.1 Cost models of pipeline and network components

We construct an analytical cost model for each pipeline and network component to facilitate the mathematical formulations of the mapping problems. As shown in Figure 11.9, the computational complexity of a computing module M_i is denoted as c_i , which, together with the incoming data size m_{i-1} , determines the number of CPU cycles needed to complete the subtask defined in the module. The output data of size m_i is sent to its immediate successor node in the pipeline for further processing. We use a normalized quantity p_i to represent the overall computing power of a network node v_i . The communication link between network nodes v_i and v_j is denoted as $L_{i,j}$, which is characterized by two attributes, namely bandwidth $b_{i,j}$ and minimum link delay $d_{i,j}$. We estimate the computing time of module M_i running on network node v_j to be $T_{\text{computing}}(M_i, v_j) = \frac{m_{i-1}c_i}{p_j}$ and the transfer time of message size m over a communication link $L_{i,j}$ to be $T_{\text{transport}}(m, L_{i,j}) = \frac{m}{b_{i,j}} + d_{i,j}$.

11.4.3.2 Mapping problem formulation

We consider an underlying transport network that consists of k geographically distributed computing nodes denoted by $v_1, v_2, \dots, v_{k-1}, v_k$. Node $v_i, i = 1, 2, \dots, k - 1, k$ has a normalized computing power p_i and is connected to its neighbor node $v_j, j = 1, 2, \dots, k - 1, k, j \neq i$ with a network link $L_{i,j}$ of bandwidth $b_{i,j}$ and minimum link delay $d_{i,j}$. The transport network is represented by a graph $G = (V, E), |V| = k$, where V denotes the set of network nodes (vertices) and E denotes the set of communication links (edges). Note that the transport network may or may not be a complete graph, depending on whether the node deployment environment is the Internet or a dedicated network. As shown in Figure 11.10, the general computing pipeline consists of n sequential modules, $M_1, M_2, \dots, M_{u-1}, M_u, \dots, M_{v-1}, \dots, M_w, \dots, M_{x-1}, M_x, \dots, M_n$ where M_1 is a data source and M_n is an end user. Module $M_j, j = 2, \dots, n$ performs a computing module of complexity c_j on data of size m_{j-1} received from its predecessor module M_{j-1} and generates and sends data of size m_j to its successor module M_{j+1} .

The general mapping scheme is to decompose the pipeline into q groups of modules, denoted by $g_1, g_2, \dots, g_{q-1}, g_q$, and map them onto a selected path

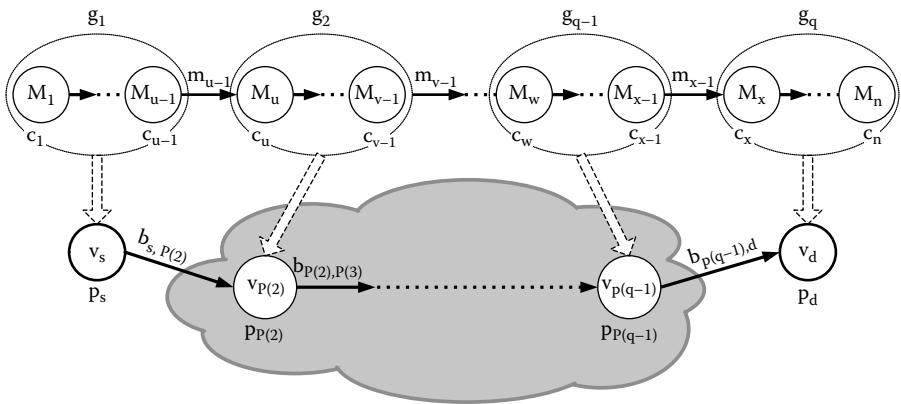


Figure 11.10 Mathematical formulation of the general computing pipeline mapping problems.

P of q nodes from a source node v_s to a destination node v_d in the transport network, where $q \in [1, \min(k, n)]$ and path P consists of a sequence of not necessarily distinct nodes $v_{P[1]} = v_s, v_{P[2]}, \dots, v_{P[q-1]}, v_{P[q]} = v_d$. Note that the path reduces to a single computer when $q = 1$. We define the objective functions for two different types of applications as follows:

1. For interactive applications, we achieve the fastest system response by minimizing the total computing and transport delay of the pipeline from the source node to the destination node:

$$T_{\text{total}}(\text{Path } P \text{ of } q \text{ nodes}) = T_{\text{computing}} + T_{\text{transport}} \quad (11.7)$$

$$\begin{aligned} &= \sum_{i=1}^q T_{g_i} + \sum_{i=1}^{q-1} T_{L_{P[i], P[i+1]}} \\ &= \sum_{i=1}^q \left(\frac{1}{p_{P[i]}} \sum_{j \in g_i, j \geq 2} (c_j m_{j-1}) \right) + \sum_{i=1}^{q-1} \left(\frac{m(g_i)}{b_{P[i], P[i+1]}} \right) \end{aligned}$$

2. For streaming applications, we identify and minimize the time incurred on a bottleneck link or node to maximize the frame rate to produce the smoothest data flow, which is defined as:

$$\begin{aligned}
T_{bottleneck}(\text{Path } P \text{ of } q \text{ nodes}) &= \max_{\substack{\text{Path } P \text{ of } q \text{ nodes} \\ i=1,2,\dots,q-1}} \left\{ \begin{array}{l} T_{computing}(g_i), \\ T_{transport}(L_{P[i],P[i+1]}), \\ T_{computing}(g_q) \end{array} \right\} \\
&= \max_{\substack{\text{Path } P \text{ of } q \text{ nodes} \\ i=1,2,\dots,q-1}} \left\{ \begin{array}{l} \frac{1}{p_{P[i]}} \sum_{j \in g_i \text{ and } j \geq 2} (c_j m_{j-1}), \\ \frac{m(g_i)}{b_{P[i],P[i+1]}}, \\ \frac{1}{p_{P[q]}} \sum_{j \in g_q \text{ and } j \geq 2} (c_j m_{j-1}) \end{array} \right\}
\end{aligned} \quad (11.8)$$

To optimize the network performance of computing pipelines in distributed network environments, we propose a polynomial-time mapping approach, optimal linear pipeline configuration (OLPC), based on dynamic programming, to strategically group the modules and map them onto various network nodes for minimum end-to-end delay or maximum frame rate.

11.4.3.3 Optimal linear pipeline configuration (OLPC)

We present two OLPC algorithms based on dynamic programming that solve the mapping problems for minimal end-to-end delay and maximum frame rate, respectively.

Let $T^j(v_i)$ denote the minimal total delay, with the first j modules mapped to a path from the source node v_s to node v_i under consideration in the computer network. Then, we have the following recursion leading to $T^n(v_d)$:

$$T^j(v_i) = \min_{\substack{j=2 \text{ to } n, v_i \in V \\ u \in adj(v_i)}} \left\{ \begin{array}{l} T^{j-1}(v_i) + c_j m_{j-1} p_{v_i}, \\ \min_{u \in adj(v_i)} (T^{j-1}(u) + c_j m_{j-1} p_{v_i} + m_{j-1} b_{u,v_i}) \end{array} \right. \quad (11.9)$$

with the base conditions computed as:

$$T^2(v_i) = \begin{cases} c_2 m_1 p_{v_i} + m_1 b_{v_s, v_i}, & \forall e_{v_s, v_i} \in E \\ \infty, & \text{otherwise} \end{cases} \quad (11.10)$$

on the first column and

$$T^t(v_s) = \sum_{i=1}^t (c_i m_{i-1} p_{v_s}), t = 1, 2, \dots, n \quad (11.11)$$

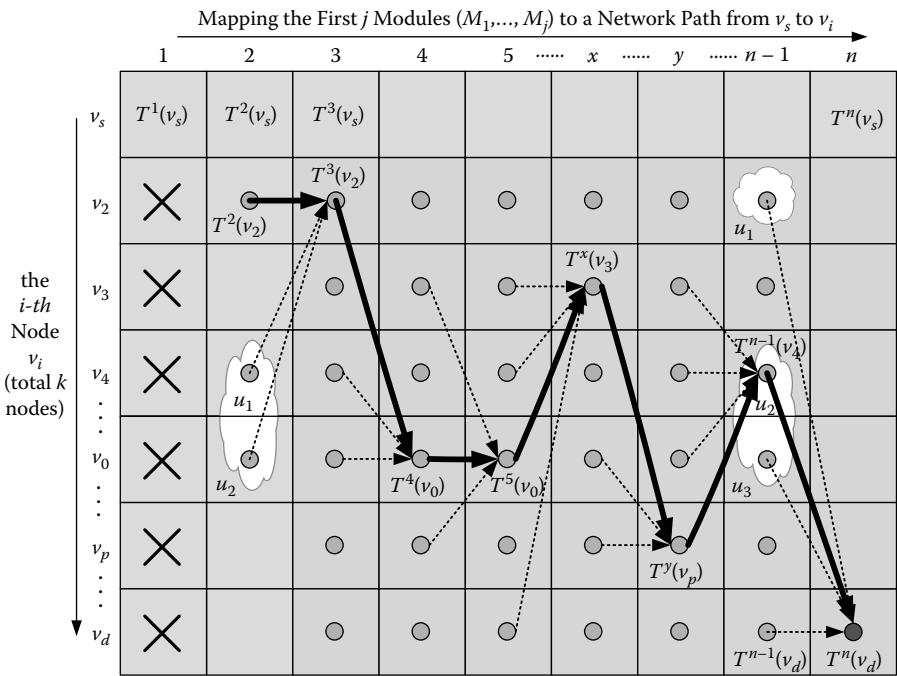


Figure 11.11 Construction of 2-D matrix in dynamic programming for minimum end-to-end delay.

on the first row in the two-dimensional table as shown in Figure 11.11. Every cell $T^j(v_i)$ in the table shown in Figure 11.11 represents an optimal mapping solution that maps the first j modules in the pipeline to a path between the source node v_s and node v_i in the network and is calculated from the intermediate mapping results stored in the left column, i.e., $T^j(v_i)$ is calculated from T^{j-1} . The complexity of this algorithm is $O(n \times |E|)$, where n denotes the number of modules in the linear computing pipeline, and $|E|$ is the number of edges in the distributed network.

The maximum frame rate that a computing pipeline can achieve is limited by the bottleneck unit, i.e., the slowest transport link or computing node along the entire pipeline [46]. Let $1/T^j(v_i)$ denote the maximal frame rate, with the first j modules mapped to a path from source node v_s to node v_i in an arbitrary computer network. We have following recursion leading to $T^n(v_d)$ when nodes are not reused in the mapping:

$$T^j(v_i) = \min_{u \in \text{adj}(v_i)} \left(\max \left(T^{j-1}(u), c_j m_{j-1} p_{v_i}, m_j b_{u, v_i} \right) \right) \quad (11.12)$$

with the base conditions computed as:

$$T^2(v_i)_{v_i \in V, \text{and } v_i \neq v_s} = \begin{cases} \max(c_2 m_1 p_{v_i}, m_1 b_{v_s, v_i}), & \forall e_{v_s, v_i} \in E \\ \infty, & \text{otherwise} \end{cases} \quad (11.13)$$

on the first column in the table.

The computing steps of the maximum frame rate are similar to those of the minimum total delay. The main difference is that we do not cumulate the minimum delay of each mapping subproblem. Instead, we identify the bottleneck unit and calculate its maximum delay in each mapping subproblem.

11.4.4 Sensor data fusion

We propose a binary decision fusion rule that reaches a global decision in the presence of a target by integrating local decisions made by multiple sensors. Without requiring *a priori* probability of target presence, the fusion threshold bounds derived using Chebyshev's inequality ensure a higher hit rate and lower false alarm rate compared to the weighted averages of individual sensors. The Monte Carlo-based simulation results show that the proposed approach significantly improves target detection performance.

11.4.4.1 Problem formulation

We consider N sensor nodes deployed in a three-dimensional region of interest (ROI), centering around a target within radius R . At sensor i , the noise n_i in a sensor measurement is independently and identically distributed (iid) according to the normal distribution.

$$n_i \sim \mathcal{N}(0, 1) \quad (11.14)$$

The sensor measurements are subjective to an additive term due to such noise. Each sensor i makes a binary local decision as:

$$\begin{aligned} H_1 : r_i &= w_i + n_i \\ H_0 : r_i &= n_i \end{aligned} \quad (11.15)$$

where r_i is the actual sensor reading, w_i is the ideal sensor measurement, and n_i is the noise term. We consider an isotropic signal attenuation power model defined by:

$$w_i = \frac{w_0}{\sqrt{1 + \beta d_i^n}} \quad (11.16)$$

where w_0 is the original signal power emitted from the target located at point (x_0, y_0, z_0) , β is a system constant, and d_i represents the Cartesian distance between the target and the sensor node, which is defined in Equation 11.17.

Parameter n is the signal attenuation exponent, typically ranging from 2 to 3, and the distance is defined as:

$$d_i = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2 + (z_i - z_0)^2} \quad (11.17)$$

This model describes a three-dimensional unobstructed region monitored by a set of sensors that detect the signal emitted from a target within the monitoring area.

Suppose that every sensor node employs the same threshold τ for decision making regardless of its distance to the target. The expected signal strength w_i for sensor i can be computed from Equation 11.16 according to its distance to the target. Thus, the hit rate p_{h_i} and false alarm rate p_{f_i} for sensor i can be derived as follows:

$$p_{h_i} = \int_{\tau}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-w_i)^2}{2}} dx \quad (11.18)$$

$$p_{f_i} = \int_{\tau}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \quad (11.19)$$

A simplified homogeneous system ignores the impact of various distances to the target on sensor detection capabilities so that every sensor has the same hit rate and false alarm rate.

11.4.4.2 Threshold-OR fusion method

Each sensor i makes an independent binary decision S_i as either 0 or 1. The fusion center collects local decisions and computes S as:

$$S = \sum_{i=1}^N S_i \quad (11.20)$$

which is then compared with a threshold T to make a final decision. Under the assumption that sensor measurements are statistically independent under H_1 , the mean and variance of S are given as follows when a target is present:

$$E(S | H_1) = \sum_{i=1}^N p_{h_i}, \quad (11.21)$$

$$Var(S | H_1) = \sum_{i=1}^N p_{h_i}(1 - p_{h_i})$$

Similarly, under the assumption that sensor measurements are statistically independent under H_0 , the mean and variance of S when a target is absent can be defined.

The threshold value T is critical to the system performance. Let P_h and P_f denote the hit rate and false alarm rate of the fused system. We give reasonable value bounds for T as

$$\sum_{i=1}^N p_{f_i} < T < \sum_{i=1}^N p_{h_i}.$$

The weighted averages of p_{h_i} and p_{f_i} , $i = 1, 2, \dots, N$ are defined as follows:

$$\sum_{i=1}^N \frac{p_{h_i}}{\sum_{j=1}^N p_{h_j}} p_{h_i} = \frac{\sum_{i=1}^N p_{h_i}^2}{\sum_{i=1}^N p_{h_i}} \quad (11.22)$$

$$\sum_{i=1}^N \frac{1-p_{f_i}}{\sum_{j=1}^N (1-p_{f_j})} p_{f_i} = \frac{\sum_{i=1}^N (1-p_{f_i})p_{f_i}}{\sum_{i=1}^N (1-p_{f_i})} \quad (11.23)$$

We desire better detection performance of the fused system than the corresponding weighted averages in terms of higher hit rate and lower false alarm rate such that:

$$P_h > \frac{\sum_{i=1}^N p_{h_i}^2}{\sum_{i=1}^N p_{h_i}} \quad (11.24)$$

$$P_f < \frac{\sum_{i=1}^N (1-p_{f_i})p_{f_i}}{\sum_{i=1}^N (1-p_{f_i})} \quad (11.25)$$

We first consider a lower bound of the hit rate of the fused system:

$$P_h = P\{S \geq T | H_1\} \geq P\left\{\left|S - \sum_{i=1}^N p_{h_i}\right| \leq \left(\sum_{i=1}^N p_{h_i} - T\right) | H_1\right\} \geq 1 - \frac{\sigma^2}{k^2} \quad (11.26)$$

$$= 1 - \frac{\sum_{i=1}^N p_{h_i} (1 - p_{h_i})}{\left(\sum_{i=1}^N p_{h_i} - T\right)^2}$$

where we applied Chebyshev's inequality in the third step as shown in Figure 11.12 and

$$k = \left(\sum_{i=1}^N p_{h_i} - T\right).$$

Now the condition in Equation 11.24 can be ensured by the following sufficient condition.

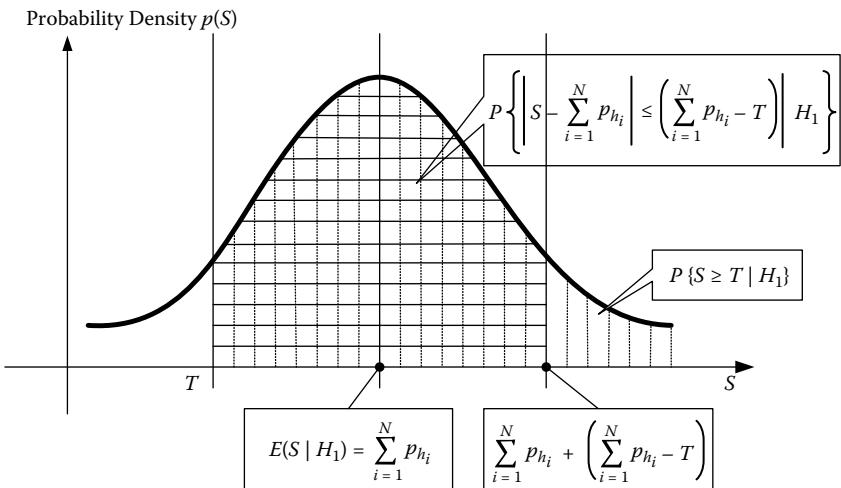


Figure 11.12 Application of Chebyshev's inequality in calculating the lower bound of the system hit rate.

$$1 - \frac{\sum_{i=1}^N p_{h_i} (1-p_{h_i})}{(\sum_{i=1}^N p_{h_i} - T)^2} \geq \frac{\sum_{i=1}^N p_{h_i}^2}{\sum_{i=1}^N p_{h_i}} \quad (11.27)$$

Following that, an upper bound of T can be derived from Equation 11.27 as follows:

$$T \leq \sum_{i=1}^N p_{h_i} - \sqrt{\sum_{i=1}^N p_{h_i}} \quad (11.28)$$

Similarly, for the false alarm rate, we carry out a similar procedure to compute the lower bound from Equation 11.29 to Equation 11.33. Again, Chebyshev inequality is applied in the second step in Equation 11.31.

$$P_f = P\{S \geq T | H_0\} = 1 - P\{S < T | H_0\} \quad (11.29)$$

$$P\{S < T | H_0\} \geq P\{|S - \sum_{i=1}^N p_{f_i}| \leq (T - \sum_{i=1}^N p_{f_i}) | H_0\} \quad (11.30)$$

$$P_f \leq 1 - P\{|S - \sum_{i=1}^N p_{f_i}| \leq (T - \sum_{i=1}^N p_{f_i}) | H_0\} \leq \frac{\sum_{i=1}^N p_{f_i} (1-p_{f_i})}{(T - \sum_{i=1}^N p_{f_i})^2} \quad (11.31)$$

Now we consider the condition that ensures the false alarm probability of fuser is smaller than that of weighted average given by:

$$\frac{\sum_{i=1}^N p_{f_i} (1-p_{f_i})}{(T - \sum_{i=1}^N p_{f_i})^2} \leq \frac{\sum_{i=1}^N (1-p_{f_i}) p_{f_i}}{\sum_{i=1}^N (1-p_{f_i})} \quad (11.32)$$

$$T \geq \sum_{i=1}^N p_{f_i} + \sqrt{\sum_{i=1}^N (1-p_{f_i})} \quad (11.33)$$

Therefore, we define the range of T using the upper bound in Equation 11.28 and lower bound in Equation 11.33 as follows:

$$\left[\sum_{i=1}^N p_{f_i} + \sqrt{\sum_{i=1}^N (1-p_{f_i})}, \sum_{i=1}^N p_{h_i} - \sqrt{\sum_{i=1}^N p_{h_i}} \right] \quad (11.34)$$

To ensure that the upper bound is larger than the lower bound, we have the following restrictions on individual hit rates, individual false alarm rates, and the number of sensor nodes, for the heterogeneous and homogeneous systems in Equation 11.35 and Equation 11.36, respectively:

$$\sum_{i=1}^N p_{f_i} + \sqrt{\sum_{i=1}^N (1-p_{f_i})} - \sum_{i=1}^N p_{h_i} + \sqrt{\sum_{i=1}^N p_{h_i}} \leq 0 \quad (11.35)$$

$$p_h - p_f \geq \frac{\sqrt{p_h} + \sqrt{1-p_f}}{\sqrt{N}} \quad (11.36)$$

11.4.4.3 Simulation results

Our simulation examples are based on 100,000 runs for better probability estimation with large sampling size. The receiver operative characteristic (ROC) curve, a plot of the hit rate against the false alarm rate for the different possible thresholds, can be obtained by our simulation. There is tradeoff between sensitivity and specificity, namely any increase in sensitivity as hit rate will be accompanied by an increase in nonspecificity as false alarm rate. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the system is. The closer the curve comes to the 45-degree diagonal of the ROC space, the less accurate the test.

In a homogenous system, all sensor nodes assume the same hit rate and false alarm rate. In the first simulation test, we set sensor hit rate to be 0.65 and sensor false alarm rate to be 0.2. In Figure 11.13, four system ROC curves with sensor node numbers going from 15 to 25 are plotted. The desirable segment on the ROC curve is located by restricting thresholds selection. Due to the discrete nature of our binary decision system, we identify the selected ROC segment as individual enlarged markers. From Figure 11.13, we observe that all selected points locate in the top left corner of ROC space and bear hit rate and false alarm rate that are greatly superior to that of a single sensor node. As the number of sensor nodes increases, our system performance is improved due to ample resources.

In a heterogeneous system, from Equation 11.18 and Equation 11.19, we know that sensors have the same false alarm rate and different hit rate due to various distance to the target. We assume the known hit rate and false alarm rate for sensor with zero distance to the target to be p_h and p_f , respectively. Threshold τ can be calculated from Equation 11.19, since false alarm rate is known. Then, original signal power S_0 can be computed from Equa-

tion 11.18. S_i can be derived from Equation 11.16, and consequently, p_{hi} can be computed according to Equation 11.18. This heterogeneous system serves as a good approximation model for real scenario. Our approach can also handle systems with heterogeneous false alarm rates if needed. The original hit rate is set to be 0.75, and the false alarm rate is set to be 0.2. Deployment radius goes from 1 to 3 for result comparison. Table 11.1 gives the simulation results under different radius. The increased radius lowers the weighted average hit rate and negatively affects the system performance in terms of system hit rate, system false alarm rate, and system thresholds. It further demonstrates the fact that sensors should be deployed as close to potential targets as possible. From Figure 11.14, our fusion system achieves a much higher hit rate

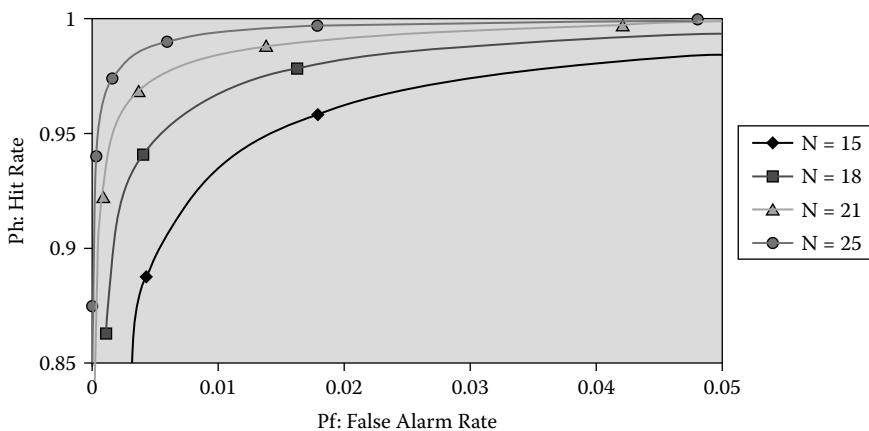


Figure 11.13 Homogeneous ROC curve with different sensor node number.

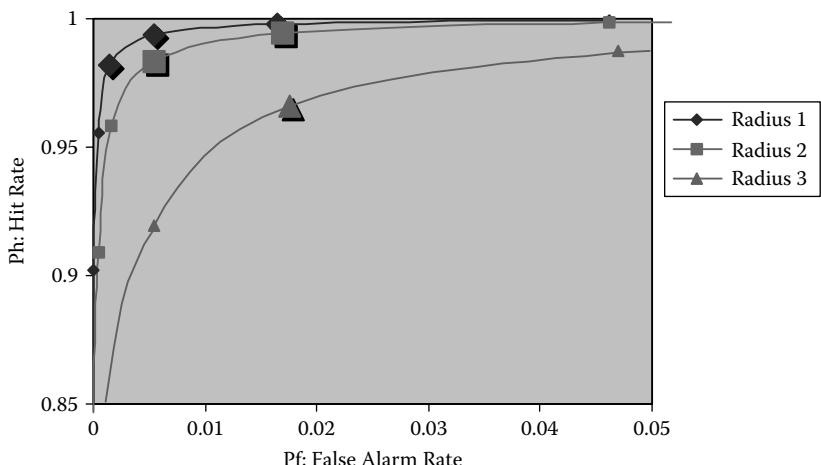
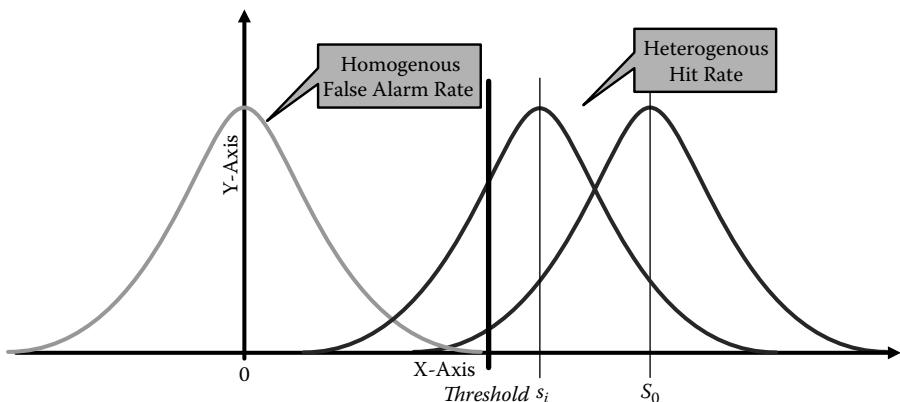


Figure 11.14 Heterogeneous ROC curve with different deployment radius.

Table 11.1 Numeric Results with Different Deployment Radius

$N = 25, p_j = 0.2, p_h = 0.75$	T_{lower}	T_{upper}	W_{p_h}
Radius = 1	10	12	0.67
Radius = 2	10	11	0.64
Radius = 3	10	10	0.58

**Figure 11.15** Normal distribution based hit rate and false alarm rate calculation.

close to 0.95 and lower false alarm rate below 0.02 as comparison to weighted hit rate and false alarm rate of 0.67 and 0.2, respectively. Figure 11.15 demonstrates the usage of normal distribution function in computing individual sensor hit rate and false alarm rate.

11.5 Conclusion

In recent years, the rapid advances in sensing and computing technologies have generated a great deal of interest in the development of new computational structures and strategies for large-scale computation-intensive applications based on distributed sensor networks. The success of such large-scale applications requires the integration of solutions to the problems of sensor deployment, data routing, distributed data processing, and information fusion in both sensor and computer networks. We proposed a general framework IDSS-SC for the design of an intelligent decision support system that integrates sensing and computing power in support of various sensor network applications.

The spatial structure of IDSS-SC is composed of three major parts: wireless sensor network for data collection, wired computer network for data processing, and command control center for intelligent decision making. For each subsystem of IDSS-SC, we analytically formulated the problem, presented

a technical solution, and evaluated the solution with either experimental or simulation data. These problems and their corresponding solutions demonstrate the wide scope of the present research efforts in this area. The effectiveness of a multisensor system depends on the individual solutions to various problems in sensor model construction, sensor deployment scheme, sensor network architecture, information translation cost, and network fault tolerance. So far, very little basic research has been done on a general framework needed to provide a systematic approach to the design of distributed sensor and computer network systems.

IDSS-SC aims at integrating the physical world and the cyberspace into a global self-organizing information space through the emerging sensor and computer science technologies. As the Internet has changed the way people communicate in the virtual world, IDSS-SC extends this vision to the physical world, thus enabling novel ways for humans to interact with environments and facilitating interactions among entities of the physical world and finally making intelligent decisions.

Acknowledgments

This research is sponsored by National Science Foundation under Grant No. CNS-0721980 and Oak Ridge National Laboratory, U.S. Department of Energy, under Contract No. PO 4000056349 with University of Memphis.

References

1. Bashir, A. F. and V. Susarla and K. Vairavan. 1975. A statistical study of the performance of a task scheduling algorithm. *IEEE Transactions on Computer*, 32(12):774–777.
2. Agarwalla, B., N. Ahmed, D. Hilley, and U. Ramachandran. 2006. Streamline: a scheduling heuristic for streaming application on the grid. In *Thirteenth Multi-media Computing and Networking Conference*, San Jose, CA.
3. Bulusu, N. and D. Estrin. 2002. Scalable ad hoc deployable RF-based localization. In *Proceedings of the Grace Hopper Celebration of Women in Computing Conference 2002*, Vancouver, British Columbia, Canada.
4. Buyya, R. 2002. *Economic-based Distributed Resource Management and Scheduling for Grid Computing*. Ph.D. thesis, Monash University, Melbourne, Australia.
5. Buyya, R., D. Abramson, and J. Giddy. 2000. Nimrod/G: an architecture for a resource management and scheduling system in a global computational grid. In *High Performance Computing, ASIA*. 4th International Conference on High Performance Computing in Asia-Pacific Region, Beijing, China, IEEE Computer Society Press.
6. Cao, J., S. A. Jarvis, S. Saini, and G. R. Nudd. 2003. Gridflow: workflow management for grid computing. In *The 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 198–205.
7. Chan, H. and A. Perrig. 2003. Security and privacy in sensor networks. *Computer*, 36(10):103–105.

8. Chan, H., A. Perrig, and D. Song. 2003. Random key predistribution schemes for sensor networks. In *Proc. of the IEEE Symposium on Security and Privacy*, pp. 197–213.
9. Chaudhary, V. and J. K. Aggarwal. 1993. A generalized scheme for mapping parallel algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 4(3):328–346.
10. Chen, L. and G. Agrawal. 2004. Resource allocation in a middleware for streaming data. In *The 2nd Workshop on Middleware for Grid Computing*, Toronto, Canada.
11. Chen, L. and G. Agrawal. 2006. Supporting self-adaptation in streaming data mining applications. In *IEEE International Parallel and Distributed Processing Symposium*.
12. Deng, J., R. Han, and S. Mishra. 2003. Security support for in-network processing in wireless sensor networks. In *Proc. of the First ACM Workshop on the Security of Ad Hoc and Sensor Networks*.
13. Douceur, J. R. 2002. The sybil attack. In *Proc. of IPTPS '02*, Cambridge, MA.
14. Du, W. et al. 2004. A key management scheme for wireless sensor networks using deployment knowledge. In *INFOCOM 2004. Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies*.
15. Du, W., J. Deng, Y. Han, and P. Varshney. 2003. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proc. of the 10th ACM Conference on Computer and Communications Security (CCS'03)*, pp. 42–51.
16. Eschenauer, L. and V. D. Gligor. 2002. A key management scheme for distributed sensor networks. In *Proc. of the 9th ACM Conference on Computer and Communication Security*, pp. 41–47.
17. Ganesan, D., R. Cristescu, and B. Beferull-Lozano. 2006. Power-efficient sensor placement and transmission structure for data gathering under distortion constraints. *ACM Transactions on Sensor Networks* 2(2):155–181.
18. Gerasoulis, A. and T. Yang. 1992. A comparison of clustering heuristics for scheduling DAG's on multiprocessors. *Journal of Parallel and Distributed Computing*, 16(4):276–291.
19. Guibas, L., D. Lin, J. C. Latombe, S. LaValle, and R. Motwani. 2000. Visibility-based pursuit evasion in a polygonal environment. *International Journal of Computational Geometry Applications*, 9(5): 471–494.
20. Heinzelman, W., A. Chandrakasan, and H. Balakrishnan. 2000. Energy-efficient communication protocols for wireless microsensor networks. In *Proc. of 33rd Hawaiian Int'l Conf. on System Science*.
21. Intanagonwiwat, C., R. Govindan, and D. Estrin. 2000. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCOM '00)*, August 2000, Boston, MA.
22. Karlof, C., N. Sastry, and D. Wagner. 2004. Tinysec: a link layer security architecture for wireless sensor networks. In *Proc. of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*.
23. Kim, S. J. and J. C. Browne. 1988. A general approach to mapping of parallel computation upon multiprocessors architectures. In *Proceedings of International Conference on Parallel Processing*, pp. 1–8.
24. Kwok, Y.-K. and I. Ahmad. 1996. Dynamic critical-path scheduling: an effective technique for allocating task graph to multiprocessors. *IEEE Transactions on Parallel and Distributed Systems*, 7(5):506–521.

25. Liu, D. and P. Ning. 2003. Establishing pairwise keys in distributed sensor networks. In *Proc. of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pp. 52–61.
26. Martinez, S. and F. Bullo. 2006. Optimal sensor placement and motion coordination for target tracking. *Automatica*, 42(4):661–668.
27. Meguerdichian, S., F. Koushanfar, M. Potkonjak, and M. Srivastava. 2001. Coverage problems in wireless ad hoc sensor networks. In *Proc. of IEEE INFOCOM 2001*, Anchorage, AK.
28. Perrig, A. et al. 2001. Spins: security protocols for sensor networks. In *Proceedings of Seventh Annual International Conference on Mobile Computing and Networks MOBICOM 2001*.
29. Qi, H., S. S. Iyengar, and K. Chakrabarty. 2001. Multi-resolution data integration using mobile agents in distributed sensor networks. In *IEEE Trans. Systems, Man, and Cybernetics Part C: Applications and Rev.* 31:383–391.
30. Reibman, A. R. and L. W. Nolte. 1987. Design and performance comparison of distributed detection networks. *IEEE Trans. Aerosp. Electron. Syst.*, AES (23):789–797.
31. Reibman, A. R. and L. W. Nolte. 1987. Optimal detection and performance of distributed sensor systems. *IEEE Trans. Aerosp. Electron. Syst.*, AES (23):24–30.
32. Royer, E. M. and C.-K. Toth. 1999. A review of current routing protocols for ad hoc mobile wireless networks. *IEEE Personal Communication*, April 1999.
33. Sadjadi, F. A. 1986. Hypothesis testing in a distributed environment. *IEEE Trans. Aerosp. Electron. Syst.*, AES (22):134–137.
34. Shirazi, B., M. Wang, and G. Pathak. 1990. Analysis and evaluation of heuristic methods for static scheduling. *Journal of Parallel and Distributed Computing*, (10):222–232.
35. Tenney, R. R. and N. R. Sandell. 1981. Detection with distributed sensors. *IEEE Trans. Aerosp. Electron. Syst.*, AES (17):501–510.
36. Thomopoulos, F. A., R. Viswanathan, and D. C. Bougooulias. 1987. Optimal decision fusion in multiple sensor systems. *IEEE Trans. Aerosp. Electron. Syst.*, AES (23):644–653.
37. Tsitsiklis, J. 1993. *Decentralized detection*. In *Advances in Statistical Signal Processing*, vol. 2, eds. H. V. Poor, and J. B. Thomas. JAI Press, Greenwich, CN, pp. 297–344.
38. Varshney, P. 1997. *Distributed Detection and Data Fusion*. Springer-Verlag, New York.
39. Wang, G., G. Cao, and T. L. Porta. 2004. Movement-assisted sensor deployment. In *Proc. IEEE INFOCOM*.
40. Wood, A. D. and J. A. Stankovic. 2002. Denial of service in sensor networks. *Computer*, 35(10):54–62.
41. Wu, Q., S. S. Iyengar, N. S. V. Rao, J. Barhen, V. K. Vaishnavi, H. Qi, and K. Chakrabarty. 2004. On computing mobile agent routes for data fusion in distributed sensor networks. *IEEE Transactions on Knowledge and Data Engineering*, 16(6):740–753.
42. Wu, Q., N. S. V. Rao, X. Du, S. S. Iyengar, and V. K. Vaishnavi. 2007. On efficient deployment of sensors on planar grid. *Computer Communications* 30(14–15):2721–2734.
43. Ye, F., S. Lu, and L. Zhang. 2005. Gradient broadcast: A robust, long lived sensor network. *Wireless Networks*, 11(3):285–298.
44. Ye, F., H. Luo, J. Cheng, S. Lu, and L. Zhang. 2002. A two tier data dissemination model for large-scale wireless sensor networks. In *Proc. of ACM MOBICOM'02*, Atlanta, GA.

45. Ye, F., H. Luo, S. Lu, and L. Zhang. 2004. Statistical en-route detection and filtering of injected false data in sensor networks. In *Proc. of IEEE INFOCOM*, 2004.
46. Zhu, M., Q. Wu, N. S. V. Rao, and S. S. Iyengar. 2007. Optimal pipeline decomposition and adaptive network mapping to support remote visualization. *Journal of Parallel and Distributed Computing* 67(8):947–956.
47. Zhu, S., S. Setia, and S. Jajodia. 2003. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proc. of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, Washington DC.

chapter twelve

Defense applications of SoS

Charles E. Dickerson

Contents

12.1	Background	319
12.2	The move toward capability engineering	320
12.3	Late twentieth century SoS history in U.S. defense systems	321
12.3.1	Description of forces and technologies	321
12.3.2	A revolution in military affairs	322
12.4	A transition to network enablement and system of systems	323
12.4.1	Network enablement of naval forces	323
12.4.2	Network-centric warfare report to Congress	324
12.4.3	From network-centric enablement to system of systems	326
12.4.4	SoSE for Air Force capability development	326
12.5	System of systems or network enablement?	327
12.5.1	A lesson from history	327
12.5.2	Systems engineering considerations	328
12.5.3	Critical roles of information exchange and capabilities integration	330
12.6	Examples of U.S. DoD defense applications of SoS	332
12.6.1	OSD SoS SE Guide	333
12.6.2	Future Combat System (FCS)	334
12.6.3	Single integrated air picture (SIAP)	334
12.6.4	Naval integrated fire control–counter air (NIFC-CA)	335
12.6.5	Commissary advanced resale transaction system (CARTS)	335
12.7	Related work and research opportunities	336
	References	336

12.1 Background

As the defense community has continued to move toward increasingly complex weapon systems that must support joint and coalition operations, the need for system of systems (SoS) engineering becomes more critical to the achievement of military capabilities. The U.S. Department of Defense (DoD)

and the U.K. Ministry of Defence (MoD) continue to face a critical challenge: the integration of multiple capabilities across developing and often disparate legacy systems that must support multiple warfare areas.

Over the past decade, the need for SoS-enabled mission capabilities have led to changes in policy for the acquisition of systems and the assemblage of battle forces. Defense acquisition structures have been reorganizing in order to integrate acquisition activities in a way that leads to the achievement of capabilities through a system-of-systems approach rather than from just the performance of individual systems or platforms.

Earlier efforts to meet the challenge faced by the defense community have included solutions using Network Centric Operations (NCO) in the United States and Network Enabled Capability (NEC) in the United Kingdom. Although progress has been made, future progress can be expected to depend upon the advancement of the practice of systems engineering at the SoS level. System of systems engineering (SoSE) processes and practices must enable the acquisition and implementation of systems of systems that support the integration of multiple capabilities across multiple warfare areas. Recently there has been extensive debate across a broad community of government, industry, and academic stakeholders about systems engineering processes and practices at the SoS level. But whatever form SoSE takes for defense systems, the technical processes of SoSE must support the realization of mission capabilities attainable from a system of systems that cannot be attained from a single system.

12.2 *The move toward capability engineering*

Defense systems are designed and developed under defense acquisition policy. Recent changes in U.S. DoD acquisition policy were motivated by a focus on capabilities-based acquisition. The idea of organizing the defense acquisition strategy around specific military capabilities is traceable to reforms in the British Ministry of Defence (MoD) in the late 1990s. These reforms were a significant departure from the traditional practice of organizing the acquisition strategy around specific threats to be countered by individual systems, platforms, or military components. The concept of acquiring capabilities has been institutionalized in the DoD and across the individual services through the Joint Capability Integration and Development System (JCIDS) process and revisions in the DoD 5000 policies. Capabilities-based planning, to include prototyping and JCIDS, is now the emerging overarching concept being used in DoD acquisition. In the United Kingdom, the MoD has continued to evolve its strategy for the acquisition of defense systems, emphasizing that the “whole of their defence acquisition community, including industry” must be “able to make the necessary shifts in behaviours, organisations, and business processes” [1]. The basic principles of the MoD’s Smart Acquisition

are: "the primacy of through-life considerations; the coherence of defence spread across research, development, procurement, and support; and successful management of acquisition at the Department level."

In line with this policy, the U.K. MoD regards Military Capability as comprising contributions from across the set of "Defence Lines of Development" (DLoD).^{*} One of these is Equipment, the others being "Training, Personnel, Infrastructure, Concepts and Doctrine, Organisation, Information and Logistics." A ninth DLoD, Interoperability, is also defined, and is seen as embracing all the others. An integrated capability thus requires suitable components from all of the DLoDs to be assembled into a coherent whole, to meet a specific mission need.

The United Kingdom has adopted the term NEC to reflect the aspiration to ensure that military capability is network enabled, i.e., that all elements of a force not only communicate effectively with one another, but enjoy all the benefits of shared awareness and understanding, and which lead to better informed decision making. This is now fundamental to U.K. force development and represents a major contribution to the Interoperability DLoD mentioned above.

To do all of this effectively requires a broader approach to system of systems engineering, in which all of the components of the capability, at all stages of their respective lifecycles, are regarded as systems which must be engineered to operate together. Work continues in the United Kingdom to mature this concept and devise practical approaches to its realization; this is beyond the present scope of this chapter, which concentrates on developments within the U.S. DoD in this area.

12.3 Late twentieth century SoS history in U.S. defense systems

In the last decades of the twentieth century, dramatic changes occurred in both the commercial markets of technology and the traditional role of government-sponsored research and development for defense technologies.

12.3.1 Description of forces and technologies

The decade of the 1990s was a time of transition for coalition defense systems, and especially so for U.S. defense systems. In the previous decades, the Cold War had set the concept of operations and determined the types of systems needed. A classical symmetric war against a massive ground force was envisioned. Among the advanced technologies that were intended to give coalition forces a significant advantage were stealth aircraft and precision guided

* The equivalent in the United States is DOTMLPF.

munitions, but information technology (IT) was in its infancy in the defense community. But by the end of the 1990s, the commercial investment in research and development (R&D) for IT vastly exceeded that of the U.S. DoD. R&D for the microchip for example, which had been developed at Texas Instruments in the 1960s by the DoD, was now being driven by the commercial market so strongly that defense acquisition strategies emphasized the need for commercial off-the-shelf technologies (COTS). By the end of the 1900s, the U.S. Navy had seen the development of its first COTS open-architecture combat system, which would be deployed on the Virginia Class submarines shortly after the turn of the century.

12.3.2 A revolution in military affairs

By the 1990s, the power of precision weapons coupled with an effective architecture based on sensor technology and the emergence of IT had been amply demonstrated in the first Persian Gulf War and later in Kosovo. The apparent discontinuous advantage in military capability and effectiveness of this power was evidence of the viability of concepts originating from Soviet military thought in the 1970s and 1980s, which are known as the Revolution in Military Affairs (RMA). This theory of future warfare is intimately related to the concepts of system of systems and network-centric warfare. Toward the end of the 1990s, this theory was becoming part of the Joint Vision 2010 [2] for warfare in what is now the twenty-first century. The interest in RMA as an organizing concept goes beyond the national boundaries of the Cold War nations, such as the United States and the United Kingdom in a symmetric posture against the communist regime of the Soviet Union in the 1970s and 1980s. India, Singapore, and the People's Republic of China are currently counted among the many nations with an interest in RMA. But the limiting factor for a country to enter directly into the RMA is perhaps the infrastructure cost of the architecture.

The RMA point of view can be compared with radical changes in military thought that emerged from new concepts of warfare between the first and second World Wars, such as Blitzkrieg. This was a revolution in offensive military capabilities based on the integration of weapons and communications technology to achieve new capabilities—and the command structure designed to exploit both simultaneously. All three elements were essential to the success of this battlefield tactic. And the Nazi war machine enjoyed a discontinuous advantage in military capability and effectiveness comparable to those enjoyed by coalition forces in the Persian Gulf and Kosovo.

12.4 A transition to network enablement and system of systems

12.4.1 Network enablement of naval forces

At the turn of the decade, in the year 2000, under the auspices of the National Academy of Sciences, the Naval Studies Board released the report, *Network-Centric Naval Forces* [3]. This report was motivated in part by a declaration from the Chief of Naval Operations that the Navy would be shifting its operational concept from one based on platform-centric warfare to one based on the concepts of network-centric warfare. The concept of NCO was introduced as a new approach to war fighting. In an NCO system, a set of assets, balanced in their design and acquisition, were envisioned to be integrated so as to operate together effectively as one complete system to accomplish a mission. The assets assembled in such a system were envisioned to encompass naval force combat, support, and C4ISR (Command, Control, Communications, Computing, Intelligence, Surveillance, and Reconnaissance) elements and subsystems integrated into an operational and combat network. Although the design and development of the assets in an NCO system would require systems engineering, the assets would be part of an integrated network supported by a common command, control, and information infrastructure.

NCO was envisioned by the Naval Studies Board to derive its power from a geographically dispersed naval force embedded within an information network that would link sensors, shooters, and command and control nodes to provide enhanced speed of decision making, and rapid synchronization of the force as a whole to meet its desired objectives. The NCO approach would enable naval forces to perform collaborative planning and to achieve rapid, decentralized execution of joint actions. In this way, NCO forces could focus the maneuvers and fires of widely dispersed forces to carry out assigned missions rapidly and with great economy of force.

Figure 12.1 illustrates these concepts and how NCO can enable the massing of effects without a massing of forces; i.e., in Figure 12.1, the weapons effects can be massed against the threat even though the platforms may be geographically dispersed. Figure 12.1 also illustrates that network enablement is as much about the capabilities achieved through the interoperation of systems as it is about networks. The networks simply enable the interoperation [4]. The conceptual shift from a platform-centric system architecture to a network-centric system architecture is also illustrated. Platform-centric operations usually involve a sectored Battlespace as a means of weapon systems control and threat engagements (as illustrated in the left-hand panel of Figure 12.1). Platforms carry sensors, processors, and weapons (or combinations thereof),

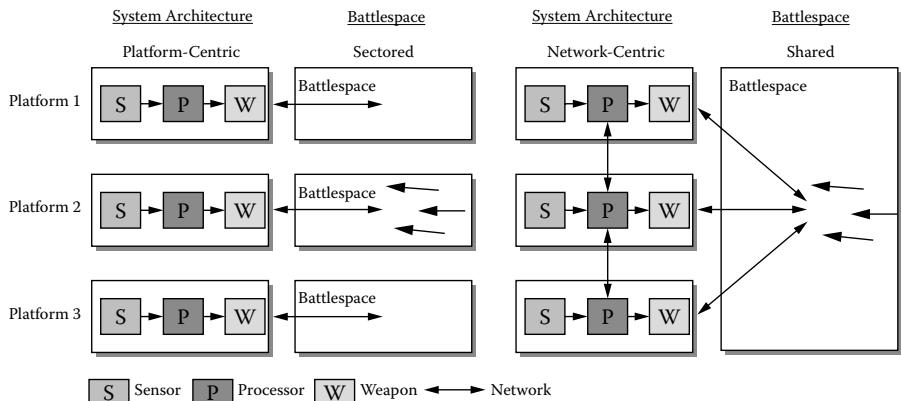


Figure 12.1 System of systems integration through networks. (Adapted from [3].)

the effectiveness of which may be increased by the integration enabled by a network-centric architecture (as illustrated in the right-hand panel of Figure 12.1). The real-time fusing of multiple sensor outputs is another driver for the target engagement architecture illustrated in Figure 12.1. Its importance cannot be overemphasized; it is fundamental to NCO.

12.4.2 Network-centric warfare report to Congress

By the year 2001, all of the Armed Services and many of the agencies were actively seeking to become network centric. The United States Congress called for a report so as to better understand NCW and consider its impact on defense spending. The Office of the Secretary of Defense (OSD) collected and integrated the inputs from the various Services and the agencies into the NCW DoD Report to Congress [5]. The Army, Navy, Air Force, Marine Corps, National Security Agency/Central Security Office, Ballistic Missile Defense Office, National Imagery and Mapping Agency, and Defense Threat Reduction Agency had all adopted visions and implementation plans for NCW.

NCW was now viewed as a maturing approach to warfare that would allow the DoD to achieve *Joint Vision 2020* operational capabilities. Furthermore, NCW sought to achieve an asymmetrical information advantage over threat forces. As illustrated in Figure 12.2, this information advantage is achieved, to a large extent, by allowing force access to a previously unreachable region of the information domain—the network-centric region—that is broadly characterized by both increased information richness and increased information reach.

The source of the transformational combat power enabled by NCW concepts was attributed by the Report to Congress to be the relationships in

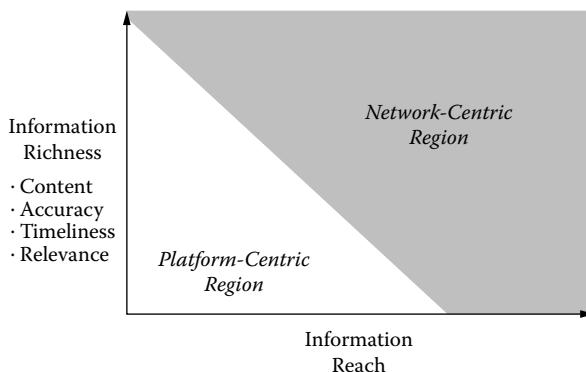


Figure 12.2 Network-centric region of the information domain.

warfare that take place simultaneously in and among the *physical*, the *information*, and the *cognitive* domains. Key elements of the transformation included: Information Superiority, Decision Superiority, Dominant Maneuver, Precision Engagement, Focused Logistics, and Information Operations.

Each Service submitted NCW vision statements, which are briefly summarized below.

In its NCW vision, the Army stated that accomplishing its vision was strongly dependent on the potential of linking together and networking of geographically dispersed combat elements. The theory behind the Army's NCW vision was that by linking sensor networks, Command and Control (C^2) networks, and shooter networks, it could achieve new efficiencies in all military operations from the synergy that would be derived by simultaneously sharing information in a common operating environment. In addition, such linkages would allow for the discovery of new concepts of operations both among Army forces and Joint forces in theater.

The Navy's Network Centric Operations (NCO) articulated its path to NCW. It stated that, "In developing NCW systems, a different approach to applying the principles must be taken. NCW requires that technology, tactics, and systems be developed together." The Navy submission pointed to three military trends: a shift toward Joint effects-based combat, heightened reliance on knowledge superiority, and use of technology by adversaries to rapidly improve capabilities in countering U.S. strengths. It noted that, "The power, survivability and effectiveness of the future force will be significantly enhanced through the networking of war fighters."

The Air Force's NCW vision recognized that dominating the information spectrum is just as critical today as controlling air and space or occupying land was in the past. And the time available for collecting information, processing it into knowledge, and using it to support war fighting initiatives

is shrinking. Processing, exploiting, and manipulating information have always been essential parts of warfare, but information has evolved beyond its traditional role. "Today, information is itself both a weapon and a target." The vision also stated that the key to improving Air Force capabilities would involve not just improvements to individual sensors, networking of sensors, and C² for sensors, but also in new ways of thinking about warfare and the integration of U.S. forces.

While the Marine Corps has not historically used the term network-centric warfare, the Corps noted in its vision that the principles embodied by the term have long been an integral part of Marine Corps operations. The Corps acknowledged that its continued capabilities will depend on its ability to capitalize on and expand its networked C² structure to train and educate future forces in mission effects-sensitive decision making.

12.4.3 From network-centric enablement to system of systems

The link between these two fundamental concepts can be traced back to a perspective on the RMA that emphasizes the integration of advanced weapons technology and information technology and the military organization and doctrine to exploit both simultaneously. A system-of-systems point of view was taken by Admiral William Owens [6] when he considered how Command, Control, Communications, Computing, Intelligence, Surveillance, and Reconnaissance (C4ISR) should be organized to support future warfare. Specifically, he considered that the force assets for these key functions should be organized into three "systems" that must interact with each other:

- Intelligence, Surveillance, and Reconnaissance
- Command, Control, Communications, and Intelligence Processing
- Precision Force

This SoS concept is clearly enabled by the concepts of NCO.

12.4.4 SoSE for Air Force capability development

Four years after the submission of the *NCW DoD Report to Congress*, the United States Air Force Scientific Advisory Board delivered an SoSE report to the Air Force [7]. While recognizing existent policy and practices for systems engineering in the acquisition of defense systems, but given the emergence of both the concepts and concerns about systems of systems, the Board considered the question of how best to proceed with SoSE. At that time there was confusion about what constituted SoSE, and there was little in terms of codified practices that could be adapted for use within the DoD. And case studies from the commercial world were not readily applicable to the DoD. Four considerations were emphasized in the study in order to gain insight

into the way ahead: the role of the human in SoS, discovery and application of convergence protocols, motivation issues, and experimentation venues.

The role of the human in an SoS can be part of the overall design, but it could also equally result from a lack of adequate interfaces to support the interactions of the systems. Either can create a challenging environment for the human. The lack of sound human system interface designs can exacerbate the challenges. The term convergence protocol was used to characterize the election of a single protocol or standard by a group that simplifies connection among different systems. Examples include simple standards like S-video connections for entertainment products and communications protocols like the Transmission Control Protocol/Internet Protocol (TCP/IP). A highly successful convergence protocol for the DoD for SoS applications could enable the use of dynamic discovery technologies for information management among the systems. Motivations within the commercial sector that have led to the successful development of SoS solutions are not readily applicable to DoD. The Report recognized that further research was needed in order to learn how to motivate, for example, DoD program managers to seek SoS solutions. Three experimentation venues were recommended. The first was one for developing concepts of operations, the second was for the evaluation of candidate convergence protocols, and the third was focused on the rapid fielding of SoS-enabled capabilities. The enhancement of infrastructure was also seen as essential for the successful development of SoS capabilities.

12.5 System of systems or network enablement?

The development of military concepts related to the RMA over the last two decades raises the question of whether the RMA is based on SoS or is driven by network enablement.

12.5.1 A lesson from history

Using Blitzkrieg as a case study can shed some light on the question of SoS versus network enablement of military capabilities. The Blitzkrieg RMA was based on the integration of weapons and communications technology to achieve military capabilities. Each tank in a Panzer division can be considered as an autonomous system, and in fact this was fundamental to the concept of operations for Blitzkrieg. So, the tanks were a collection of systems, which rightfully could be considered to be an SoS.

But the lynchpin of the new tactic was the radio [8]. Although radios had been available to the military in World War I, they were bulky due to power supply limitations. During the time period between the World Wars, early efforts at miniaturization had reduced power demands, allowing reliable radios to be installed in both tanks and aircraft. Portable radio sets were provided as far down in the military echelons as the platoon. In every tank there was at least one radio. Although radios and communications links should

not be confused with networks, it is a simple extrapolation to apply the SoS concepts for C4ISR and precision force advocated by Admiral Owens to the concepts of Blitzkrieg. That is to say, in a more modern time, the SoS (of tanks) would have been network enabled.

12.5.2 Systems engineering considerations

Although there are various interpretations of the terms SoS and SoSE, it still is useful at this point to consider formal definitions of the term *system* and understand the implications for SoS and SoSE. The issue to be resolved is whether an accepted definition of the term *system* can be applied to itself to give a technically consistent definition of the term *system of systems*. Two well-known definitions of the term *system* will be used as case studies for this purpose. However, it is not the purpose of this chapter to advocate or challenge specific definitions of the terms of systems engineering. Fundamental issues like this are currently a subject of discussion amongst the systems engineering community.

The INCOSE definition of system will be used as a case study for this purpose, primarily because of its simplicity. The *INCOSE Systems Engineering Handbook* [9] defines the term system:

A system is a combination of interacting elements organized to achieve one or more stated purposes.

A similar definition put forth by Hitchins [10] emphasizes the nature of emergence:

A system is an open set of complementary, interacting parts with properties, capabilities, and behaviors emerging both from the parts and their interactions.

The concept of open systems (referred to via open sets in the Hatley definition) derives from biological systems, which ingest and interact with their environment, in contrast with the concept of closed systems in physics, where energy is neither gained nor lost (at the system level).

The case study for Blitzkrieg showed how radios were the lynchpin for this RMA and were the mechanism for the interaction of the tanks in a Panzer division. Consequently, when either of the definitions is considered in light of the Panzer SoS, radios and information exchange become key enablers. Will this be generally true of most or all SoSs?

The basis for answering this question and resolving the broader issue of a technically consistent definition of *system* and *system of systems* can be

expected to be resolved by modeling the definitions rather than just relying on arguments based on the natural language definitions.

Figure 12.3 provides an example of a model that satisfies both of the above definitions of *system* and aligns well with traditional principles of systems engineering. The diagram in Figure 12.3 uses graphical notations from both the Unified Modeling Language (UML) and the Systems Modeling Language (SysML). The SysML notation makes clear that the *realization* of the system derives from the organized *actions* and *interactions* of the *elements* that comprise the *combination*. This is a commonly accepted principle of systems engineering, referred to as emergent behavior.

Can the model presented in Figure 12.3 give a technically consistent definition of *system* and *system of systems*? It is clear that this model for *system* can be logically substituted for the key word *elements* in Figure 12.3. This could provide a model of the term *system of systems*. Whether this model can be interpreted to reasonably satisfy the various accepted definitions of the term *system of systems* across the systems engineering community is a separate question. But it should be clear from these arguments that the issue of a technically consistent definition will not be resolved until methods (like modeling languages) that are more rigorous than natural language are brought to bear.

With regard to the question of information exchange raised by the case study of the Panzer SoS, it is useful in the model presented in Figure 12.3 to consider what is being transported by the interactions between the elements (of the *system*). Hatley has considered that *interactions* in man-made *systems* are described by exactly three types of *transport elements*: energy, material, and information [11]. Now, if an SoS is considered to be a system whose elements are other systems, then there are exactly three types of exchanges that

A **system** is a *combination* of *interacting elements* organized to realize *properties, behaviors*, and *capabilities* that *achieve* one or more *stated purpose(s)*.

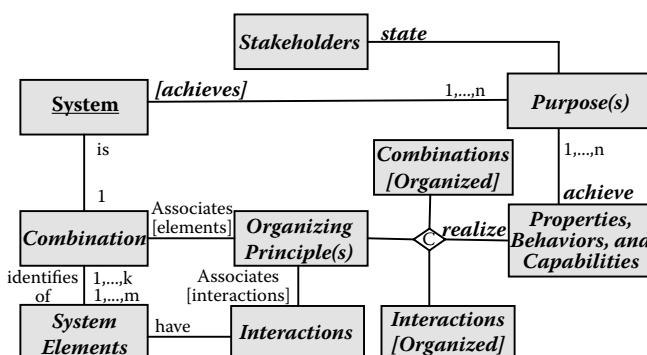


Figure 12.3 Logical model of system.

can be made between the systems. If these systems are considered individually to have any kind of autonomy, then the exchange of material or energy between the individual systems, while possible or necessary is not the primary type of exchange. Again information exchange is the critical and perhaps primary type of exchange between the systems.

The next case study will again emphasize the importance of information exchange between systems in an SoS. It will also introduce the idea that the integration of systems can occur at the capabilities level.

12.5.3 *Critical roles of information exchange and capabilities integration*

This section presents another case study, which illustrates the interplay between the concepts of SoS and network enablement: Fleet Battle Experiment–India (FBE-I), Time Critical Strike (TCS) [12]. The success of these concepts was demonstrated in actual combat by coalition land forces in Afghanistan after the attacks of September 11, 2001, and again by the Marine Corps in their swift march to Baghdad during the invasion of Iraq from March 18 to May 1, 2003. This was a marked departure from a centuries-long tradition of how the Marines had fought.

The FBEs were a series of joint experiments initiated by the Chief of Naval Operations (CNO) and designed to investigate NCO as a framework for the development of new doctrine, organizations, technologies, processes, and systems for future war fighting. The Navy Warfare Development Command (NWDC) was the CNO's agent for planning and implementing these experiments in partnership with the numbered fleets. It should be noted that shortly after the time of the FBE-I, the Navy adopted the Air Force (and ultimately joint) terminology of Time Sensitive Targeting (TST) instead of TCS.

Figure 12.4 exhibits the operational concept (more specifically the DoDAF Operational View- 1 (OV-1)) for the TCS experiment. The OV-1 may not look like the kind of operational concept that is more frequently seen, i.e., a graphic or cartoon with systems operating in a theater or scenario. This is because the planners for FBE-I were truly focused on NCO, i.e., on operations not systems. This is an operator's view of the experiment, exactly as it was intended to be executed.

There are three types of entities in Figure 12.4: high-level operational activities depicted in the context of an abstract theater, force elements and the C2 structure that supports them, and an identification of the combination of systems that enable the TCS capabilities sought in the objectives of the experiment:

- Operational activities
- Ship to Objective Maneuver (STOM)
- Time Critical Strike (TCS)
- Force elements

Littoral Penetration Task Force (LPTF)
 Special Operations Force (SOF)
 C2 structure
 Commander, Joint Task Force (CJTF)
 Joint Force Air Component Commander (JFACC)
 Joint Force Maritime Component Commander (JFMCC)
 Enabling systems
 Extended Range Guided Munition (ERGM)
 Land Attack Standard Missile (LASM)
 Tactical Tomahawk Land Attack Missile (TTLAM)
 Advanced Land Attack Missile (ALAM)
 Tactical Aircraft (TACAIR), in this case F/A-18 strike fighters

The theater is divided into two parts by a littoral interface depicted by a brown dashed line: (1) land (to the right of the line) and (2) sea (to the left). Historically the Marines would “storm the beach,” establish a safe beach head, and then build up supplies (called the iron mountain) at the beach head in preparation for further fighting to secure a larger area. Once the larger area was secure, the Army would land and take over inland operations.

In this depiction of the theater, the CJTF leads the operation. The JFMCC both secures a portion of the littoral interface (with fires, for example) and provides various weapon systems to perform TCS operations. The JFACC controls the air space, so the JFMCC and the JFACC must coordinate and collaborate. The joint forces in this OV-1 are geographically dispersed.

What has changed for the Marines in FBE-I when they become an LPTF? In this new concept they are employing classic fire and maneuver tactics in order to penetrate more deeply and swiftly than ever before. The Tactical

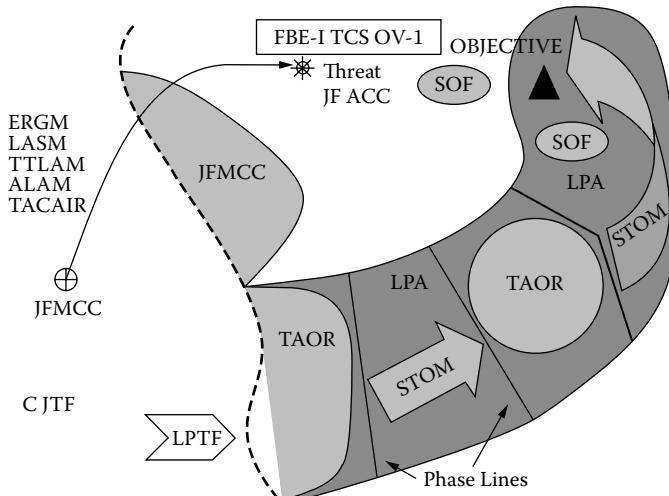


Figure 12.4 Fleet Battle Experiment-India operational concept.

Areas of Regard (TAOR in the OV-1) are areas where they expect to fight. The Littoral Penetration Areas (LPA) are areas of maneuver, where the STOM is a maneuver toward the objective. (More than one STOM may be required.) Why have they not done this before? In the past they only went a short distance and carried a more significant amount of fire power than in the LPTF concept. To penetrate deeper (to a designated objective) and more swiftly, it was necessary to leave fire power behind. So, where will the requisite fire power come from when they need it? It came from TCS in FBE-I and, in current terminology, will come from TST. They are also supported by SOF.

This experiment embodies both the tenets of NCO and the SoS concepts of Admiral Owens, where three (grand) systems of ISR, C4, and Precision Force are integrated to create emergent behaviors and military capabilities not otherwise achievable. Based on the systems engineering considerations of the previous section, it is clear that the mission-essential interactions between these three systems are based on the exchange of information. This is not to diminish the essential need for energy (in the form of kinetic weapons) and material (in the form of logistic support) to be placed into the Battlespace. However, it is the integration of the ISR, C4, and weapon systems at the mission level that enabled the new Precision Force capabilities (i.e., TCS) sought by FBE-I.

This case study should prompt the systems engineer (and the SoS engineer) to consider two important things: first, the exchange of information is both central and critical for an SoS to enable a military mission, and second, but equally important, mission capabilities enabled by an SoS (or any system, for that matter) must be understood through integration of operational capabilities in a mission context.

The concepts of capabilities engineering and integration are currently being explored by the defense community, but will face technical challenges until stricter standards are established for the terminology of systems engineering and SoSE. For example, the INCOSE natural language definition of system is stressed by the concept of capabilities integration because, if interactions are only between the elements of the system, then there is no mechanism (in the definition) for the interactions of the effects caused in the Battlespace. However, the model depicted in [Figure 12.3](#) could support concepts where the system elements interact with elements of the Battlespace (which are not part of the system). Ongoing research in the defense community can be expected to address these types of questions in the years to come.

12.6 Examples of U.S. DoD defense applications of SoS

The U.S. DoD has defined SoS through the joint staff instruction CJCSI 3170.01E, 11 May 2005, as follows:

A system of systems (SoS) is a set or arrangement of interdependent systems that are related or connected to provide a given capability.

In the DoD concept of SoS, the loss of any part of the SoS will significantly degrade the performance or capabilities of the whole. This definition clearly supports the SoS concept of Admiral Owens. A priori it exhibits no preference for the nature of the interdependence (e.g., does information exchange have primacy for enabling SoS capabilities?). And it may be a candidate definition for a technically consistent definition of *system* and *system of systems*.

12.6.1 OSD SoS SE Guide

The writing of the *System of Systems Systems Engineering Guide* was initiated in May 2006, when a task was issued by the Deputy Under Secretary of Defense for Acquisition and Technology (DUSD (A&T)) to develop a *Guide* for SoS SE. The objective was to provide guidelines to defense system program manager for a better understanding of how the system under their management might be part of a larger SoS, and how the management of a defense acquisition in an SoS environment might differ from the acquisition of a single system. Government, industry, and academia collaborated under the leadership of DUSD (A&T), and Version 0.9 of the *Guide* was released on December 22, 2006 [13]. The *Guide* describes the application of systems engineering at a system-of-systems level and has served as a mechanism for the broader community to come together, to discuss the wide spectrum of views on this topic, and capture the views that everyone does agree upon.

For the six months after its release, OSD used the *Guide* to conduct a pilot study on how SoS SE is being practiced in the DoD. The defense community was then asked to submit their comments as part of the effort to pilot the recommendations in the *Guide*. Over twenty U.S. and international stakeholder organizations from government, industry, and academia have participated in the pilot study.

Emerging insights from the pilot show the following characteristics of a system of systems in the DoD today: (1) they tend to be ongoing efforts to satisfy user capability needs through an ensemble of systems, (2) they are not new acquisitions per se, (3) the SoS manager typically does not control the requirements or funding for the individual systems, (4) the SoS is focused on the evolution of capability over time, and (5) a functioning SoS requires start-up time but, in a steady state, seems well suited to routine incremental upgrades.

The comments on the Version 0.9 release have been collected and were assembled and reviewed by the working group for the *Guide*. A new version of the *Guide* has been issued for broad review and use.

In the original OSD *SoS SE Guide* the U.S. DoD Services submitted SoS program descriptions, some of which are summarized below.

12.6.2 Future Combat System (FCS)

The U.S. Army's FCS is a new development program that is intended to operate as a cohesive system of systems where the whole of its capabilities is to be greater than the sum of its parts. As the key to the Army's transformation, the network, and its logistics, and Embedded Training (ET) systems, enable the Future Force to employ revolutionary operational and organizational concepts. The network enables soldiers to perceive, comprehend, shape, and dominate the future battlefield at unprecedented levels. The FCS network consists of four overarching building blocks: System of Systems Common Operating Environment (SoS COE); Battle Command (BC) software; communications and computers (CC); and intelligence, reconnaissance and surveillance (ISR) systems. The four building blocks synergistically interact, enabling the Future Force to see first, understand first, act first, and finish decisively.

FCS is an example of an SoS currently under development, with many degrees of freedom from an engineering perspective. A traditional SE approach (top-down analysis, synthesis, and evaluation) has been applied

12.6.3 Single integrated air picture (SIAP)

The single integrated air picture (SIAP) under development by the Joint SIAP Systems Engineering Organization (JSSEO) consists of common, continual, and unambiguous tracks of airborne objects of interest in a surveillance area. SIAP is derived from fused real-time and near real-time data and consists of correlated air object tracks and associated information from multiple sensors. The purpose of a SIAP is to solve a confusing air picture. Real-world operations, exercises, and evaluations highlight joint war fighting shortfalls. Disparate systems within a service as well as across multiple services that share track management data have different views. The views, as well as the track identification number, cause confusion in communication among system operators.

The SIAP requirements must provide a solution that is scalable and filterable and supports situation awareness and battle management. Each airborne object must have one and only one track identifier and associated characteristics. The solution will reduce the risk of fratricide to U.S. and coalition forces caused by incorrect correlation and track identifier association, provide confidence in the air picture, provide point and area defenses the opportunity to engage beyond their self-defense zones, and quickly coalesce to repel asymmetric threats such as cruise and ballistic missiles.

The SIAP will initially be introduced via Capability Drops on host platforms of the U.S. Navy Aegis, SSDS, E2, and the Air Force E3 AWACS, Battle Control System (BCS) and the RC-135V/W Rivet Joint. The Capability Drops are targeted at eliminating specific interoperability issues, providing C4I enhancements, and delivering an executable integrated architecture in a two-year spiral delivery process.

The SIAP operational Joint Battle Management and Command and Control (BM/C2) reflect new capabilities of networked interoperable heterogeneous systems as opposed to traditional monolithic systems with a single system orientation. Tactical system networks are dynamic with autonomous mobile nodes and ad hoc membership. The autonomous constituent systems make up the SIAP SoS to fulfill a specific mission that may be presented. This is a hybrid SoS, whereby there is some level of command and control and autonomous systems fall under the area commander to fulfill a mission. This is characterized by loosely coupled, independently controlled nodes sharing collaboration rules via centralized guidance.

The SIAP is implemented by applying Model Driven Architecture (MDATM) in developing an executable peer-to-peer (P2P) integrated architecture behavior model to form a platform-independent model (PIM). The services receive the PIM for their host platforms and create platform-specific models (PSM) for their hosts applying adaptive layers to interface with their specific sensors shown.

12.6.4 Naval integrated fire control–counter air (NIFC-CA)

NIFC-CA seeks to evolve a family of systems of mixed maturity to extend the Naval Theater Air and Missile Defense Battlespace to distances that are well beyond the existing, stand-alone capability of surface ship–controlled air defense weapons. NIFC-CA is a capabilities-based program that takes current and emerging technology from Core Pillar Programs (Cooperative Engagement Capability (CEC), Aegis, Standard Missile (SM-6), and E-2D Advanced Hawkeye (AHE)) and other related programs and integrates them together to form the successful implementation of a system of systems capability. The NIFC-CA focus is on integrated fire control for over-the-horizon (beyond visual range) and engage-on-remote capability. NIFC-CA is a key component of Navy Transformation “Sea Shield.” NIFC-CA is an SoS systems engineering capability designed to define the functional allocation for the pillar elements within NIFC-CA (SM-6, E-2D, CEC, and Aegis). The synchronization of programs across cost/schedule/performance is the key challenge for this SoS.

12.6.5 Commissary advanced resale transaction system (CARTS)

CARTS is the DeCA replacement point-of-sale (POS) system used to process customer purchases, capture sales and financial data from those purchases, produce management reports, and provide information to other Defense Commissary Agency (DeCA) business systems. CARTS will be a COTS system with middleware to support the interfaces to the DeCA business systems. It will also provide its customers with services similar to future grocery industry technology advances without disrupting commissary operations.

This program has a development strategy of utilizing COTS to the fullest extent and provides the complete system capability in one step, and software reuse is very common. Challenges with COTS include making the determination whether the COTS architecture can support military requirements. A careful examination of the architecture is required to determine if there is sufficient capability for growth to accommodate the additional military requirements. A CAIV (cost as independent variable) analysis is important to conduct over the complete product life cycle.

12.7 Related work and research opportunities

In addition to the breadth of work on SoSE, NCO, and NEC that is ongoing, the concepts of systems engineering itself are currently an active topic of discussion amongst the larger community. The International Organization of Standards (ISO) is engaged with issues of architecture concepts through two Working Groups: JTC1/SC7/WG42 and TC184/SC5/WG1. The International Council on Systems Engineering (INCOSE) has various initiatives to include a Special Workshop on Architecture Concepts at the International Workshop 2008 (IW08), which will also support the work of the ISO Working Groups. The IEEE has also recently initiated the International Council on Systems of Systems (ICSOS). And for the past eight years the Object Management Group (OMG) has been maturing the practices of Model Driven Architecture (MDATM), which are being linked to the practice of systems engineering through collaboration with INCOSE.

Research on the foundations of systems engineering as evidenced by INCOSE and ISO should lead to a sharper understanding of SoSE and the role of NCO, NEC, and related topics such as capabilities engineering and integration. IEEE efforts through ICSOS can be expected to further promote a broad community interest in SoSE that will both seek and advance solutions to SoS problems and extend the methods and practices of systems engineering to SoSE. Research on the extension of the OMG MDA to systems and system of systems engineering should also be an increasingly active area of active research over the next several years.

References

1. The Secretary of State for Defence. 2005. *Defence Industrial Strategy*. Defence White Paper Cm 6697, London.
2. Joint Chiefs of Staff. 1996. *Joint Vision 2010*. Government Printing Office, Washington, DC.
3. Committee on Network-Centric Naval Forces. 2000. Naval Studies Board, *Network-Centric Naval Forces: A Strategy for Enhancing Operational Capabilities*. National Academy Press, Washington, DC.
4. Alberts, D. S. et al. 1999. *Network Centric Warfare: Developing and Leveraging Information Superiority*, CCRP Press, Washington, DC.

5. Office of the Secretary of Defense. 2001. *Network Centric Warfare. Department of Defense Report to Congress*. Office of the Secretary of Defense, Washington, DC.
6. Owens, W.A. 1996. *The Emerging U.S. System-of-Systems*, Number 63. The National Defense University, Institute of National Security Studies, Washington, DC.
7. Saunders, T. et al. 2005. *Systems-of-Systems Engineering for Air Force Capability Development*, Report SAB-TR-05-04. United States Air Force Scientific Advisory Board, Washington, DC.
8. Hall, C. 2000. MITRE Corporation, private communication.
9. International Council on Systems Engineering. 2007. *INCOSE Systems Engineering Handbook*, v.3.1. INCOSE Central Office, Seattle, WA.
10. Hitchins, D. K. 2003. *Advanced Systems Thinking, Engineering, and Management*. Artech House, Boston, MA.
11. Hatley, D. J., Hruschka, P., Pirbhai, I. 2000. *Process for System Architecture and Requirements Engineering*. Dorset House Publishing, New York.
12. Dickerson, C.E. et al. 2003. *Using Architectures for Research, Development, and Acquisition*. Office of the Chief Engineer of the Navy, Assistant Secretary of the Navy, August 2003. Available via Defense Technical Information Center (DTIC): (www.dtic.mil). AD Number ADA427961.
13. Department of Defense. 2006. *System of Systems Engineering Guide*, Version 9. Director, Systems and Software Engineering, Deputy Under Secretary of Defense (Acquisition and Technology), Office of the Under Secretary of Defense, (Acquisition, Technology and Logistics).

chapter thirteen

System of air vehicles

Richard Colgren

Contents

13.1	International regions	342
13.2	Air traffic management.....	342
13.3	Air traffic control.....	343
13.4	Pilot certification	343
13.5	Aircraft certification	344
13.6	Aircraft registration.....	344
13.7	The National Airspace System	344
13.8	Airspace description criteria	345
13.8.1	Volume	345
13.8.2	Proximity	345
13.8.3	Time	345
13.8.4	Attributes	345
13.9	Flight rules	345
13.10	ICAO airspace categories	346
13.11	Overview of ICAO/FAA airspace designations	347
13.12	Description of ICAO/FAA airspace classifications	347
13.12.1	Class A airspace	348
13.12.2	Class B airspace.....	348
13.12.3	Class C airspace	349
13.12.4	Class D airspace	349
13.12.5	Class E airspace.....	349
13.12.6	Class F airspace.....	350
13.12.7	Class G airspace	350
13.13	Enroute structures	350
13.14	Low-altitude airways (Victor airways)	350
13.15	Jet routes	351
13.16	VFR flyways.....	351
13.17	VFR transition routes.....	351
13.18	Air traffic control assigned airspace	352
13.19	VFR waypoints chart program	352
13.20	Special use airspace	352
13.20.1	Prohibited areas	352

13.20.2	Restricted areas	353
13.20.3	Military operations areas	353
13.20.4	Alert areas.....	354
13.20.5	Warning areas	355
13.20.6	Controlled firing areas.....	356
13.21	Military training routes	356
13.22	Other military airspace structures	357
13.22.1	Slow routes.....	358
13.22.2	Low altitude tactical navigation areas.....	359
13.22.3	Local flying areas.....	359
13.22.4	Aerial refueling routes	359
13.22.5	Temporary special use airspace.....	360
13.22.6	Cruise missile routes.....	360
13.22.7	National security areas	360
13.23	Oceanic and offshore operations	360
13.24	Air defense identification zone	361
13.25	Environmentally sensitive areas.....	361
13.26	Airdrops	362
13.27	Special federal aviation regulation.....	362
13.28	Noise abatement procedures	362
13.29	Air traffic control communications	362
13.30	Summary and conclusions	363
	Acknowledgments	363
	References	364

The global ICAO airspace provides an excellent example of a system-of-systems (SoS) concept. This global airspace is composed of 190 coordinated National Airspace Systems (NASs). Each NAS is one part of a member nation's National Transportation System (NTS). A NTS appears as layered networks composed from heterogeneous systems [1]. They achieve synergy by enabling individuals and groups to operate within each NTS composing this global system to achieve the desired overall system performance with an extremely high safety rate. This concept has been successfully implemented and operated over many years. To successfully operate within this airspace it is necessary for the aircraft operator to understand applicable regulations and their structure. The regulations prescribe the training required by aircraft operators based on the flight activities within which they participate. Regulations cover operating standards and flight rules, airspace classes, airways, and routes. Aircraft certification requirements are based on the type of flight activities for which they are designed and constructed. The aircraft's systems must all be certified to comply with these regulations and standards.

Benefits are derived from a SoS approach to the global ICAO airspace. Published papers detail specific benefits to this approach [1–5]. A SoS analysis allows for consideration of scope dimensions (e.g., multimodal impacts

and policy, societal and business enterprise influences) and network interactions (e.g., layered dynamics within a scope category). This treatment accommodates the higher level interactions seen within each NTS as well as at the global ICAO airspace architecture level. This ensures that modifications to the Air Transportation System (ATS) will have the intended effect [1]. This ultimately leads to improved outcomes from high-consequence ATS technological, socioeconomic, operational, and political decisions [2]. This is especially true in efforts to transform air transportation by incorporation of superior transportation architectures and technologies within an evolutionary framework [4].

Air transportation networks consist of concourses, runways, parking areas, airlines, cargo terminal operators, fuel depots, retailers, catering establishments, etc. [1]. SoS methodologies are required to rapidly model, analyze, and optimize air transportation systems [4]. By approaching such problem spaces from a SoS perspective, we can find the proper balance between such system characteristics as risk, flexibility, and productivity. The SoS characteristics of the ICAO/FAA airspace are shown in Table 13.1. Its successful operation as a SoS requires standardized communications among individuals and groups across enterprises through protocols approved by regulations and verified by certification.

Table 13.1 SoS Characteristics of the ICAO/FAA Airspace

	Individual Enterprise	or Group	Communications	Protocol
Air Traffic Mgt.	X			X
Aircraft Oper.	X	X		
Pilot Cert.		X		X
Aircraft Cert.		X		X
Aircraft Reg.				X
Airspace System	X			
Flight Rules			X	X
Airspace Cat.				X
Enroute Str.				X
Victor Airways				X
Jet Routes				X
VFR Flyway				X
VFR Transition Routes				X
Special Use Air.				X
Temporary Special Use				X
National Security Areas				X
Air Defense Id. Zone				X
Air Traffic Coms.			X	

The International Civil Aviation Organization (ICAO), which manages the international Air Traffic Control (ATC) system, has been in operation since April 1947. It was built around pre-existing international agreements and regulations as an agency of the United Nations [6]. It codifies the principles and techniques of international air navigation and fosters the planning and development of international air transport to ensure safe and orderly growth [7]. It ensures standards, such as the international use of English for communications and the specific frequency ranges to be used for these and all other required command and control operations. The ICAO Council adopts standards and recommended practices for member states concerning air navigation, the prevention of unlawful interference, the facilitation of border crossings for civil aviation, and the protocols for air accident investigations used by transport safety authorities. These protocols are also known as the Chicago Convention.

The FAA provides for the safe separation of aircraft within U.S. airspace, including in and out of U.S. airports [8]. While the basic design of the U.S. ATC system dates back to the 1950s, this design has been adapted as demands on the system's capacity have continued to rise. There are three critical components of the ATC: communications, navigation, and surveillance. These are continuously modernized to maintain the safety and efficiency of the air transportation system. Such updates must be accomplished under international agreements to maintain global ATC compatibility.

13.1 International regions

The International Civil Aviation Organization (ICAO) has divided the world's airspace into multiple regions served by seven regional offices [6]. These regions and the locations of their offices are as follows: (1) Bangkok, Thailand, serving the Asia and Pacific regions, (2) Cairo, Egypt, serving the Middle East, (3) Dakar, Senegal, serving Western and Central Africa, (4) Lima, Peru, serving South America, (5) Mexico City, Mexico, serving North America, Central America, and the Caribbean, (6) Nairobi, Kenya, serving Eastern and Southern Africa, and (7) Paris, France, serving Europe and the North Atlantic. Each region has several national airspace control organizations that are members of the ICAO, such as the FAA for the United States within North America, served by the Mexico City office.

13.2 Air traffic management

Air traffic management is the means by which the flow of air traffic is orchestrated within each National Airspace System (NAS). It is based on capacity and demand. This is accomplished by using a systems approach that considers the impact of individual actions on the whole [9]. Traffic management personnel consider who or what may be impacted to focus on a coordinated effort to ensure equity in delivering air traffic services. Air traffic managers

can then use tools, known as Traffic Management Initiatives (TMIs) to help manage the flow of air traffic. Examples of these are: Special Traffic Management Programs (STMPs), Ground Delay Programs (GDPs), Airspace Flow Programs (AFPs), Miles-in-Trail (MIT) Restrictions, and Coded Departure Routes (CDRs).

13.3 Air traffic control

Air traffic control (ATC) is a service provided by ground-based controllers, who direct aircraft on the ground and in the air. An air traffic controller's primary task is to separate aircraft to prevent them from coming too close to each other by enforcing lateral, vertical, and longitudinal separation standards. Their secondary tasks include ensuring safe, orderly, and expeditious flow of traffic, and providing information to pilots, such as weather, navigation information, NOTAMs (NOTices To AirMen), and traffic advisories to Visual Flight Rules (VFR) traffic. In many countries, ATC services are provided throughout the majority of airspace, and its services are available to all users (private, military, and commercial). When controllers are responsible for separating some or all aircraft, such airspace is called a *controlled airspace* in contrast to an *uncontrolled airspace*, where aircraft may fly without the use of the air traffic control system. Depending on the type of flight and the class of airspace, ATC may issue instructions that pilots are required to follow, or merely flight information (in some countries known as advisories) to assist pilots operating within the airspace. In all cases, the pilot in command has final responsibility for the safety of the flight and may deviate from ATC instructions in an emergency. To ensure communication, all pilots and all controllers everywhere are required to be able to speak and understand English, although they may use any compatible language [10].

13.4 Pilot certification

To ensure safe aircraft operations, pilots are required to undergo training to approved standards. Once this training is successfully completed and the pilot's performance is verified, pilots are appropriately certified [11]. Pilot certification in the United States is accomplished under the direction of the appropriate FAA region, following their certification requirements documents such as Federal Acquisition Regulations (FAR) Part 61 [12]. Using such controlling documents and regulations means that a great majority of this work can be accomplished by local, non-FAA personnel. Special variations on this process exist for some aircraft types and methods of operation. The general groupings of pilot certification requirements are: (1) no pilot certification, such as with ultralight aircraft, (2) required ground training on regulations and conventional aircraft operations, (3) required ground training and instructor sign-off for unsupervised solo operations, (4) successful passage of a written test, such as the FAA glider pilot written examination,

(5) issuing a special pilot certificate by the FAA based on satisfactory completion of an examination and observed performance, and (6) conforming to the certification requirements of FAR Part 61 [12] for student and private pilots. More advanced ratings, such as for commercial operations and for operating large, heavy aircraft, also exist. Airman certification for the operators of air vehicles, with some exceptions, is required. This is to ensure that all airmen meet aeronautical knowledge, age, medical, and experience requirements for operating these vehicles.

13.5 Aircraft certification

In addition to regulations for type and operation, aircraft/equipment and/or their component parts are required to meet the airworthiness certification standards specified for aircraft [9]. Special variations on this process exist for some aircraft types and methods of operation, such as ultralights or experimental aircraft.

13.6 Aircraft registration

Most air vehicles, with special exceptions based on air vehicle type and operation, must be nationally registered and are required to display their registration number. The reasons for registration center on the identification of any offenders of airspace operational rules. The ICAO's and the FAA's experience in identification of offenders and processing enforcement action validates this need for vehicle identification. National registration systems are immediately accessible to the FAA and the ICAO for such enforcement actions [11].

13.7 The National Airspace System

The world's navigable airspace is divided into three-dimensional segments, each of which is assigned to a specific class. Most nations adhere to the classifications specified by the ICAO and described within this chapter. Individual nations also designate special-use airspace, which places additional rules on air navigation for reasons of safety or national security [13,14]. The National Airspace System (NAS) within the United States consists of all airspace above the ground up to 60,000 feet Mean Sea Level (MSL). Additionally, although the NAS officially only extends to 60,000 feet MSL, some special-use airspace extends to exo-atmospheric limits, and the FAA does want to be aware of vehicles operating above the NAS. Despite the apparent vastness of this resource, it has become crowded (in some places), and competition for its use is increasing. By law, the FAA is the controlling authority for all airspace within the United States. To provide orderly and safe use of this airspace, the FAA has developed and published numerous regulations which are found within Chapter 14 of the U.S. Code of Federal Regulations [15]. Part 11 prescribes the procedures to be followed in the initiation, administrative

processing, issuance, and publication of rules, regulations, and FAA orders. Part 71 designates the airspace structure including airspace classes, airways, routes, and reporting points. Part 73 designates special use airspace and prescribes the requirements for the use of that airspace.

13.8 Airspace description criteria

When defining a section of airspace, four criteria are considered [13].

13.8.1 Volume

Volume is a key concept to understanding the amount of airspace actually being used [13]. The length and width of airspace are visible on a two-dimensional map, but the floor and ceiling must also be included to see the complete picture, as airspace is defined using three dimensions. Airspace used for flight operations could begin as low as the surface and extend upward to 60,000 feet MSL. This characteristic of airspace enables numerous users to safely operate at different altitudes within the same geographical location.

13.8.2 Proximity

Airspace is often associated with geographic areas, such as an airport, an airfield, or a military installation. Proximity affects the utility of the airspace [13].

13.8.3 Time

Airspace is allotted for use at specific times. A designated airspace can be used to separate unusual flight maneuvers from other aircraft [13].

13.8.4 Attributes

Airspace attributes describe the physical characteristics of the underlying land that make certain pieces of airspace unique [13]. Those attributes might be ranges needed to meet testing and training requirements. These can include open water, desert, or mountains.

13.9 Flight rules

The terms *instrument flight rules* (IFR) and *visual flight rules* (VFR) are used to denote two different types of aircraft operations requiring different equipment onboard the aircraft and different levels of pilot certification [16]. General aviation aircraft flying between local airports, sight-seeing, etc. comprise the majority of flying completed under VFR rules. VFR operations generally allows pilots to fly off published routes using visual references such as highways, power lines, railroads, etc. In order to fly under VFR rules, the weather

must meet or exceed the minimum requirements. This generally means there must be at least three miles of visibility, and the pilot must be able to remain clear of clouds by at least 500 feet MSL.

The actual minimum requirements depend on the exact airspace classification. VFR flight is restricted to altitudes below 18,000 feet MSL and does not require flight clearances from Air Traffic Control (ATC). Note that "Special VFR" is an exception to this rule. VFR pilots exercise "See and Avoid" clearance precautions, which means that they must be vigilant of their surroundings and alter their course or altitude, as necessary, to remain clear of other traffic, terrain, populated areas, clouds, etc.

IFR operations require pilots to be trained and certified in IFR navigational methodologies and to adhere to ATC clearances containing specific flight route and altitude directions. ATC provides clearances for these IFR operations. ATC uses radar and navigational aids to keep aircraft operating under IFR rules separated from each other. Regardless of whether operating under VFR or IFR rules, the pilot in command is ultimately responsible for the safe operation of the aircraft and can deviate from ATC instructions to maintain safe operation.

Within the United States [12] Part 91 prescribes general operating and flight rules governing the operation of U.S. registered aircraft both within and outside of the United States. Subsections of Part 91 of Title 14 of the Code of Federal Regulations (14 CFR) [15] that may be of particular importance are: (1) careless or reckless operation, (2) operating near other aircraft, (3) right of way rules (except water operations), and (4) minimum safe altitudes (91.119).

13.10 ICAO airspace categories

Each nation's airspace is divided into two broad categories, *controlled* and *uncontrolled* airspace [16]. Within these two categories, there are a variety of ICAO classifications which determine flight rules, pilot qualifications, and aircraft capabilities required in order to operate within any airspace. On March 12, 1990, the ICAO adopted the current airspace classification scheme. The classes are defined in terms of flight rules and interactions between aircraft and ATC. Classes A through E are referred to as controlled airspace. Classes F and G are referred to as uncontrolled airspace. The specific classification of any area within the United States is determined by the FAA following ICAO classifications and is broadly based upon the following:

- Complexity or density of aircraft movements
- Nature of operations conducted within the airspace
- Required level of safety
- National and public interest

It is important that all pilots, dispatchers, and system managers be familiar with the operational requirements for each of the various types of airspace.

This is required to assess potential impacts on the ground activity underlying them and potential conflicts for any aircraft operating within that airspace. Both the pilot and the dispatcher must be familiar with all points of contact regarding controlled and Special Use Airspace. Because of the complexity of the airspace, it is not realistic to expect a single point of contact to solve all airspace coordination issues. Each type of airspace has its own designated unit that is responsible for controlling, scheduling, and/or coordinating the use of the designated portion of the NAS. This information is published and available to the pilots, dispatchers, and system managers whose training includes procedures for efficiently accessing this information and using it.

13.11 Overview of ICAO/FAA airspace designations

Each national aviation authority determines how it uses the ICAO classifications within its airspace [14]. In some countries, the rules are modified slightly to fit the airspace rules and air traffic services that existed before ICAO standardization. The U.S. adopted a slightly modified version of the ICAO system on September 16, 1993, when regions of airspace designated according to older classifications were converted wholesale. The exceptions within the United States are some Terminal Radar Service Areas (TRSA), which have special rules and still exist in a few places. The ICAO airspace is now structured and managed within the United States by the FAA according to the major categories and subcategories described in the following sections.

13.12 Description of ICAO/FAA airspace classifications

The primary designation of airspace utilized within the NAS is *class*. There are seven classes, "A" through "G" (see Figure 13.1). In addition to classes,

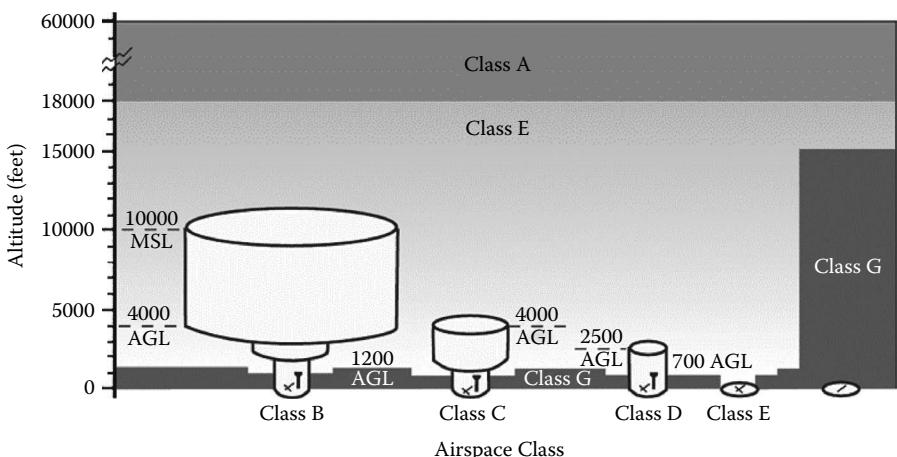


Figure 13.1 FAA airspace classifications.

there are a variety of terms utilized to identify operational structures, hazards, and unique areas within the airspace [13]. *Controlled* and *uncontrolled* airspace are generic terms that broadly cover all airspace. These refer to the level of air traffic control required to operate within the airspace. Most controlled airspace has specific, predetermined dimensions. Uncontrolled airspace can be of almost any size. Class F and Class G are the only uncontrolled airspace classes, and the only uncontrolled airspace class used in the U.S. is Class G airspace. Except as noted in the following descriptions, the FAA normally is the controlling agency for each area of the NAS within the United States.

13.12.1 Class A airspace

Class A airspace [11,13,14] include airspace from 18,000 feet Mean Sea Level (MSL) up to 60,000 feet MSL, including the airspace overlying the waters within 12 nautical miles (NM) of the coast of the 48 contiguous United States and Alaska (but not Hawaii). All operations within Class A airspace must be under Instrument Flight Rules (IFR). They are under the direct control of air traffic controllers. Class A airspace starts at 18,000 feet MSL and is not specifically charted or designated on commonly used maps. The flight activity within this airspace is used for the point-to-point transportation of passengers and cargo. All flights in Class A airspace are under positive air traffic control with communications using defined radio frequencies.

13.12.2 Class B airspace

Class B airspace surrounds the busiest commercial airports [11,13,14]. This is the most congested airspace and has the most complex mix of aircraft operations. At its core, it extends from the surface to 10,000 feet MSL. The shape of Class B airspace looks like an upside-down wedding cake with several layers. Each layer is divided into sectors with dimensions and shape tailored to meet local traffic and safety needs. The outer limit of Class B airspace can extend to 30 nautical miles (NM) from the primary airport. Air traffic control (ATC) clearance is required to operate in Class B airspace areas. To increase safety, the airspace is designed to minimize the number of turns aircraft are required to perform as they descend to an airport, while still enabling other aircraft to safely transition the area. Class B airspace is depicted on sectional charts, IFR enroute (low altitude) charts, and terminal area charts. Flight operations within Class B airspace are generally very complex and require considerable planning and coordination. Temporary Flight Restriction (TFR) coordination within U.S. Class B airspace must be carefully coordinated with the FAA due to a significant impact on the airport. A TFR will generally not be issued in Class B airspace areas because the area is already a controlled airspace. Operations must be with ATC clearance.

13.12.3 Class C airspace

Class C airspace surrounds busy airports which service mid-sized cities with a large number of commercial flight operations. They also surround some military airports [11,13,14]. An operating control tower at the primary airport and radar services are key components of Class C airspace. The overall shape is also that of an upside down wedding cake with two layers. The inner ring has a radius of 5 NM and is from the surface up to, but not including, 4,000 feet above airport elevation. The outer ring has a radius of 10 NM and is from 1,200 feet AGL to 4,000 feet above airport elevation. A third ring with a 20 nautical mile radius exists in which ATC provides traffic separation services to VFR aircraft who voluntarily request this service.

Radio communications must be established with ATC prior to entering Class C airspace, but specific permission to operate within the airspace is not required as it is in Class A or Class B airspaces. Class C airspace is depicted on sectional charts, IFR enroute (low altitude) charts, and in specific terminal area charts. Agency flight operations within Class C airspace should be viewed as complex and will normally require planning and coordination similar to that for operations in Class B airspace. TFR requests within Class C airspace must be carefully coordinated with the FAA.

13.12.4 Class D airspace

Class D airspace is applied to airports with operating control towers, but where the traffic volume does not meet Class C or Class B airspace standards [11,13,14]. Traffic usually lacks heavy jet transport activity, but often includes a complex mix of general aviation, turboprop, and business jet traffic. Radar service is often available.

Class D airspace encompasses a five nautical mile radius surrounding an operational control tower from the surface up to, but not including, 2,500 feet AGL. Class D airspace may have one or more extensions to accommodate IFR traffic. Where radar service is available, ATC will provide separation service to IFR traffic and to participating VFR traffic. All traffic must maintain radio communication with the tower or have prior arrangements for operating within Class D airspace. Class D airspace is depicted on sectional charts and on IFR enroute (low altitude) charts. Flight operations must be coordinated with the control tower. A large amount of civilian and military flight training occurs in and around Class D airspace.

13.12.5 Class E airspace

Class E airspace exists primarily to assist IFR traffic [11,13,14]. It includes all airspace from 14,500 feet MSL up to, but not including, 18,000 feet MSL. It extends upward from either the surface or a designated altitude to the overlying or adjacent controlled airspace. Radar coverage may or may not

be available. There are no requirements for VFR communications with ATC. Class E airspace below 14,500 feet MSL is plotted on sectional charts, terminal area charts, and IFR Enroute Low Altitude Charts. Aviation operations should be coordinated with the applicable Air Route Traffic Control Center (ARTCC) or Terminal Radar Approach CONtrol (TRACON). This will help to avoid conflicts with IFR traffic.

13.12.6 Class F airspace

Class F airspace is designated by the ICAO as uncontrolled airspace [11,13,14]. This airspace classification is not utilized within the United States. Operations may be conducted under IFR or VFR flight rules. ATC separation will be provided, so far as practical, to aircraft operating under IFR flight rules. Traffic information may be given as far as is practical with respect to other flights. As an example, in Germany, Class F airspace is used for IFR flight in uncontrolled airspace. In Canada, which generally follows the United States in its application of airspace classes, the term Class F airspace is used for Special Use Airspace; this includes Advisory Airspace and Restricted Airspace.

13.12.7 Class G airspace

Class G airspace is uncontrolled airspace [11,13,14] and includes all airspace not otherwise designated as A, B, C, D, or E (or F internationally). It is virtually nonexistent in the eastern United States, but relatively large blocks of Class G airspace can be found in some areas of the west and Alaska. Operations within Class G airspace are governed by the principle of “see and avoid.”

13.13 Enroute structures

Enroute structures consist of several routing corridors, essentially “highways in the sky,” utilized by both IFR and VFR traffic [13]. Relatively large amounts of traffic are concentrated along these routes.

13.14 Low-altitude airways (Victor airways)

Victor airways are the primary “highways” utilized by both IFR and VFR traffic [13]. They are 8 NM wide and generally range from 1,200 feet AGL up to, but not including, 18,000 MSL. The airway floor varies to ensure that aircraft operating on the airway remain clear of ground obstructions and have the ability to receive the radio signals from the navigational facilities. They are depicted on sectionals as blue shaded lines with a “V” (hence the name “Victor”) followed by a number (i.e., V-500, see [Figure 13.2](#)).

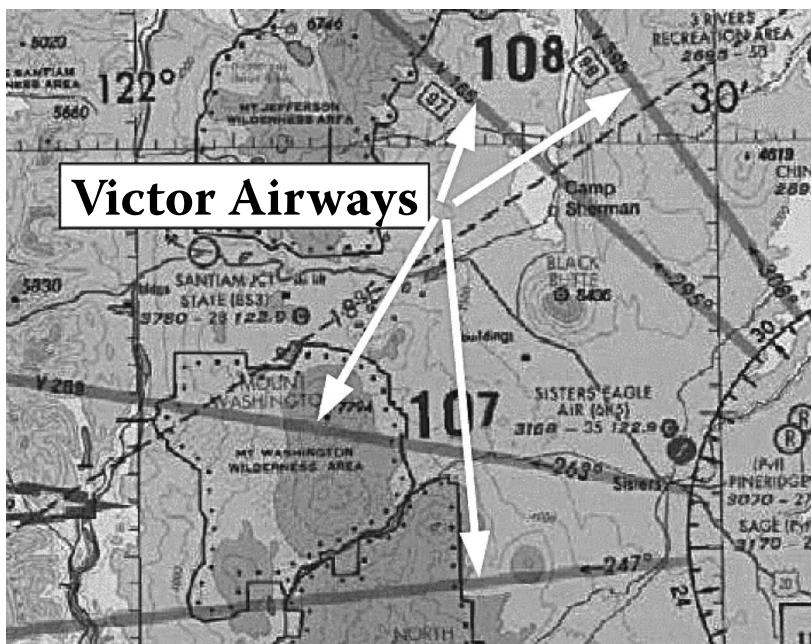


Figure 13.2 Victor airways.

13.15 Jet routes

Jet routes serve the same function as Victor low-altitude airways, except that they are found between 18,000 MSL and to 45,000 MSL [13]. Traffic on a jet route is always IFR and is managed by air traffic control. Jet routes are shown on the high altitude charts as a gray line and are represented by the letter "J" followed by a number.

13.16 VFR flyways

These are general routes for VFR traffic wishing to fly through, or near [13], Class B airspace. The intent is to provide VFR aircraft with a way to transition the airspace. An air traffic control clearance is not required to utilize a flyway. Flyways may be charted on the back of terminal area charts. The best way to locate a flyway is to ask the ATC facility controlling the Class B airspace area.

13.17 VFR transition routes

These are similar to VFR flyways and are used to accommodate VFR traffic transitioning certain Class B airspace [13]. They differ from a VFR flyway in

that an ATC clearance is required to operate within the route. Also, radar separation service is always provided. VFR transition routes are identified by a notation on terminal area charts.

13.18 Air traffic control assigned airspace

Air traffic control assigned airspaces (ATCAA) were established to permit the continuation of MOA activities above 18,000 feet MSL [13]. From the standpoint of an aircraft within a MOA, an MOA and an ATCAA are combined into one airspace, with 18,000 feet MSL acting as an administrative boundary. Usually, the ATCAA is activated concurrently with the MOA. VFR aircraft are permitted to enter a MOA, but are not permitted to enter most ATCAAs because they are not permitted to fly VFR above 18,000 feet MSL. A MOA is depicted on aeronautical charts, an ATCAA is not depicted.

13.19 VFR waypoints chart program

The VFR waypoint chart program was established to provide VFR pilots with a supplemental tool to assist with position awareness [13]. The program was designed to enhance safety, reduce pilot deviations and provide navigation aids for pilots unfamiliar with an area in or around Class B, Class C, and special use airspace. The name of a VFR waypoint (for computer entry and flight plans) consists of five letters beginning with "VP."

13.20 Special use airspace

The designation special use airspace (SUA) is designed to alert users about areas of military activity, unusual flight hazards, or national security needs, and to segregate that activity from other airspace users to enhance safety [13]. While most SUAs involve military activity, others involve civilian users. SUAs are established by the FAA. Detailed information regarding the process for establishing SUA and other types of airspace is contained in [17]. The Department of Defense (DoD) flight information publication [18] contains detailed information about current SUA. There are six different SUAs: (1) prohibited areas, (2) restricted areas, (3) military operations areas, (4) alert areas, (5) warning areas, and (6) controlled firing areas.

13.20.1 Prohibited areas

A prohibited area (PA) is established over sensitive ground facilities such as the White House [13]. The dimensions of the prohibited area vary. All aircraft are prohibited from flight operations within a prohibited area unless specific prior approval is obtained from the FAA or the controlling agency. Prohibited areas are charted on sectionals, IFR enroute charts, and terminal area charts. They are identified by the letter "P" followed by a number.

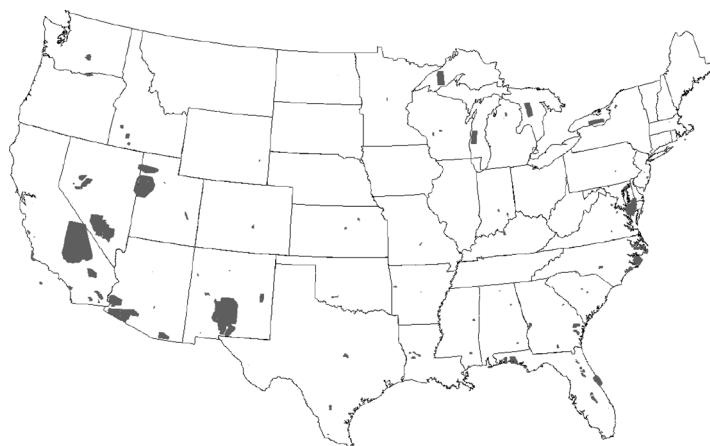


Figure 13.3 Restricted areas within the United States.

13.20.2 *Restricted areas*

A restricted area (RA) is established in areas where on-going or intermittent activities occur which create unusual, and often invisible, hazards to aircraft such as artillery firing, aerial gunnery, practice bomb dropping, and guided missile testing [13]. Dimensions of the restricted areas vary depending upon the needs of the activity and the risks to aircraft. Examples of restricted areas within the U.S. are shown in Figure 13.3.

Restricted areas differ from prohibited areas in that most RAs have specific hours of operation, and entry during these hours requires specific permission from the FAA or the controlling agency. In addition, there may be a separate scheduling agency who must also grant permission. Aviators must understand that hazardous flight activity is occurring in the RA when it is active. Restricted areas are depicted on sectionals, IFR enroute charts, and terminal area charts. They are identified by the letter "R" followed by a number. The floor and ceiling, operating hours, and controlling agency for each restricted area can be found in the chart legend.

13.20.3 *Military operations areas*

A military operations area (MOA) is an area of airspace designated for military training activities [13]. MOAs were established to contain certain military activities such as air combat maneuvers, intercepts, etc. Civilian VFR flights are allowed within a MOA even when the area is in use by the military. Air traffic control will separate IFR traffic from military activity. A clearance is not required for VFR operations. Users may encounter high-speed military aircraft involved in flight training, abrupt flight maneuvers, and formation flying. Military pilots conducting training within an active

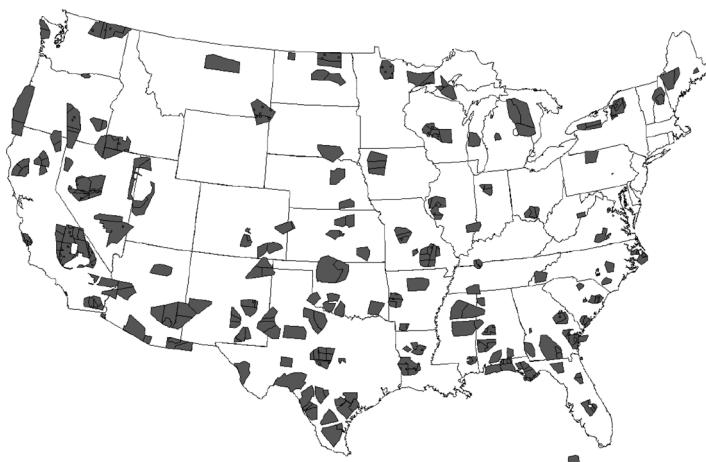


Figure 13.4 Military operations areas within the United States.

MOA are exempt from the provisions of the Federal Aviation Regulations prohibiting acrobatic flight within federal airways and control zones. They are also exempt with respect to the Federal Aviation Regulations for flights at speeds in excess of 250 knots below 10,000 feet MSL. MOAs have a defined floor and ceiling which can range up to the floor of Class A airspace (18,000 feet). MOAs are identified by a specific name, the letters "MOA," and are charted on sectionals, IFR enroute charts, and terminal area charts. MOA dimensions, hours of use, and controlling agency can be found in the chart legend. Restricted Areas within the United States are shown in Figure 13.4.

Note that MOAs can be "stacked" on top of each other. The status of an MOA can change rapidly and should be checked frequently when flight operations are occurring. MOAs will have a scheduling agency responsible for scheduling all military flights intending to use the airspace. If the scheduling agency does not have a continuous point of contact, then an alternate scheduling agency will be designated. Flight Information Publication AP/1A [18] is used for scheduling MOA information. Communications are normally established with the controlling agency of any MOA during flight operations in proximity to a MOA, even if the MOA is not active. Pilots contact air traffic control prior to entering an MOA to get the most current status information.

13.20.4 Alert areas

An alert area (AA) may contain a high volume of pilot training or an unusual type of aerial activity (see [Figure 13.5](#)). There are no special requirements for operations within alert areas, other than heightened vigilance [13]. All operations must be in compliance with FAA regulations. The types of flying involved could be military, flight testing by aircraft manufacturers, or a high concentration of flights (i.e., helicopter activity near oil rigs). Alert areas are

depicted by defined areas marked with the letter "A" followed by a number. Alert area dimensions differ for each area and are depicted on sectional charts, IFR enroute charts, or terminal area charts.

13.20.5 Warning areas

A warning area (WA) contains the same kind of hazardous flight activity as a restricted area, but they have a different title since they are located offshore over domestic and international waters (see Figure 13.6). Examples of likely hazards include artillery firing, aerial gunnery, guided missile exercises, and fighter interceptions [13]. Warning areas generally begin three



Figure 13.5 Alert areas within the United States.



Figure 13.6 Warning areas within the United States.

miles offshore. U.S. Executive Order 10854 [19] extends the application of the amended Federal Aviation Act of 1958 [20] to the overlying airspace of those areas of land or water outside the United States beyond the 12-mile offshore limit. It includes areas where the United States has appropriate jurisdiction or control under international treaty agreement.

Warning areas overlying the territorial waters of the United States are under FAA jurisdiction. However, any airspace or rulemaking action that concerns airspace beyond the 12-mile offshore limit from the United States requires coordination with the U.S. Departments of Defense and the adjacent State. Although VFR operations are permitted within warning areas, the FAA does not guarantee traffic separation, and aircraft operators weigh the risks of such operations. Warning areas are represented on sectionals, IFR enroute charts, and some terminal area charts. They are depicted by a "W" with a number following it. Dimensions for each warning area can be determined by consulting the appropriate chart legend.

13.20.6 *Controlled firing areas*

A controlled firing area (CFA) contains civilian and military activities which, if not contained, could be hazardous to nonparticipating aircraft [13]. These include rocket testing, ordinance disposal, small arms fire, chemical disposal, blasting, etc. CFAs are differentiated from MOAs and restricted areas in that radar or a ground lookout is utilized to indicate when an aircraft might be approaching the area. All activities are then suspended. The FAA does not chart CFAs because a CFA does not require a nonparticipating aircraft to change its flight path. Airspace users may find information about CFAs from the nearest regional FAA headquarters.

13.21 *Military training routes*

A military training route (MTR) is designed for low-level, high-speed terrain-following training missions. These routes are provided for military training at speeds of more than 250 knots and at altitudes that range from ground level (surface) to 18,000 feet MSL, though most operations are conducted well below 10,000 feet MSL. There are more than 500 routes, roughly divided in half for VFR and IFR operations (see [Figure 13.7](#)). They pose flight hazards to any uncoordinated aviation mission within their perimeters.

A complete description of the MTRs [21] includes the originating/scheduling activity, the hours of operation, the geographical points of each segment, the altitude limitations for each segment, the route width, special operating procedures, and the flight service stations (FSS) within 100 NM that have current information on activities within the MTR. Additional symbols indicate entry/exit points for military aircraft, turning points, departure routes, etc. In addition to charted route altitudes, additional restrictions or changes in width may be imposed to avoid sensitive areas or other conditions of use.

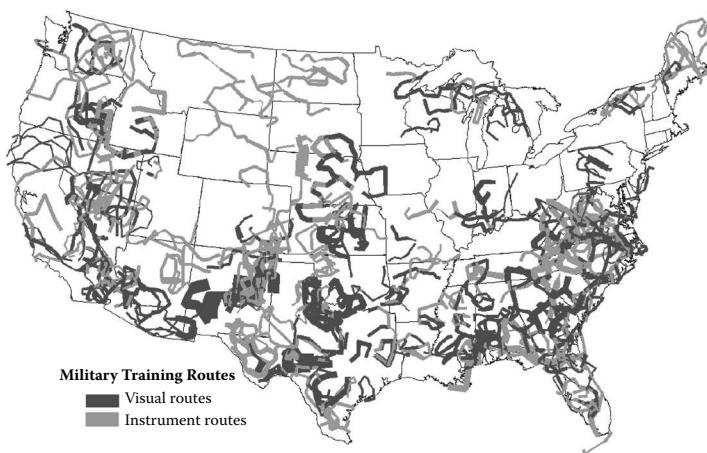


Figure 13.7 Military training routes within the United States.

These restrictions are published [21] under the standard operating procedures (SOPs) for each route.

An MTR may have a designated segment where DoD aircraft may perform various maneuvers dictated by operational requirements. Aircraft may freely maneuver within the lateral and vertical confines of the MTR segment before resuming flight on the remainder of the route. There are also designated areas within an MTR that indicate alternate exits and entrances. This accommodates a training mission that might require use of an MOA or an airport. Low Altitude Air to Air Training (LOWAT) refers to maneuvers within MTRs for the purpose of simulating an aerial attack and defense response.

Flight planning should take into account the existence of MTRs and the in-flight risks they pose. The FSS will have information available on these activities to include times of scheduled activity, altitudes in use on each route segment, and route width. Often the FAA will only have the schedule as received from the military the night before. Military pilots check in prior to entry on IFR MTRs. However, they are not required to check in with ATC prior to entering VFR MTRs. All MTRs must be scheduled through the assigned scheduling activity prior to use. Flying an MTR (in excess of 250 knots) without being scheduled is a violation of FAA regulations [12].

13.22 Other military airspace structures

Due to the unique nature of military training operations, training, and testing requirements, other airspace for special military use has been developed outside the special use airspace (SUA) program [13]. These include: (1) slow routes (SR), (2) low altitude tactical navigation areas (LATN), (3) local flying areas, (4) aerial refueling routes, (5) temporary special use airspace (TSUA),

(6) cruise missile routes, and (7) national security areas (NSAs). All of the SUAs and SRs are shown with the MTRs in Figure 13.8.

13.22.1 Slow routes

A slow-speed low-altitude training route or Slow route (SR) is used for military air operations flown from the surface up to 1,500 feet above ground level (AGL) at air speeds of 250 knots or less [13]. Route widths are published in individual route descriptions [21] and may vary (see Figure 13.9). Slow routes

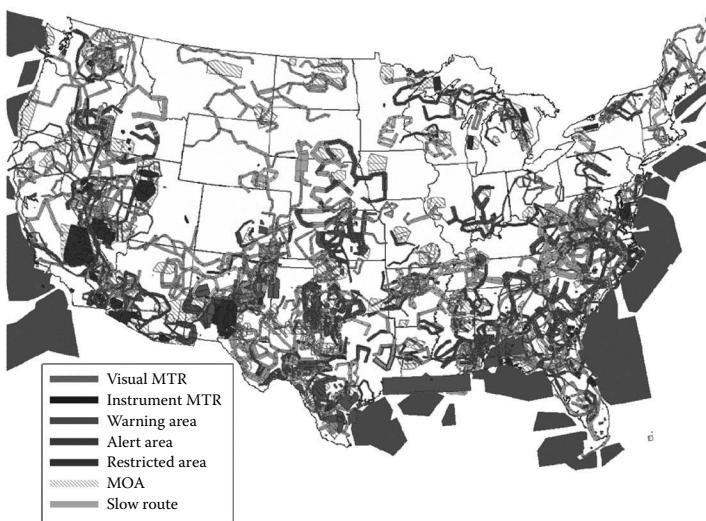


Figure 13.8 MTRs, SUAs, and SRs within the United States.

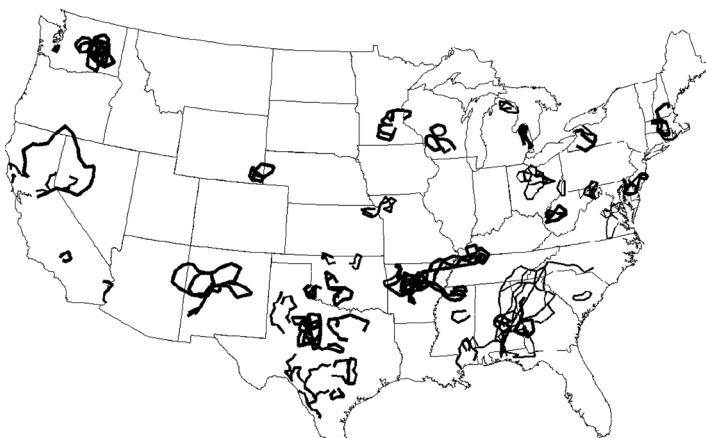


Figure 13.9 Slow routes within the United States.

technically are not considered MTRs. High-speed aircraft are not allowed to use slow routes. Generally, these routes are utilized by the U.S. Air Force. Many of these routes are flown by cargo aircraft, such as C-17s, that use drop zones for military purposes. There are about 200 slow routes in the United States. They are represented on AP/1B charts [21]. They are not depicted on sectionals.

13.22.2 Low altitude tactical navigation areas

A low altitude tactical navigation area (LATN) is a large, clearly defined geographical area where the U.S. Air Force practices tactical navigation that typically ranges from the surface to 1,500 feet AGL [13]. These areas are not depicted on aeronautical charts. Current information concerning LATNs is available from local U.S. Air Force facilities. These areas are flown at or below 250 knots, when multiple aircraft are not flying the same ground track. MOA acrobatic type activity is not appropriate for a LATN area.

13.22.3 Local flying areas

Most military facilities develop local flying areas within which they can conduct routine, nonhazardous training activity [13]. These areas are normally developed in conjunction with local FAA controllers and airspace managers. They are developed so that they will not conflict with other airspace usage and are locally published. Although use of these areas is generally limited to assigned units, the airbase airspace managers will make them available to interested parties. These areas are not depicted on standard published charts or publications.

13.22.4 Aerial refueling routes

There are over 100 aerial refueling routes utilized by the military over the United States [13]. The majority of them are located at high altitudes. However, there are VFR helicopter refueling tracks at low altitudes that do affect operations at these lower altitudes. The information on the VFR refueling tracks is located in [21]. There are four types of aerial refueling: (1) tracks (2 to 400 miles long), (2) anchors (20 to 50 miles long holding pattern associated with a MOA or RA), (3) special anytime routes (e.g., emergency, military exercises), and (4) low altitude air refueling (LAARs) routes located below 3,000 feet AGL. Some VFR refueling routes are designed to be flown at or below 1,500 feet AGL. They are designed to permit aircraft flying the route to avoid charted, uncontrolled airports by three NM or 1,500 feet. The track is normally 50 to 100 NM long and normally four NM in width to either side of a centerline unless otherwise specified. Aerial refueling may be conducted within an SUA assigned altitude. This includes both low-altitude (helicopter and fixed wing) refueling as well as higher-altitude tracks.

13.22.5 Temporary special use airspace

The U.S. military and the FAA have the ability to create temporary military operations areas or temporary restricted areas to accommodate the specific needs of a particular military exercise [13]. This information is available via either the NOTice To AirMen (NOTAM) system or by direct contact with the FAA regional headquarters. Temporary military operating areas are published in the NOTAM publication. This publication may be purchased on a subscription basis from the Government Printing Office (GPO) in Washington, D.C. [22].

13.22.6 Cruise missile routes

Cruise missile operations are conducted on selected IFR military training routes [13]. They may be flown in excess of 250 knots and below 10,000 MSL. Cruise missiles may be accompanied by two chase aircraft. The chase aircraft must always maintain the ability to maneuver the missile out of the flight path of conflicting traffic. A high-altitude communications aircraft may be used in conjunction with the cruise missile. It maintains communications and radar contact with the appropriate ATC facility. Cruise missile operations are conducted in daylight hours under VFR conditions, with a flight visibility of at least five miles, maintaining 2,000 feet horizontal and 1,000 feet vertical separation from all clouds. Special charting on sectional charts designate Unmanned Aerospace Vehicle RouteS (UAVRS).

13.22.7 National security areas

A national security area (NSA; [Figure 13.10](#)) is an area where there is a requirement for increased security [13]. Pilots are requested to voluntarily avoid flying through the depicted NSA. When it is necessary to provide a greater level of security and safety, flights in NSAs may be temporarily prohibited under the provisions of the Federal Aviation Regulation Part 99.7 [15]. Since 11 September 2001, special security measures have been implemented within the United States. Pilots are advised to avoid the airspace above, or in proximity to, sites such as power plants, dams, refineries, industrial complexes, and military facilities.

13.23 Oceanic and offshore operations

As oceanic air traffic continues to grow and automation improvements are made, the FAA continues to pursue airspace changes to enhance efficiency and capacity. The Oceanic and Offshore Operations Program is a program where the FAA and industry is working to further define regulations and operations [22].



Figure 13.10 Example United States national security area.

13.24 Air defense identification zone

All aircraft entering U.S. airspace from points outside must provide identification prior to entry [13]. To facilitate early identification of all aircraft in the vicinity of the U.S. and international airspace boundaries, air defense identification zones (ADIZs) have been established. Generally for all flights entering an ADIZ the following are required: (1) a filed flight plan, (2) an operable two-way radio, and (3) an operable radar beacon transponder with an altitude reporting capability. An ADIZ is normally located "off shore" or along U.S. boundaries. After September 11, 2001, an ADIZ has been created over both Washington D.C. and over New York City.

13.25 Environmentally sensitive areas

There are areas of airspace within the United States that are considered environmentally sensitive [13]. The physical presence or noise associated with aircraft overflight may conflict with the purpose of these areas. Examples

include wilderness areas, national parks, areas with threatened and/or endangered species, religious areas, wildlife refuges, Native American areas, and primitive areas. Pilots are voluntarily requested to maintain a minimum altitude of 2,000 feet above the surface of the following: National Parks, National Monuments, seashores, lake shores, recreation areas, and National Scenic River Ways administered by the National Park Service (NPS), National Wildlife Refuges, Big Game Refuges, Game Ranges and Wildlife Ranges administered by the Fish and Wildlife Service (FWS), and Wilderness and Primitive Areas administered by the U.S. Forest Service (USFS).

13.26 Airdrops

Federal regulations prohibit airdrops (by parachute or other means) of persons, cargo, or objects from aircraft on lands administered by the National Park Service, the Fish and Wildlife Service, or the Bureau of Land Management without authorization from the respective agency [13]. Exceptions due to emergencies and other threats exist in the regulations.

13.27 Special federal aviation regulation

Federal statutes prohibit certain types of flight activity and/or provide altitude restrictions over designated national wildlife refuges, national parks, and national forests [13]. These special federal aviation regulation (SFAR) areas are represented on sectional charts.

13.28 Noise abatement procedures

U.S. civilian and Department of Defense airfields may have published noise abatement procedures within their Class C, D, or E airspace or within transition routes [13]. They may only be published by the local airport manager or noise abatement officer. Concentrated VFR traffic along these routes may result in increased midair potential. When operating out of an unfamiliar airport, it is recommended that aviators contact the airport manager to become familiar with procedures and restrictions. SUA and MTRs also may impose noise abatement procedures on their users. It is recommended that a DoD Flight Information Publication [18 or 21], or the using/scheduling agency, be consulted for specific information.

13.29 Air traffic control communications

Air traffic controllers monitor and direct traffic within a designated volume of airspace called a sector [23]. Each sector has a separate radio channel assignment for controllers to communicate with aircraft flying within that sector. As the amount of air traffic grows, the need for additional sectors and channel assignments also increases. The ICAO's present air-ground

communications system operates in a worldwide, very-high-frequency (VHF) band reserved for safety communications within the 118 to 137 megahertz (MHz) range. Within this range of frequencies, the FAA currently has 524 channels available for air traffic services. During the past four decades, FAA has primarily been able to meet the increased need for more channel capacity within this band by periodically reducing the space between channels (a process known as channel splitting). For example, in 1966, reducing the space between channels from 100 kHz to 50 kHz doubled the number of channels. The second channel split in 1977, from 50 kHz to 25 kHz, again doubled the number of channels available. Each time the FAA reduced this space, owners of aircraft needed to purchase new radios to receive the benefits of the increased number of channels. More recent changes, such as the 1995 ICAO adoption of the plan to subdivide the standard 25-kHz separation by three down to 8.33 kHz, were made to address frequency shortages in Europe. Well over half of the U.S. airline fleet's radios are incompatible with both spacings. FAA can use or assign its 524 channels several times around the country (as long as channel assignments are separated geographically to preclude frequency interference). Through channel reuse, FAA can make up to 14,000 channel assignments nationwide. Additional frequencies are used internationally. The Global Air Traffic Management (GATM) subdivides communications channels down to 17 KHz.

13.30 Summary and conclusions

The global airspace provides an excellent example of a system-of-systems (SoS) concept which has been successfully implemented and operated over many years. To successfully operate within this airspace, it is necessary for the aircraft operator to have a degree of understanding of the regulations and their structure. The regulations themselves prescribe the amount of training required by aircraft operators based on the type of flight activities within which they will participate. Regulations also cover operating standards and flight rules, airspace classes, airways, and routes. The certification requirements for the aircraft themselves are similarly based on the type of flight activities for which they are designed and constructed. A complete listing of the regulations would be quite extensive. Within this chapter we have briefly discussed the general operating and flight rules governing the operation of aircraft within national and international airspaces. Also introduced are the procedures to be followed in the initiation, administrative processing, issuance, and publication of rules, regulations, and other airspace control orders.

Acknowledgments

The author would like to thank the FAA for materials and illustrations provided for use within this publication.

References

1. De Laurentis, D.A., C.E. Dickerson, M. DiMario, P. Gartz, M. M. Jamshidi, S. Nahavandi, A.P. Sage, E.B. Sloane, and D.R. Walker, A case for an international consortium on system-of-systems engineering, *IEEE Systems Journal*, DOI 10.1109/JST.2007.904242, pp. 1–6, 2007.
2. DeLaurentis, D.A., Understanding transportation as a system-of-systems design, Problem, in *Proc. AIAA Aerosp. Sci. Meeting Exhibit*, 2005, Article AIAA-2005-123.
3. DeLaurentis, D.A. and R. K. Callaway, A system-of-systems perspective for future public policy, *Rev. Policy Res.*, 21(6): 829–837, 2006.
4. Nahavandi, S., Modeling of large complex system from system of systems perspective, presented at the IEEE SoSE Conf., San Antonio, TX, 2007.
5. Lopez, D., Lessons learned from the front lines of aerospace, presented at the IEEE Int. Conf. Syst. Syst. Eng., Los Angeles, CA, 2006.
6. Wikipedia. 2007. International Civil Aviation Organization. http://en.wikipedia.org/wiki/International_Civil_Aviation_Organization.
7. International Civil Aviation Organization. 2007. <http://www.icao.int>.
8. Air Transportation Association. 2007. <http://www.airlines.org>.
9. National Business Aviation Association, Inc. 2007. <http://www.nbaa.org>.
10. Wikipedia. 2007. Air traffic control. http://en.wikipedia.org/wiki/Air_traffic_control, 1 October 2007.
11. United States Ultralight Association. 2007. <http://www.usua.org>.
12. General Services Administration. 2007. Federal Acquisition Regulation (as amended). Department of Defense, Washington, DC.
13. Federal Aviation Administration. 2003. Interagency Airspace Coordination Guide. FAA, Washington DC.
14. Wikipedia. 2007. Airspace class. http://wikipedia.org/wiki/Airspace_classes.
15. U.S. Code of Federal Regulations. U.S. Government Printing Office, Washington, DC.
16. Wikipedia. 2007. Controlled airspace. http://en.wikipedia.org/wiki/Controlled_airspace.
17. Federal Aviation Administration. 2000. *Procedures for Handling Airspace Matters*, FAA Handbook 7400.2. U.S. Department of Transportation, Washington, DC..
18. U.S. Department of Defense. 2003. Flight Information Publication (FLIP), AP/1A. U.S. Department of Defense, Washington, DC.
19. U.S. Executive Order 10854. 27 November 1959. 24 *Federal Register* 9565, 3 CFR, 1959-1963 Comp., p. 389.
20. Federal Aviation Act of 1958 (as amended), Washington, DC, December 2003.
21. U.S. Department of Defense. 2003. Flight Information Publication (FLIP), AP/1B. U.S. Department of Defense, Washington, DC.
22. Federal Aviation Administration. 2007. <http://www.faa.gov>.
23. U. S. Government Accountability Office. 2002. National Airspace System: FAA's Approach to Its New Communications System Appears Prudent, but Challenges Remain. <http://www.gao.gov/htext/d02710.html>.
24. www.freqofnature.com/aviation/aviation_frequencies.html, 1 October 2007.

chapter fourteen

System of autonomous rovers and their applications

*Ferat Sahin, Ben Horan, Saeid Nahavandi,
Vikraman Raghavan, and Mo Jamshidi*

Contents

14.1	System of autonomous rovers	366
14.1.1	Stationary sensors and sensor networks.....	367
14.2	Haptically controlled base robot.....	367
14.2.1	Electrical and mechanical construction	368
14.2.2	Haptic control: the haptic gravitational field (HGF).....	369
14.2.3	Operation in the system of autonomous rovers	370
14.2.3.1	Communication schemes.....	370
14.2.3.2	Control schemes.....	371
14.3	Swarm robots.....	371
14.3.1	Mechanical construction and components.....	371
14.3.2	Navigation solution with GPS	372
14.3.2.1	The interface	372
14.3.2.2	Understanding the parameters.....	373
14.3.2.3	Calculating the heading angle.....	373
14.3.3	Sensor fusion	374
14.4	Application: robust threat monitoring	375
14.4.1	Introduction.....	375
14.4.2	The scenario.....	376
14.5	Swarm of micromodular robots.....	376
14.5.1	Electrical and mechanical construction	376
14.5.1.1	Power layer.....	376
14.5.1.2	Control layer	376
14.5.1.3	Ultrasonic layer	377
14.5.1.4	Infrared layer.....	377
14.5.1.5	Communication layer.....	377
14.5.1.6	GPS layer	377
14.5.2	Communication scheme	378
14.5.3	Swarm behavior: the ant colony-based swarm algorithm....	378

14.5.4 Application: mine detection.....	379
14.5.4.1 Mine hardware.....	379
14.5.4.2 Experimental results	380
14.6 Conclusion.....	382
References	383

14.1 System of autonomous rovers

In this chapter a system of autonomous rovers will be presented in the context of system of systems. In addition, a system of homogenous modular microrobots will be presented in the context of system of systems. The chapter starts with the introduction of the components and their roles in the system of autonomous rovers. Then, each system will be presented focusing on electrical, mechanical, and control characteristics and their capabilities in the system of autonomous rovers. Robust data aggregation and mine detection are then examined as applications of the system of autonomous rovers.

The system of autonomous rovers comprises four components: base robot, swarm robots, sensors and a threat. Figure 14.1 depicts the physical components of the system of rovers. The sensors represent a standalone system able to measure temperature and pressure and the ability to communicate with the Base Robot. The threat is spatially dynamic and it is assumed it is detectable by the chosen sensors.

In this particular scenario, the temperature local to a particular sensor or combination of sensors has been increased manually for ease and controllability of experimentation. Once the sensors have appropriately detected the

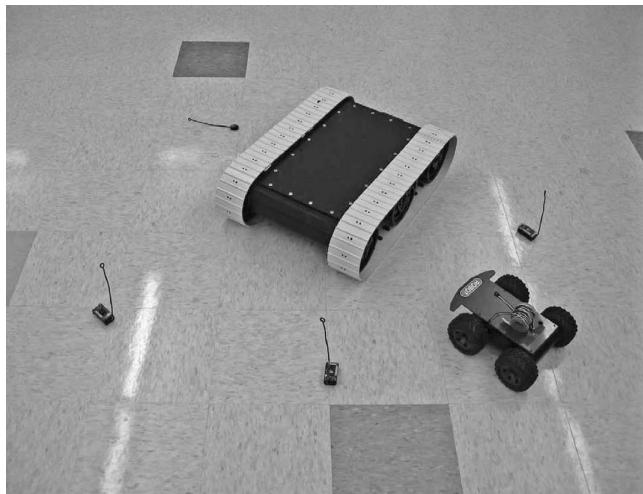


Figure 14.1 Components of the system of rovers: base robot, swarm robot, and sensors.

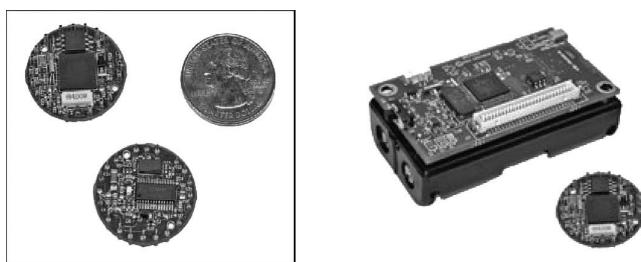


Figure 14.2 Stationary and mobile sensor platforms currently in use/development.

threat (temperature change), the base robot is informed of the event and subsequent location. The base robot then informs the swarm robot of the location of the threat. Upon receiving the threat information, the swarm robots navigate to the target location and use onboard sensory systems to validate the information obtained by the static sensors. Finally, the swarm rovers' sensor readings are communicated to the base robot for decision making based on its robust data aggregation algorithm. The base robot, swarm robot, and microrobots are discussed in the following sections. Sensor units are discussed here briefly, since we focus on autonomous rovers in this chapter.

14.1.1 *Stationary sensors and sensor networks*

For the stationary sensor platforms, we are currently using Crossbow's sensor motes (Mica 2 and Mica2Dot) that are equipped with processor-radio board and a multisensor board, shown in Figure 14.2. The processor-radio board consists of a 433-MHz multichannel transceiver and a low-power Atmel Atmega-128L 4-Mhz processor with 128 KB program flash memory. The multisensor board has the following sensors: temperature, humidity, barometric pressure, ambient light, 2-axis accelerometer (ADXL202), and a GPS receiver.

14.2 *Haptically controlled base robot*

The haptically teleoperated base robot plays an important role in the presented system of systems (SoS). The base robot provides onboard computational power, an advanced suite of sensors, and tracked locomotion for all-terrain navigation. Given the mechanical and load capabilities of the base robot, this system provides the capability for initial deployment of the sensor nodes to desired locations in order to appropriately monitor the target environment. The base robot also provides a communication link between each swarm robot, as well as providing a communication medium between the swarm robot and sensor network systems. The base robot utilizes its onboard processing power to transfer communication and commands between the swarm robot and sensor network systems. In the context of system of

autonomous rovers and their applications, the base robot can be considered as a semistatic base station, which is responsible for autonomous handling of information between the swarm robots and sensor networks. This is of course, based on the assumption that the sensor nodes have previously been placed appropriately in the target environment. Facilitating intuitive navigation and deployment of sensor nodes, a human-in-the-loop approach was adopted. Intuitive haptic control methodologies [1] and application-specific augmentation have been developed [1,2] in order to improve teleoperator performance in the navigation to and deployment of sensor nodes.

14.2.1 Electrical and mechanical construction

The mobile platform developed in this work is an open-architecture articulated-track rover. The requirements of specific sensory, computation, communication and all-terrain capabilities necessitated the development of a custom platform for implementation in this system. The developed prototype is presented in Figure 14.3(a). The robot's tracked locomotion offers superior all-terrain capabilities, including the ability to traverse sand, mud, and shrubs and to climb rocks and stairs. The locomotion of this platform was chosen specifically to facilitate traversal of challenging real-world terrain. The haptic attributes of this teleoperation system therefore have the potential to improve the operator's control capabilities when attempting to navigate difficult real-world scenarios.

In order to facilitate task-relevant haptic augmentation, the robot is equipped with various sensory and control systems. The robot's onboard sensors include a GPS for absolute positioning in outdoor environments, wireless video for a view of the remote environment, ultrasonic range-finding for obstacle detection, 3-axis gyro for orientation, 3-axis accelerometer for motion capture, and encoders for monitoring the vehicle's velocity. The robot's computation is achieved through an on-board Windows-based laptop. This platform was designed specifically to meet the necessary requirements in this haptically teleoperated scenario. In order to reduce the required

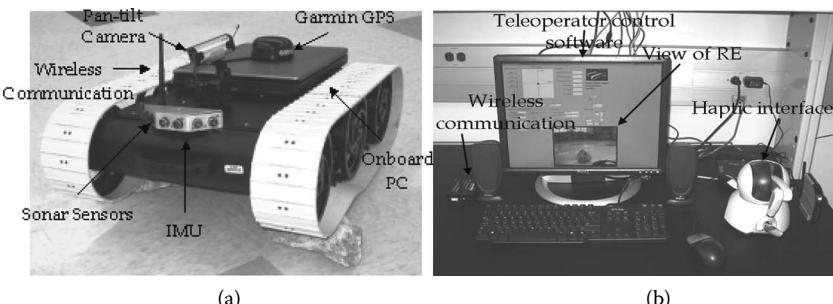


Figure 14.3 (a) Base robot and (b) haptic teleoperator control interface.

communication bandwidth between the mobile platform and teleoperator control station, the robot processes all of its sensory information locally. This reduces the amount of communicated data, sending only information directly pertaining to the appropriate haptic cues, thus contributing to improved real-time system responsiveness.

The teleoperator control station provides a medium for human-in-the-loop control of the remote robotic system. Many teleoperator interfaces currently used in real-world applications are controlled by a simple joystick-type device, while an onboard camera provides information from the remote environment. In order to implement the haptic human-robotic interaction, a commercial single-point haptic interface is utilized. This haptic device is a grounded, manipulator-style device offering 6-DOF motion input with 3-DOF force feedback. The implemented teleoperator control station is designed to facilitate bilateral haptic human-robot interaction in order to improve performance when navigating in a remote environment. The teleoperator can then receive application-specific information from the mobile robot using both the visual and haptic sensory modalities.

14.2.2 Haptic control: the haptic gravitational field (HGF)

Controllability is often as important as platform capabilities. The effectiveness of an immersive operator interface in providing the operator with the necessary mission-critical information can prove highly advantageous. Considering the scenario where the operator is required to command the base robot to a specific location in order to deploy the sensor nodes, the haptic gravitational field [2] is introduced in the aims of assisting the operator in such a task. As a basis for the HGF, the following assumptions are made:

1. The absolute location of the desired goal is known, including direction relative to the robot.
2. The environment is so unstructured that determination and evaluation of obstacles and safe navigational paths is not feasible by an autonomous robot, but better performed by the human operator.

In order to deploy the sensor nodes to the desired locations, the overall objective of the teleoperator is to safely navigate the remote mobile robot from a start location to the goal or target location. In order to travel to a known goal location, the HGF can utilize the robot's capabilities to provide haptic indication to the teleoperator of direction and distance to the desired location. The HGF is intended to provide the teleoperator with a force-based haptic indication of the current distance and direction to the goal location. This can prove extremely valuable to the teleoperator when the goal location is not clearly identifiable by visual information alone.

The HGF is demonstrated by [Figure 14.4](#), where x_r, y_r is the current position of the robot, and x_g, y_g the position of the desired sensor deployment

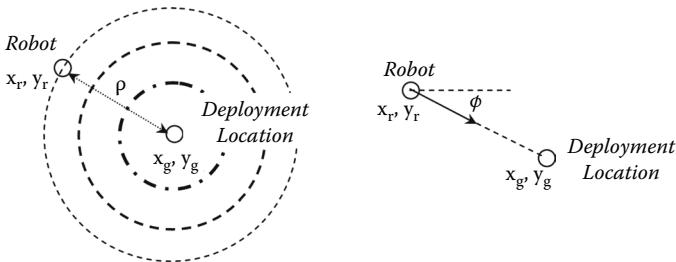


Figure 14.4 The haptic gravitational field (HGF) [2].

location, with respect to a world coordinate system. Given the current location of the rover and a known goal location, the magnitude of the haptic indicative force (ρ) resulting from the HGF is given by (14.1)

$$\rho = k_2 + k_3 \cdot \left(\sqrt{ (y_g - y_r)^2 + (x_g - x_r)^2 } \right)^{-1} \quad (14.1)$$

where k_2 is the minimum possible haptic force, and k_3 is a constant of proportionality relating to the distance to the goal location. The direction to the goal location ϕ is given by (14.2)

$$\phi = \arctan((y_g - y_r) / (x_g - x_r)) \quad (14.2)$$

Given the current location of the robot (x_r, y_r) and a known goal position (x_g, y_g) , the HGF results in the haptic force vector acting across an implemented haptic control surface [1]. The use of the HGF (including directionality) provides the teleoperator with a method to haptically determine the direction and distance to a goal location, when visual information may not be sufficient on its own. Furthermore, the HGF allows the teleoperator to concentrate their visual sense on local navigation of the challenging terrain, while inferring global navigation objectives from the haptic information.

14.2.3 Operation in the system of autonomous rovers

Having successfully placed the sensor nodes in the desired locations, the base robot assumes a semistatic and autonomous role within the system of autonomous rovers. This role involves receiving and processing information from the active sensor network and providing commands to the robotic swarm.

14.2.3.1 Communication schemes

The communication between sensor network and the base robot follows the star configuration (Figure 14.5). The sensor networks also utilize a centralized controller physically present within the base robot's onboard PC. Given this

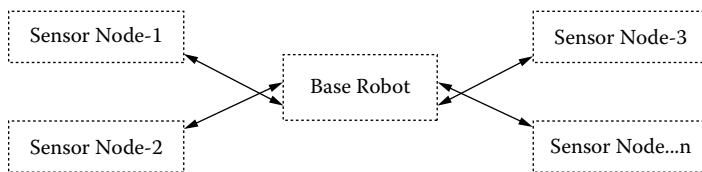


Figure 14.5 Base robot and sensor network communication—star configuration.

configuration, as the number of sensor nodes increases, the required communication channels only increase proportionally with the number of nodes.

14.2.3.2 Control schemes

As mentioned before, the base robot receives information regarding the detection of a possible threat from the sensor network. Given the prior knowledge of the location of any deployed sensor (x_n, y_n), the base robot receives sensory information pertaining to the monitored environment. If it is deduced that a threat is present, then the nature and location (GPS coordinates) of the threat location are communicated to the robotic swarm.

14.3 Swarm robots

The swarm robotic system comprises a set of identical robots which are relatively smaller in size, inexpensive, and hence less capable than the base robot. Each robot in the swarm has the same physical and functional characteristics. This section explains about the mechanical and functional characteristics of each of the robots in the swarm.

14.3.1 Mechanical construction and components

The robots use an off-the-shelf robotic mechanical platform. Four DC motors are coupled to the four wheels of the robot with appropriate gears. Control to the motors is provided with the help of an H-bridge servo controller. A field-programmable gate array (FPGA) board with a multiprocessor architecture is used as a controller. FPGA provides superior open architecture over conventional microcontrollers, which is desired for typical laboratory and field research.

A GPS receiver capable of sending data in National Marine Electronics Association (NMEA) 0183 format is part of the design. The receiver provides the navigation information to the robot. A magnetic compass that is used to complement the GPS information is also included in the design. Moreover, additional sensors for navigation and surveillance are included in the design. A radio modem is connected to provide connectivity to the rest of the system. A battery with appropriate A-h rating is essentially part of the system. The hardware block diagram is shown in [Figure 14.6](#).

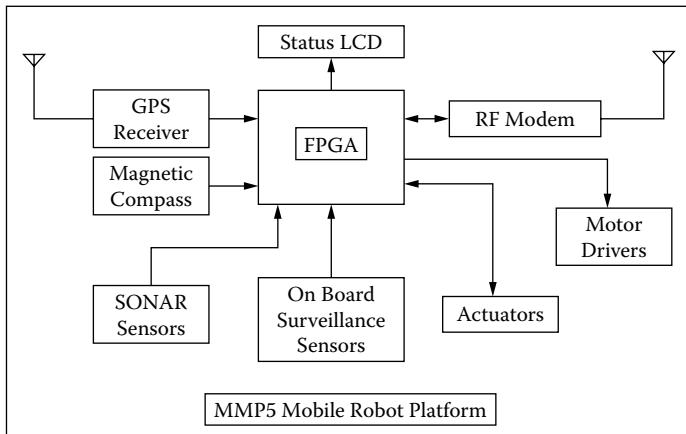


Figure 14.6 Architecture of the robot.

14.3.2 Navigation solution with GPS

14.3.2.1 The interface

GPS receivers calculate their position using the trilateration techniques. The basic position information includes the latitude, longitude, and their respective hemispheres. Based on those parameters, the receiver calculates dynamic parameters such as speed, magnetic orientation, etc. The receiver then formats all the parameters into sentences defined by the NMEA 0183 standard. The NMEA 0183 standard is an industrial standard for communication between marine electronics devices, and it widely used by GPS receivers. The most basic and powerful sentence of the NMEA 0183 standard is GPRMC, the recommended minimum specific GPS/transit data. The GPRMC sentence provides all the basic navigation information such as latitude, longitude, corresponding hemispheres, UTC fix, course over ground, speed over ground, and the mode in which the device works. Out of all the information, latitude, longitude, and corresponding hemispheres are the original data calculated by the receiver. Course over ground and speed over ground are calculated based on the rate of change of the read latitude and longitude data. For autonomous navigation the data necessarily needed are latitude, longitude, and hemisphere information. Also, we use course over ground information in our navigation algorithm.

The receivers communicate with a microcontroller/computer through a standard RS232 serial port or serial port-based USB interface. The sentences are transmitted in ASCII format through the interface. The NMEA 0183 specification suggests that the communication may be established at 4800 bps, 8 data bits, and no parity, which are the default connection parameters for all the receivers.

14.3.2.2 Understanding the parameters

With appropriate hardware and software interfaces the latitude, longitude, their hemispheres, and magnetic orientation data are parsed from the GPRMC sentence and converted from ASCII format to corresponding absolute number values. The data those are parsed and converted to in numeric format define the present location of the receiver/robot on the earth and its magnetic orientation. The destination latitude and longitude information is obtained from the user through an appropriate interface. The latitude and longitude distribution on the globe is basically two dimensional with four quadrants. The latitude and longitude intersect at right angle only at the intersection of equator and prime meridian. However, we can still consider that the latitude and longitude intersect at right angles at every point on the Earth, based on the assumption that the world looks flat to normal eyes and not elliptical. Moreover, the navigation algorithm discussed is iterative, which lets the robot recalculate its path until it reaches its destination. The iterative mechanism nullifies the error that is generated by assuming that the latitude and longitude intersect at right angles all over the Earth.

14.3.2.3 Calculating the heading angle

The present location and the destination location of the robot are mapped on a latitude-longitude layout with their respective (latitude, longitude) coordinates. With basic coordinate geometry concepts and trigonometric principles, the angle at which the robot should head to reach the destination from where it is at that point of time is calculated. Let us assume that a robot is to navigate from location A with (0 N, 0 E) as its (latitude, longitude) to location B with (3 N, 6 E) as its (latitude, longitude). Assuming that the latitude and longitude intersects at right angles,

1. Present location A and destination location B are connected with a straight line segment.
2. Applying the coordinate geometry distance formula, Δx and Δy are calculated.
3. $\theta = \tan^{-1} (\Delta y / \Delta x)$ is calculated.

In the above example, both the present and destination locations are on the first quadrant. If the present coordinate and end coordinate are in any other quadrant, the theta is correspondingly level shifted. The convention specified by NEMA for magnetic orientation is to have magnetic north as 0 or 360 degrees, south as 180 degrees, east as 90 degrees, and west as 270 degrees. After calculating the heading angle θ desired, the robot is aligned to head in the desired angle that leads it to the destination location. The actual magnetic orientation θ information extracted from the GPRMC sentence is used to verify if the robot has aligned to the desired heading angle θ .

14.3.3 Sensor fusion

It may be interesting to note that the magnetic orientation to a GPS receiver is a dynamic parameter. Dynamic parameters are calculated from the change of the latitude and longitude values calculated by the receiver. This suggests that the position data of the receiver, at each instant of time, are stored in a first-in first-out (FIFO) buffer array. The location history information is used to calculate the dynamic parameters. Once the buffer gets filled up with information, the buffer is replaced with newer location information. If the speed of the receiver is slower than the rate at which the buffer is replaced, the dynamic behavior of the receiver cannot be captured. Thus, the receiver has to be in continuous motion at a speed greater than the minimum speed required by the receiver to compute the magnetic orientation information and other dynamic parameters. Typical minimum speed that can be captured is about two miles per hour (mph). Practical robotic applications demand slow speed of less than two mph during certain maneuvers. Some common situations when the speed drops below two mph are when

1. Encountering obstacles
2. Evaluating threat
3. Waiting for a command
4. Executing a command

In these instances there is a possibility that the receiver on the robot fails to keep track of the dynamic behavior of the robot, including the orientation information. Also, during the cold start of the robot the receiver would not be able to provide magnetic orientation data and other dynamic parameters to the robot. However, the magnetic orientation data is necessary information for autonomous navigation. This forces us to complement the GPS receiver with another device that is capable of determining the magnetic orientation of the robot even when the robot is stationary. The simplest solution would be to use a dual-axis magnetic field sensor-based compass that can report the magnetic orientation information according to the NMEA 0183 specification. It may be interesting to note that the geographic north is different from magnetic north. Normal magnetic compasses work based on the Earth's magnetic field, and they would read North Pole where the Earth's magnetic north is present. But, due to the differences in the flow of metals inside the Earth's core, the magnetic north has been continuously drifting. Hence, the compass reading is not accurate. However, the GPS receiver calculates the orientation of the robot based on the way the latitude and longitude change, and it calculates the orientation with absolute north pole as reference. The error caused by the magnetic compass is calculated from the difference between the compass and GPS receiver readings. The error is compensated from the

read magnetic orientation data. Thus GPS receiver and magnetic compass complement each other, forming a dynamic sensor fusion strategy.

14.4 Application: robust threat monitoring

14.4.1 Introduction

As eluded to in the preceding sections, this system was developed in order to perform threat monitoring of a target environment. Utilizing the distinct capabilities of the haptically controlled all-terrain base robot, the sensor network, and robotic swarms, the system of systems shown in Figure 14.7 aims to robustly monitor a target environment for potential threats. Sensors play the most fundamental and trivial role in any control system. Sensors basically measure a parameter of the system for further processing. The quantity and the quality of the measured data are then processed to understand the current state of the system. With the current state of the system known, there may be a need to take some corrective actions. This philosophy is fairly common in robotic systems.

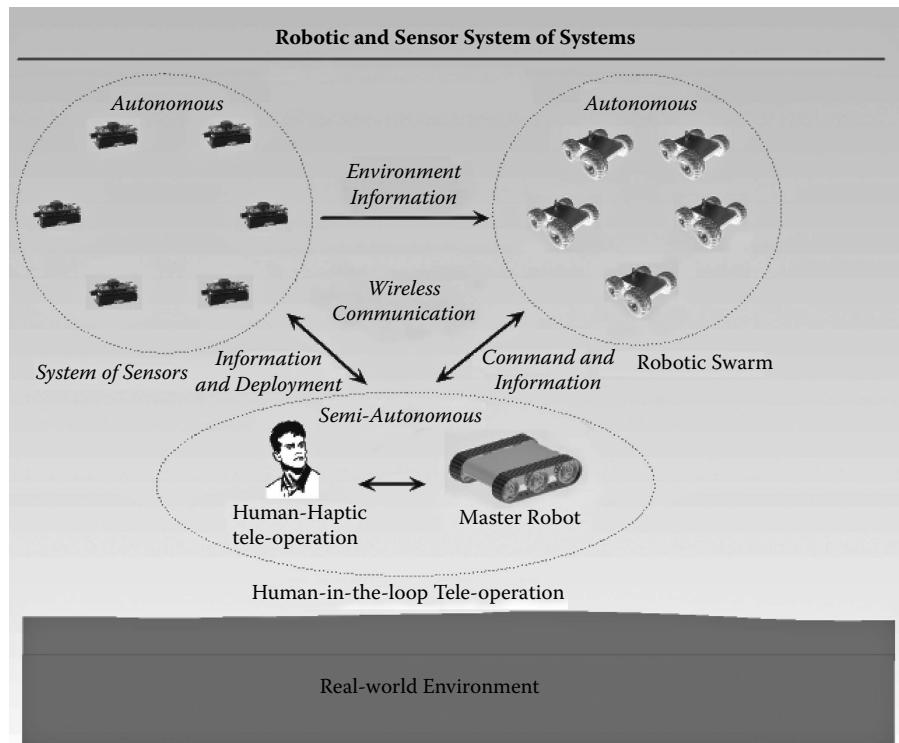


Figure 14.7 High-level system of systems architecture.

14.4.2 The scenario

To illustrate this philosophy in an SoS perspective, we introduce three individual systems that have already been explained in Sections 14.1, 14.2, and 14.3. A sensor network system is primarily used to find any exceptional condition that may prevail over the region of interest. The haptically controlled master robot acts as the central workhorse of the SoS, providing relatively exceptional physical strength, computational power, and communication node. Swarm of robots are characterized by their lower cost and relatively larger area of coverage [3,4].

14.5 Swarm of micromodular robots

In this section, we will present another set of robots in a similar application with a different hardware and software architecture. They are micromodular robots designed to study and emulate swarm intelligence techniques and applications [5,6,7]. These robots will be called GroundScouts throughout the section. In addition to the hardware and software components of the robots, the implementation of a robotic swarm as a system of systems and its application to mine detection problems will also be presented.

14.5.1 Electrical and mechanical construction

GroundScouts are cooperative autonomous robots designed to be both versatile and modular in hardware and software. The overall design was centered on modularity, creating a robot that could be easily altered to fit the conditions of almost any application. A picture of a GroundScout is shown below in [Figure 14.8](#).

For modularity, GroundScouts are divided into several independent layers. The following subsections explore each layer.

14.5.1.1 Power layer

The power layer contains the circuitry needed for power of all the layers, including the motor drivers for the locomotion layer. The batteries are stored on the locomotion layer between the wheels.

14.5.1.2 Control layer

The control layer is composed of a Phillips 80C552 as the main controller for the entire robot. It is connected to all of the other layers via a hardware bus that runs up the back and the sides of the robot. This creates a mechanical and electrical means of connecting different layers of the robot. Currently, we are working on the second generation of GroundScouts, which has ARM processor and micro operating system.

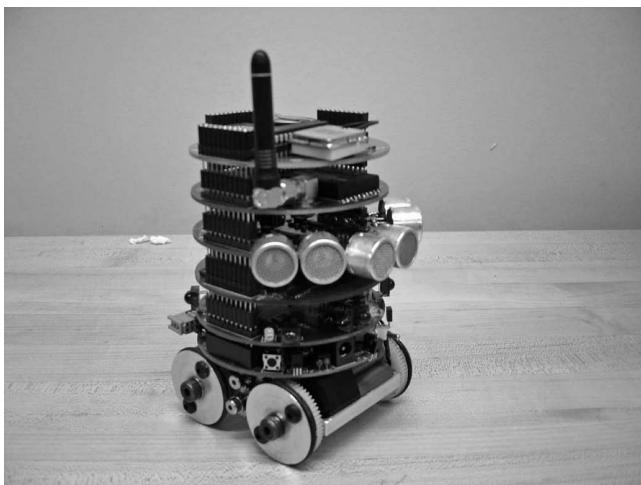


Figure 14.8 Front view of the GroundScouts.

14.5.1.3 Ultrasonic layer

The ultrasonic layer has three ultrasonic drivers, which can be arranged in two different configurations. In the first configuration, the sensors are 120 degrees offset from one another. In the second configuration, all three sensors are in the front of the robot with 60 degrees offset from one another.

14.5.1.4 Infrared layer

The infrared layer can be used for short-range communication from robot to robot along with trail following, meaning that the robot can be programmed to follow an IR signal.

14.5.1.5 Communication layer

The communication layer is composed of a PIC microcontroller and a wireless transceiver that is capable of transmitting serial data at ranges up to 300 feet. The modulation scheme that the transceiver uses is frequency-shift keying, meaning that all of the users are sharing the same medium. This created the need for an applied medium access control (MAC) protocol that was developed and described later.

14.5.1.6 GPS layer

The GPS layer was created to allow the robots to be sent off on autonomous missions and give them a way to get back to the master station by navigating based on GPS coordinates.

14.5.2 Communication scheme

Many different MAC protocols were studied in an attempt to find the one that was suitable for the robotic swarms. The protocols analyzed included frequency division multiple access (FDMA), code division multiple access (CDMA), time division multiple access (TDMA), and polling. The FDMA separates the users in the frequency domain. This is not suitable with the current hardware, since the transmit frequency of the transceivers cannot be changed. The CDMA gives each user a unique code. The message will only make sense to the user that has the code that the message was modulated with. The current hardware does not have the capability to implement a CDMA network, since the user has no control over the modulation and demodulation of the signal. Thus, we concluded that the TDMA was the most applicable method, since it is easy to implement a “collision-free” protocol and is suitable for the available hardware.

After implementing this protocol and analyzing it carefully, we discovered some inefficiencies in this type of network for robotic swarms. For example, the number of users on the network was fixed, creating a maximum number of users, and also creating unused slots if all of the users were not present. Thus, there was a need to develop an adaptive protocol that allowed the number of slots to change in accordance with the number of users on the network. This is referred to as adaptive TDMA [8–11].

The protocol works by creating a time slot at the beginning of each frame where users can request a transmit slot. The master grants each user a transmit slot. All of the other users on the network hear this and increment their transmit slot by one, creating a gap for the new user to enter. This also goes the other way. If no message is sent in a time slot, then the rest of the users on the network decide that the time slot is no longer in use, and they close it. The only contention is in the requesting time slot. This is handled by having the users generate a random number and wait that many frames before requesting another slot. Using the adaptive TDMA, swarms can organize their communication medium based on the number of robots in the swarm. By adaptively controlling the time slots, robots in a swarm can fully utilize the available transmission time. Next we discuss the swarm algorithm tested on the GroundScouts.

14.5.3 Swarm behavior: the ant colony-based swarm algorithm

The swarm behavior to be tested is ant colony behavior. The algorithm is designed based on short- and long-range recruitment behaviors of the ants when they are seeking food. The details of the algorithm can be found in [12,13]. The algorithm is applied to the mine-detection problem [6,7]. The resulting algorithm is presented in the diagram shown in [Figure 14.9](#). The boxes represent the three different states that the robot can be in, while the diamonds represent the transitions that occur.

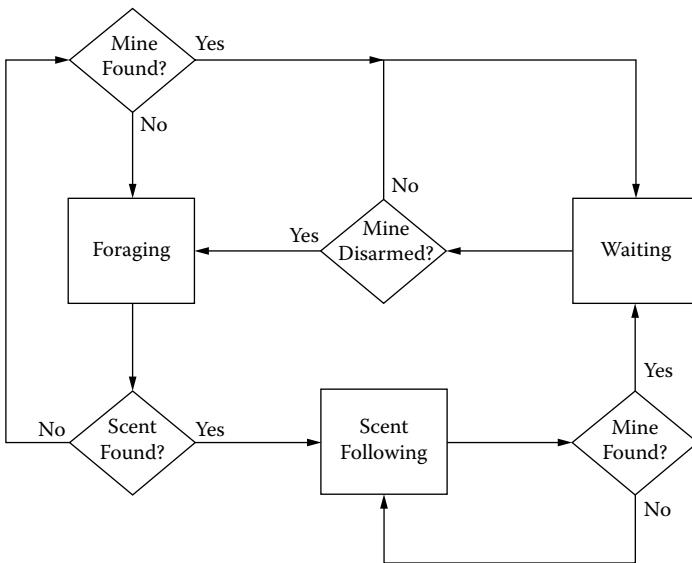


Figure 14.9 Flow chart of the implemented swarm algorithm.

The algorithm requires that multiple robots (four in our experiments) be present around a mine for the mine to be disarmed [8]. This creates the need for two different messages to be sent from robot to robot. One message indicates to other robots that the mine was found, to mimic a scent. Another message tells the other robots that the robot timed out. These two messages allow the other robots to know exactly how many robots are surrounding the mine.

One thing to note is that, when a robot times out, it turns around completely and travels fifteen feet before it begins to forage again. This gets the robot far enough away so that it does not instantly go back to the mine it was just at. Also, the timeout count is reset when another robot arrives at the mine. For the experiments, five robots are used to disarm two mines. A mine must have four robots surrounding it in order to be disarmed. The robots will start in between the mines at the same location. The mines are placed far enough apart such that the communication radius of a robot at mine 1 and a robot at mine 2 does not overlap.

14.5.4 Application: mine detection

As mentioned, the ant colony-based swarm algorithm is applied to a mine-detection problem. First we will present the systems used as mines, and then we will present the results of the experiments run in a basketball court.

14.5.4.1 Mine hardware

The mines are composed of a beacon that constantly transmits an infrared signal that is modulated at 38 KHz in all directions. A picture of the beacon

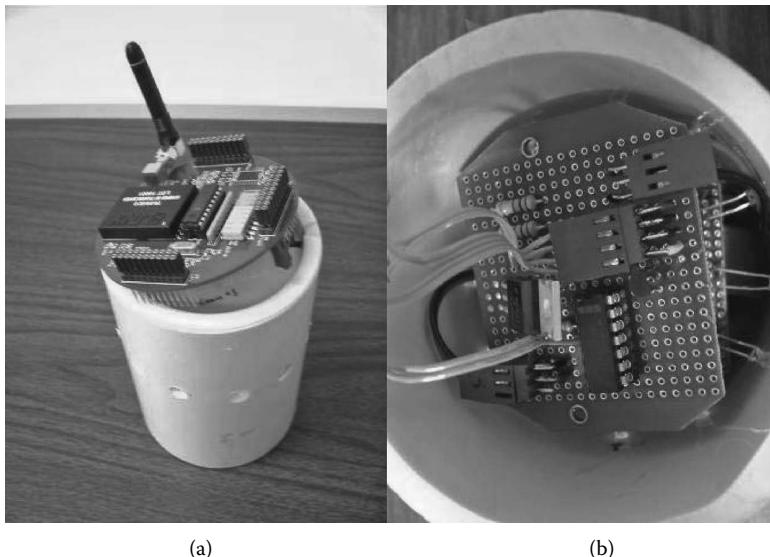


Figure 14.10 (a) Outside view and (b) inside view.

is shown in Figure 14.10. The signal can be sensed by the robot within a 7-foot radius. The robots are constantly searching for this signal. As soon as the signal is detected, the robot knows that it is close to a mine.

Directionality is found by viewing the five sensors that surround the robot. Early attempts were made to find the sensor with the best signal and assume that the mine is in that direction. This proved to be difficult, since the sensors are somewhat omnidirectional, creating a number of sensors having a good signal and making it difficult to really pinpoint the exact direction of the mine. It was concluded that finding the direction could be simplified by looking for the two sensors that have the worst signal. The robot could then move in the opposite direction, which would be directly toward the mine. The mine is disarmed using the GroundScout's communication module. A communication board was placed on the top of the mine as shown in Figure 14.10(a). When enough robots are surrounding the mine to disarm it, a message is sent by the command center to the communication board, telling it to disarm the mine. The PIC on the communication board will then toggle a pin that will turn the mine off. The robots will then shift back into the foraging state, since signals from the mine will not be available after the mine is disarmed.

14.5.4.2 Experimental results

The experiments were performed in a gymnasium so that the robots had plenty of room to work with. A picture of the starting point of the experiment is shown in Figure 14.11.



Figure 14.11 Starting point of the experimental setup.

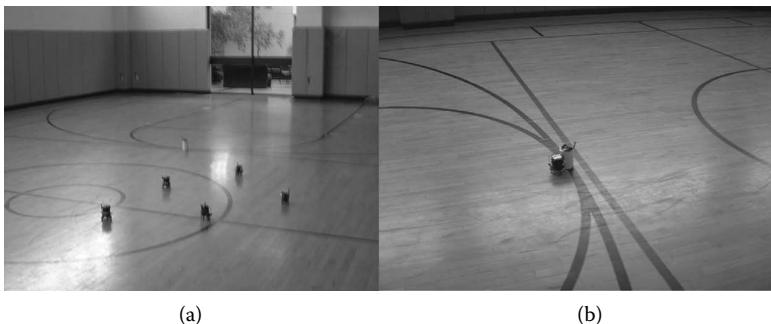


Figure 14.12 (a) Robots in the foraging state, (b) robot at the mine.

The robots are turned on one at a time, and each is allowed to move about 3 feet before the next robot is turned on. At the start of the algorithm, the robots are foraging. This is shown in Figure 14.12(a). It clearly shows the robots randomly searching for mines. The robots near the top of the figure are beginning to find the first mine. The algorithm used to find the mines using the infrared sensors will bring the robots toward the mine. The robot will then move until the back infrared sensors have no signal and the ultrasonic sensors are picking up an object that is within 6 inches. Figure 14.12(b) shows a picture of a robot at the mine.

As soon as the robot reaches the mine, it will begin sending out the recruitment signal to other robots. Since the implementation of the internal coordinate system neglects slippage, over time the robot's internal coordinates will begin to become off center. As a robot at a mine sends out the recruitment signal, other robots that are within the physical distance may not hear this

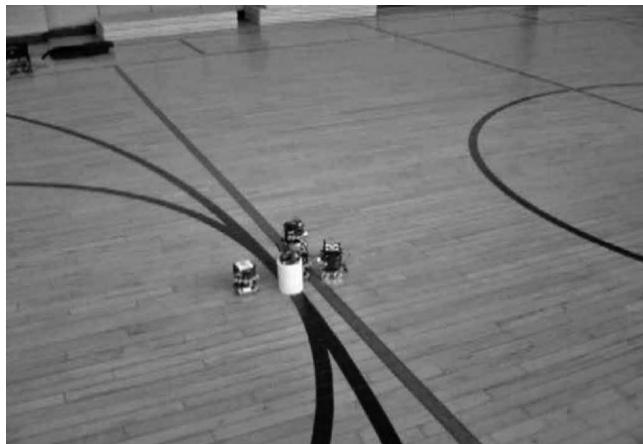


Figure 14.13 Robots disarming the mine and leaving.

signal because, according to the coordinate system, they are outside of *listening* range. Another problem is that sometimes a robot would hear the signal, but would go to the wrong location, because it is where the robot thinks the mine is.

As soon as four robots surround the mine, the mine can be turned off. The robots decide that a mine is turned off by checking their front infrared sensor. If no signal is detected, then the robots conclude that the mine has been disarmed; they then instantly switch into the foraging stage, which incorporates obstacle avoidance. This is shown in Figure 14.13.

This section presented a real-time implementation of an ant colony-based system of swarming robots to the mine-detection problem. In addition, an adaptive communication network that maximizes the efficiency of the network has also been implemented. It was shown that the algorithm can be effectively implemented with very few problems.

The robotic swarm, as a system of systems, shows fewer problems than heterogeneous systems exhibit. First of all, there is no compatibility issue among the system components, since all the robots have same or similar components. The robots also have the same software architecture and communication medium.

14.6 Conclusion

This chapter describes two examples of system of systems using autonomous rovers. In the first example, the systems were heterogeneous in terms of their hardware and software, which definitely requires the theory of system of systems. In the second example, a swarm of robots is examined as a system of systems. In this example, the hardware components were similar or the same for each robot. The advantage of this system of systems was the

common software architecture and known system hardware, even though some of the robots could have different hardware components because of the modularity. In both cases, a communication medium is crucial so that the components of the system of systems can communicate properly and operate together. By developing a communication medium, system-of-systems concepts can be studied. The common communication medium can be achieved in hardware and/or software architecture of the communication modules of each system in the SoS.

References

1. Horan, B., S. Nahavandi, D. Creighton, and E. Tunstel. 2007. Fuzzy haptic augmentation for telerobotic stair climbing. In the *IEEE International Conference on Systems, Man and Cybernetics*. Montreal, QC, pp. 2437–2442.
2. Horan, B., D. Creighton, S. Nahavandi, and M. Jamshidi. 2007. Bilateral haptic teleoperation of an articulated track mobile robot. In *IEEE International Conference on System of Systems Engineering, 2007. SoSE '07. San Antonio, TX*, pp. 1–8.
3. Azarnoush, H., B. Horan, P. Sridhar, A. Madni, and M. Jamshidi. 2006. Towards optimization of a real-world robotic sensor system-of-systems. In *Proceedings of World Automation Congress*. Budapest, pp. 1–8.
4. Sahin, F., P. Sridhar, B. Horan, V. Raghavan, and M. Jamshidi. 2007. System of systems approach to threat detection and integration of heterogeneous independently operable systems. In *IEEE International Conference on Systems, Man and Cybernetics*, Montreal, QC, pp. 1376–1381.
5. Sahin, F., W. Walter, and K. Kreigbaum. 2003. Design, build and test of modular mobile micro robots. In *Proceedings of IMECE'03 2003 ASME International Mechanical Engineering Congress and Exposition*, Washington, D.C.
6. Chapman, E. and F. Sahin. 2004. Application of swarm intelligence to the mine detection problem. In *Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics*, vol. 6, pp. 5429–5434.
7. Opp, W. J. and F. Sahin. 2004. An artificial immune system approach to mobile sensor networks and mine detection. In *Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 947–952.
8. Burr, A. G., T. C. Tozer, and S. J. Baines. 1994. Capacity of an adaptive TDMA cellular system: comparison with conventional access schemes. In *5th IEEE Conference on Personal, Indoor, and Mobile Radio Communications*, vol. 1, pp. 242–246.
9. Kanzaki, A., T. Uemukai, T. Hara, and S. Nishio. 2003. Dynamic TDMA slot assignment in ad hoc networks. In *International Conference on Advanced Information Networking and Applications*, pp. 330–335.
10. Papadimitriou, G. and A. Pomportsis. 1999. Self-adaptive TDMA protocols for WDM star networks: A learning-automato-based approach. *IEEE Photonics Technology Letters* 11:1322–1324.
11. Stevens, D. and M. Ammar. 1990. Evaluation of slot allocation strategies for TDMA protocols in packet radio networks. In *IEEE Military Communications Conference, 1990*, vol. 2, pp. 835–839.
12. Kumar, V., F. Sahin, and E. Cole. 2004. Ant colony optimization based swarms: implementation for the mine detection application. In *Proceedings of 2004 IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, pp. 716–721.

13. Kumar, V. and F. Sahin. 2003. Cognitive maps in swarm robots for the mine detection application. In *Proceedings of 2003 IEEE International Conference on Systems, Man and Cybernetics*, vol. 4, pp. 3364–3369.

chapter fifteen

Space applications of system of systems

Dale S. Caffall and James Bret Michael

Contents

15.1	Introduction	385
15.1.1	Manned space exploration	386
15.1.1.1	Orion crew capsule	386
15.1.1.2	Aries launch vehicles	387
15.1.1.3	Lunar Surface Access Module.....	387
15.1.1.4	Earth Departure Stage.....	387
15.1.2	Unmanned space exploration	388
15.1.2.1	Mars Science Laboratory.....	388
15.1.2.2	Juno	388
15.1.2.3	James Webb Space Telescope.....	389
15.2	System of systems	389
15.2.1	Manned space exploration	389
15.2.1.1	System-of-system architecture.....	390
15.2.1.2	Component framework	392
15.2.1.3	Contract interfaces	393
15.2.1.4	System-of-systems specification	394
15.2.2	Unmanned space exploration	396
15.2.2.1	Distributed systems.....	396
15.2.2.2	Dependability	399
15.3	Conclusion.....	401

15.1 Introduction

Space . . . the Final Frontier. These are the voyages of the starship Enterprise. Its five-year mission: to explore strange new worlds, to seek out new life and new civilizations, to boldly go where no man has gone before.

How many of us heard these words and dreamed of standing alongside *Star Trek's* Captain James T. Kirk and First Officer Mr. Spock to explore new worlds? Interesting enough, Chief Engineer Commander Scott performs countless technological miracles as he saves his beloved USS *Enterprise* and crew from certain destruction; however, Scotty never mentions the miracle of the *Enterprise* as a wonderful system of systems. Could it be that future engineers solved all the problems of system of systems? Could it be that Scotty never encountered problems with the system of systems comprising the *Enterprise*, freeing him up to devote his time on advancing the interplanetary communications, warp engine, phaser, transporter, deflector shield, and other systems of the *Enterprise*? Apparently not, at least as evidenced by, for instance, the emergent behavior of the *Enterprise* as it interacts with a (fictitious) legacy unmanned scientific probe named *Voyager 6*, whose onboard systems have been reconfigured and augmented by an alien race. In this chapter, we discuss the application of the system-of-systems technology for the development of a space foundation that, indeed, may lead to the Federation and star ships.

While many people almost exclusively think of space exploration as the Shuttle Orbiter that has captured worldwide interest for over twenty-five years, space exploration is multifaceted. The National Aeronautics and Space Administration (NASA) has a extraordinary history of space exploration to include the Rovers that continue to explore Mars, space telescopes that can see into deep space, a human-inhabitable space station that orbits the Earth, unmanned spacecraft that explore the Sun and the planets in our solar system, and the unforgettable lunar exploration that began on July 20, 1969, as Commander Neil Armstrong stepped onto the Moon's surface.

So, what is the relationship of space exploration with a system-of-systems concept? Well, let us first consider space exploration missions that are planned for the next couple of decades.

15.1.1 *Manned space exploration*

For future manned exploration, NASA established the Constellation Program that will eventually lead to manned spaceflights to Mars. Prior to reaching for the goal of manned exploration of Mars, NASA identified intermediate goals of docking with the International Space Station and returning to the Moon for exploration and establishing human-habitable lunar laboratories. To support the achievement of these goals, the Constellation Program will produce a number of products including the Orion crew capsule, Aries launch vehicles, the Lunar Surface Access Module, and the Earth Departure Stage.

15.1.1.1 *Orion crew capsule*

The Orion crew capsule which will consist of two main components: a crew module, which will be similar to the Apollo Command Module and capable

of holding four to six astronauts, and a cylindrical service module that will contain the primary propulsion systems and consumable supplies.

The Orion crew capsule will be propelled into space by the Aries I rocket, which will consist of a solid rocket booster that is connected at its upper end to a new liquid-fueled second stage powered by an enhanced J-2X rocket engine. The Aries I rocket will have the capability to lift more than 55,000 lbs to low Earth orbit.

15.1.1.2 Aries launch vehicles

The Aries V rocket will transport the Lunar Surface Access Module and the Earth Departure Stage into space. The Aries V rocket will have the capability to carry nearly 290,000 and 144,000 lbs to low Earth and lunar orbit, respectively.

15.1.1.3 Lunar Surface Access Module

The Lunar Surface Access Module will transport astronauts to the lunar surface. It consists of two main components: an ascent stage, which houses the four-person crew, and a descent stage, which has the landing legs, the majority of the crew consumables, and scientific equipment.

15.1.1.4 Earth Departure Stage

The Earth Departure Stage will be the main propulsion system that will send the Orion crew capsule that is coupled with the Lunar Surface Access Module from low Earth orbit to the Moon. The Earth Departure Stage is propelled by a J-2X main engine fueled with liquid oxygen and liquid hydrogen.

Once in orbit, the Orion crew capsule will dock with the orbiting Earth Departure Stage carrying the Lunar Surface Access Module. Once mated with the crew capsule, the departure stage fires its engine to achieve escape velocity, and the new lunar vessel begins its journey to the Moon. The Earth Departure Stage is then jettisoned, leaving the crew module and Lunar Surface Access Module mated. Upon achieving lunar orbit, the four astronauts will transfer to the lunar module and descend to the Moon's surface. The crew module remains in lunar orbit until the astronauts depart from the Moon in the lunar vessel, rendezvous with the crew module in orbit, and return to Earth.

Let us consider the system of systems that the Constellation Program is developing. There will be the major systems as described above: Orion crew capsule, Aries I rocket, Aries V rocket, Lunar Surface Access Module, and the Earth Departure Stage. To complete this system of systems, there will be a ground control system at the Kennedy Space Center launch site as well as mission control at Johnson Space Center. Additionally, there will be the International Space Station as well as lunar outposts. For the Mars mission, there will be the communications network that will reach from Earth to the satellites that orbit Mars.

15.1.2 Unmanned space exploration

For unmanned space exploration, NASA has a multitude of programs that will support deep space exploration. Rather than discussing all of the programs, we will consider three major programs that are representative of all the unmanned space flights: Mars Science Laboratory, Juno, and the James Webb Space Telescope.

15.1.2.1 Mars Science Laboratory

The Mars Science Laboratory is a rover scheduled to launch in September 2009 and perform a precision landing on Mars in the summer of 2010. This rover will be three times as heavy and twice the width of the Mars Exploration Rovers that landed in 2004. The Mars Science Laboratory will have higher clearance and greater mobility than any previous rover sent to Mars, traveling a distance of five to twenty kilometers from its landing site. It will carry more advanced scientific instruments than any other mission to Mars. While NASA will develop many of these instruments, the international community will provide a number of instruments—complicating the integration challenges within this system of systems.

After reaching the surface of the red planet, Mars Science Laboratory will have a primary mission time of one Martian year (approximately 687 Earth solar days) as it explores with greater range than any previous Mars rover. The Mars Science Laboratory will analyze dozens of samples scooped up from the soil and drilled powders from rocks. It will investigate the past or present ability of Mars to support life.

15.1.2.2 Juno

Juno is a NASA mission to Jupiter and is scheduled to launch in 2013. At arrival at Jupiter in 2018, the spacecraft will perform an orbit insertion burn to slow the spacecraft enough to allow capture into Jupiter orbit. The spacecraft will then orbit Jupiter about its poles with an eleven-day orbital period. Its mission will conclude in 2019, after completing thirty-two orbits around Jupiter.

The spacecraft will be placed in a polar orbit in order to study the planet's composition, gravity field, magnetic field, and polar magnetosphere. Once Juno enters into its orbit around Jupiter, infrared and microwave instruments will begin to measure the thermal radiation emanating from deep within Jupiter's dense atmosphere. These observations will complement previous studies of the planet's composition by assessing the abundance and distribution of water and, therefore, oxygen.

Juno will also be searching for evidence to help scientists understand how Jupiter formed. Specific questions that relate to the origin of planets in our solar system include the determination of whether Jupiter has a rocky core, the amount of water present within the deep atmosphere, and how the mass

is distributed within the planet. Juno will also study Jupiter's deep winds (which can reach wind velocities of 600 km/h).

15.1.2.3 *James Webb Space Telescope*

The James Webb Space Telescope is a large, infrared-optimized space telescope, scheduled for launch in 2013. It will find the first galaxies that formed in the early Universe, connecting the Big Bang to our own Milky Way Galaxy. It will peer through dusty clouds to see stars forming planetary systems, connecting the Milky Way to our own Solar System. NASA will design the James Webb Space Telescope to work primarily in the infrared range of the electromagnetic spectrum, with some capability in the visible range (0.6 to 27 μm in wavelength).

The James Webb Space Telescope is an international collaboration between NASA, the European Space Agency, and the Canadian Space Agency. It will have a large mirror that will be 6.5 m in diameter and a sunshield the size of a tennis court. Both the mirror and sunshade will not fit onto the rocket fully open, so both will fold up and open only once it is in outer space. The James Webb Space Telescope will reside in an orbit about 1.5 million km from Earth.

There will be four science instruments on the James Webb Space Telescope: a near-infrared (IR) camera, a near-IR multi-object spectrograph, a mid-IR instrument, and a tunable filter imager.

15.2 *System of systems*

With this background, let us examine potential system-of-systems issues in manned and unmanned space exploration.

15.2.1 *Manned space exploration*

The system-of-systems issues will appear in the different phases of flight. In the launch and ascent phase, our concerns will be primarily the system of systems that includes Orion crew capsule, Aries rockets, ground control station at the Kennedy Space Center, and the mission control station at the Johnson Space Center. After entering low Earth orbit and under guidance from mission control at Johnson Space Center, the Orion crew capsule will connect with the International Space Station to provide supplies. When the mission calls for going to the Moon, the Orion crew capsule will connect with the Earth Departure Stage that contains the Lunar Surface Access Module and depart for the Moon under guidance from mission control. In lunar orbit, the Lunar Surface Access Module will descend to the Moon and return after the astronauts complete their lunar missions. The Orion crew capsule will then return to Earth under guidance from mission control.

Given that there may be many system-of-systems issues for the manned flight, we will consider system-of-system architecture, component framework, components, contract interfaces, specifications, distributed systems, and dependability.

15.2.1.1 *System-of-system architecture*

System engineers can mistakenly believe that “stick-and-circle” and other cartoon-like diagrams fill the bill for documenting the architecture of a system. While this is a questionable approach for a single system, it is an atrocious approach for a system of systems. Before we continue with the discussion, let us define architecture and design.

- *Architecture:* The collection of logical and physical views, constraints, and decisions that define the external properties of a system and provide a shared understanding of the system design to the development team and the intended user of the system
- *Design:* The details of planned implementation which are defined, structured, and constrained by the architecture

In these definitions, there is an implied responsibility of the system architect to collaborate with the development team and the intended system user to document the architecture. While numerous tools, artifacts, and methodologies exist on the market to support the system architect, the content and usefulness of the architecture will be dependent on the skill of the architect to select the appropriate tools, artifacts, and methodologies that bring about a shared understanding of the system design. One can argue that the shared understanding of the system design should be the primary purpose of the specification of the architecture.

Critical interactions within system software can increase as the complexity of highly interconnected systems increases. In complex systems of systems, these possible combinations are practically limitless. System “unravelings” have an intelligence of their own as they expose hidden connections, neutralize redundancies, and exploit chance circumstances for which no system engineer might plan. A software fault at runtime in one module of the system software may coincide with the software fault of an entirely different module of the system software. This unforeseeable combination can cause cascading failures within the system of systems.

As architects make an architectural decision upon which all future design decisions will be influenced, the architecture becomes irreversible at that point; that is, an architecture that is hard to change is, for all intents and purposes, irreversible. The role of the architect is to find ways to eliminate irreversibility in designs. The architect should ensure that today’s decision does not limit the flexibility of design decisions tomorrow.

For future manned space exploration, we must consider a system-of-systems architecture that allows systems and components to dynamically enter

and exit the system of systems without disruption to the system of systems, that is, a reconfigurable plug-and-play system.

As the Aries rocket launches, the Ground Control passes the command and control over to Mission Control; however, the Orion crew capsule and the Aries rocket have decision-making mechanisms. The Guidance, Navigation, and Control (GN&C) component in the Orion crew capsule provides flight control, guidance, navigation, sensor data processing, and flight crew display. Mission Control monitors the rocket following launch and can command a mission abort that would hurl the Orion crew capsule out of harm's way.

The GN&C component will consist of two operational modes: auto and manual (control stick steering). In the automatic mode, the primary avionics software system will allow the onboard computers to fly the rocket—the flight crew simply selects the various operational sequences. The flight crew may control the rocket in the control stick steering mode using hand controls such as the rotational hand controller, translational hand controller, speed brake/thrust controller, and rudder pedals.

During launch and ascent, most of the GN&C commands will be directed to gimbal Aries' main engines and solid rocket boosters to maintain thrust vector control through the rocket's center of gravity at a time when the amount of consumables is changing rapidly. In addition, the GN&C will control throttling for maximum aerodynamic loading of the rocket during ascent and maintain an acceleration of no greater than 3 g during the ascent phase. To circularize the orbit and perform on- and de-orbit maneuvers, the GN&C will command the orbital maneuvering system engines. At external stage separation, on orbit, and during portions of entry, GN&C will control commands to the reaction control system. In atmospheric flight, GN&C will control Orion's aerodynamic flight control surfaces.

One of the major participants in bringing about the desired emergent behaviors while suppressing the undesired emergent behaviors in the system of systems will be the system-of-systems architect. In creating the architectures for manned space exploration systems, the architects should consider three aspects of the desired behavior: (1) what we want the system of systems to do, (2) what we do not want the system of systems to do, and (3) what we want the system of systems to do under adverse conditions. While the first consideration is oftentimes the foremost focus of architects, it is imperative that the system-of-systems architect for manned space exploration consider all three aspects. Undesired emergent behaviors that appear as the rocket ascents into low Earth orbit could have devastating consequences such as the sudden loss of control over the rocket's flight into space—a tumbling rocket that may not have the ability to abort.

Given all the automated as well as human decisions in space flight, the services provided and consumed must be deliberately defined. There will be a number of active participants in the command and control of the rocket, so the system-of-systems architect must identify service providers, service consumers, and mutually exclusive business rules to ensure the appropriate

command and control commands are provided by the correct components and consumed by the correct components.

15.2.1.2 Component framework

We believe that a best practice for evolving a system of systems is to compose the system through the integration of components. For this discussion, we define a component as a unit of composition with contractually specified interfaces and explicit context dependencies.

Component-based engineering is the design and construction of a system by integrating software components, interfaces, contracts, and a component framework. The concept is to employ portions of legacy software that contain the desired functionality as expressed in the software architecture, shape the legacy software into units of composition, develop new components as required, and integrate the components into a component framework to realize the desired system functionality.

The foundation for developing and designing a system using component-based engineering is the component framework. It is the “glue” that binds the components to do the desired work that is specified in the architecture, in addition to serving as a guide and orchestration mechanism for system integration and interoperability. The component framework depicts where a component resides and its interface to other software.

The role of a component framework at the control level is that of enforcer: components are compelled to behave in accordance with the methods controlled by the framework, such as the inclusion of temporal invariants to enforce system timing and sequencing constraints. To increase the capability of a system of systems, we will add other components to the framework that provide additional functionality. To increase the precision of a system of systems, we could replace the current components with new components that contain more precise algorithms.

In developing the architectural views with respect to components, we might consider the following properties of components:

- Provide services through well-specified interfaces
- Encapsulate state and behavior so that neither is visible to the component framework
- Rely on the component framework to initialize and communicate with other components

Recall the major components of the system of systems in the manned space exploration: Orion crew capsule, Aries I rocket, Aries V rocket, Lunar Surface Access Module, Earth Departure Stage, Ground Control at the Kennedy Space Center launch, Mission Control at Johnson Space Center, International Space Station, lunar outposts, and the communications network that will reach from Earth to the satellites that orbit Mars.

Each of these components has responsibilities that are independent of the other components, yet each component is a provider and consumer of services. In our component framework for manned space exploration, we want both the minimum set of interfaces to realize the provided and consumed services and the ability to replace a component in the framework without disrupting operations of the other components.

15.2.1.3 Contract interfaces

Tight coupling tends to make component maintenance and reuse much more difficult, because a change in one component automatically necessitates changes in others. Similarly, tight coupling makes extra work when an application has to adapt to changing business requirements, because each modification to one application may force developers to make changes in other connected applications.

We argue that a system of systems should employ the concepts of design by contract in the specification and design of its interfaces. Design by contract is a formalized way of writing comments to incorporate specification information into the software to express the requirements for the component. The contracts describe the assumptions the engineer makes about the system under development. The idea behind design by contract is that software entities have obligations to other entities based upon formalized rules between them. We create a functional specification (i.e., contract) for each component in the system whether it is salvaged code from legacy software or developed as new code. Thus, the execution of the software is the interaction between the component framework and the various components as bound by these contracts.

For example, if an engineer assumes that a given variable will never receive a null or negative input, then the developer should include this information in the contract. Additionally, if an engineer writes a piece of code that is always supposed to return a value greater than 100, he should add this information in a contract so the developers working with the other parts of the application know what to expect. These contracts describe requirements such as:

- Conditions that must be satisfied before a method is invoked
- Results that need to be produced after a method executes
- Assertions that a method must satisfy at specific points of its execution

A contract can specify three types of constraints in the interface. A contract specifies the constraints that the component will preserve—an invariant. The contract specifies the constraints upon the component framework (i.e., a precondition). Additionally, the contact specifies the constraints upon a component with respect to what it returns to the component framework with respect to the input to the component operation (i.e., a postcondition).

15.2.1.4 System-of-systems specification

The tendency in specifications is to document the “thou shalts” of specific system functions, design to the “thou shalts” with modifications to accommodate the development, and field a system that little resembles the collective “thou shalts” and, more importantly, contains limited user utility.

Capturing the desired system-of-systems behavior in the traditional natural language documents is a complex issue, given that the legacy systems in the system of systems have a combination of existing known and unknown system behaviors. Typically, the system-of-systems specification is reduced to a table of information exchange requirements that define the messaging that passes from one system to another.

As a result of a system-of-systems development strategy of interconnecting systems, the fielded system of systems frequently demonstrates undesired system behaviors under operational conditions. Consequently, the user is frequently irritated at the undesirable distributed-system behaviors that a system of systems may exhibit, such as halted processes without recovery and disparate versions of same-source data. Additionally, the system of systems may require dependability considerations, yet system architects and system engineers can experience a significant degree of difficulty in assessing the dependability of a system of systems.

The development community continues to fail in capturing and achieving the desired behavior of the system. Additionally, the development community continues to invest a significant amount of the program budget in system rework to correct deficiencies in specifications. Unfortunately, we seem to accept these facts and are resigned to repeat the errors of our predecessors. We continue on with the same bad practices of the past. As we have heard time and again, lunacy is defined as doing the same thing over and over while expecting a different result.

One tool for specifying and implementing the desired system behavior is formal methods. A system engineer can use formal methods in the definition, validation, and verification of system specifications. Additionally, one can implement the formal specifications with formalisms in the software. Formal methods can complement traditional techniques such as testing and can help developers improve the degree of trustworthiness in the acquisition of space systems.

Conventional software development methods may not be suitable for the development of dependable, safety-critical systems. In safety-critical systems, system faults could prove fatal to human life or lead to loss of valuable physical assets. With the use of formal methods, developers can analyze formalized statements and the associated impacts in a repeatable manner. Formal methods help engineers test a significant number of test cases and support analysis that can be checked by verified model checkers. In operational software, the use of formal methods can significantly enhance the ability to catch and handle runtime errors.

We propose the use of assertions in the development of formal specifications. Assertions can help system engineers find defects in specifications and designs earlier than they would otherwise find errors and greatly reduce the incidence of mistakes in interpreting and implementing correct requirements and designs. Additionally, the development and verification of formal specifications can support the development of error-handling specifications to appropriately manage runtime errors and logic breaks.

The use of assertions can significantly reduce the errors introduced in specifying system behavior. Assertions can considerably increase the level of clarity in the assumptions and responsibilities of system behavior, and reveal errors such as logic omissions and conflicting logic-statements. Assertions can catch common interface faults (e.g., processing out-of-range or illegal inputs) by precisely asserting the legal interface values for variables passed in through an interface.

In the specification of the component framework, components, and interfaces, we recommend that the control and coordination features be specified in the component framework and the method of computation be specified in the component. We want to specify the contract between the interface and the component framework in the interface. Thus, the interface is specified as an independent entity rather than a portion of either the component framework or the component.

In the specification of the component framework, engineers must specify more than the desired functionality of the component. Included in the specification should be (1) the required response time of a component's computation that supports the desired behavior of the system, (2) the required precision of the result to ensure the matching of precision between the component framework and a component, (3) the required throughput of the data streams to ensure that data loss does not occur as a result of a throughput mismatch, (4) required protocol and formatting to ensure the matching of data and fields during data transfer, (5) legal values for inputs and outputs, and (6) data dictionary to include the specification of the units of measure in the component framework.

In the specification of the component, one must specify more than the required input parameter from the component framework. Included in the specification should be (1) time to complete a computation, (2) the required precision of the input to ensure that matching of precision between the component framework and a component, (3) memory requirements to ensure sufficient memory is designed into the software architecture, and (4) data dictionary to include the specification of the units of measure in the component framework.

Since we used examples from the manned space exploration in the architecture discussion to include the component framework, components, contract interfaces, and specification, we will use examples from unmanned space exploration for the distributed system and dependability discussions. We should keep in mind that the system-of-systems principles previously

discussed for the development of the system of systems for unmanned space exploration systems and the discussion of distributed systems and dependability for unmanned space exploration are applicable to the manned space exploration.

15.2.2 *Unmanned space exploration*

The system-of-systems issues will appear in the different phases of flight. We can divide the unmanned space exploration into five major phases:

1. Launch and ascent
2. Travel-to-space exploration objective
3. Preparation for operations
4. Landing (for the Mars missions)
5. Data collection

The components in the system of systems for unmanned space exploration will be the payload capsule that contains the rovers or space observation platforms, Aries rockets, Ground Control, Mission Control, rovers, space observation platforms, and the space communications network.

In the launch and ascent phase, our concerns will be primarily the system of systems that includes the payload capsule, Aries rockets, Ground Control, and the Mission Control. After reaching space, the rocket will set course for its space objective. Upon reaching its space objective, space observation platforms will prepare for mission execution. For Mars rovers, the payload capsule will prepare for descent onto the surface of Mars.

While tremendously talented engineers can realize the functionality and dependability of the individual components, engineers must consider the principles previously described to construct the unmanned space exploration system of systems. To ensure correct and dependable operations, engineers must address the properties of distributed systems.

15.2.2.1 *Distributed systems*

It can be easy to connect a number of computers together with a given means of communications; however, it is significantly harder to cause the software in that gang of interconnected computers to perform and behave as desired. Leslie Lamport offered the following observation as a result of a continuing problem in a distributed system: "A distributed system is one in which the failure of a computer you didn't even know existed can render your own computer unusable." Lamport's observation describes the tendency within government acquisition organizations to equate connecting together a group of computers as effectively engineering a distributed system. Before we define what we require in a distributed system for a control element, we should define precisely what we mean by a distributed system.

For this discussion, we define a *distributed system of systems* as a system that has multiple processors that are connected by a communications structure. We will not include any desired characteristics of a distributed system of systems in the definition, given that the properties of an operational system of systems may include undesired behaviors; however, the system is a distributed system of systems nonetheless.

Key characteristics of an unmanned space exploration system of systems might include the following: (1) a distributed network, (2) an operational environment that includes Earth, space, and Mars, (3) management of concurrent rover activities, (4) automated decision making regarding the execution of experiments or rover movement, (5) stringent requirements for high levels of dependability of the systems' rovers that are on a distant planet with no local technical support.

We should consider the ability to detect failures in the distributed system to satisfy the seven dependability properties listed in Section 15.2.2.2. A crashed rover or a crashed Mars satellite is not easily distinguishable by Mission Control without careful design of a failure-detection solution in the distributed system. Without a failure-detection solution, Mission Control might wait forever for a message to arrive from a failed or lost rover. We should consider the incorporation of error-detection and error-handling methods to handle rover and satellite failures. For example, we could employ error-correction coding techniques to correct a designed number of transmission errors in a message. We could employ a time-out scheme that allows a rover to wait for a specified time before resending a message to Mission Control. We could turn to redundancy of either software or hardware to increase the opportunities for successful operations. For such exceptions, engineers should develop an appropriate error-handling technique to overcome a failed component.

Another problem among distributed components is access to the critical section of a distributed system. For the unmanned space exploration system of systems, we will consider the critical section to be the space observation platform and the rovers. Recall that a component can fulfill a single tasking request from internal controls or external components at any given time. The problem is one of mutual exclusion. The solution to the mutual exclusion problem must satisfy the properties of safety (i.e., two processes cannot have simultaneous access to the critical section), liveness (i.e., every request to access the critical section should be granted eventually), and fairness (i.e., requests to access the critical section should be granted in the order that these requests are made).

For safety reasons, it is not desirable to have the situation in which multiple control elements are attempting to realign a sensor on a space observation platform or direct rover movement on Mars.

For liveness reasons, it is desirable to have a controlling element eventually access the critical section as requested. Given that a sensor has redi-

rected its field of regard to satisfy the request of an external component, it is not desirable to break that access until the original request is fulfilled.

For fairness reasons, it is desirable to grant requests in the order that the control elements generate the requests to access the critical section. Each control element has an essential mission to perform, so each request must be granted fairly to achieve maximum mission success.

There are numerous algorithms that provide mutual exclusion in the critical section. There are pros and cons for each algorithm. For this discussion, we will consider a centralized algorithm, a distributed algorithm, and a token-ring algorithm.

In the centralized algorithm, each control element would request access to the critical section from the coordinator. If the coordinator determines that access is available, then the requesting control element would be granted access to the critical section. If another control element currently has access to the critical section, then the coordinator cannot grant access to the requesting control element; however, the coordinator queues the request for when the occupying control element vacates the critical section. This centralized algorithm satisfies the safety, liveness, and fairness properties. Only one control element can gain access to the critical section at any given time. All requests will eventually be fulfilled. Requests are satisfied in the order each request is made. The downside to this algorithm is that the coordinator becomes a single point of failure in the system of systems. If the coordinator crashes, then operations may go down.

In the token-ring algorithm, a token is passed from control element to control element in a prescribed pattern of token passing. This token must be held by a control element to access the critical section. If a control element receives a token, it checks to see whether it desires to enter the critical section. If so, the control element accesses the critical section and performs its work.

After it leaves the critical section, the control element passes the token on to another control element. The token continues to circulate in the prescribed pattern until a control element desires to access the critical section. The distributed algorithm satisfies the safety and liveness properties; however, it does not satisfy the fairness property as previously defined. Only one control element can gain access to the critical section at any given time. All requests will eventually be fulfilled. Requests are satisfied as the token becomes available. Every control element will have access to the token during every token-passing circuit on the network. Access is fairly granted but requests are not granted with respect to the time the request is made. The downside of this algorithm is that the detection of a lost token on the network is difficult to distinguish from the situation in which the token is being held by any given control element on the network. Additionally, if a control element fails or is lost, the prescribed pattern of token passing is interrupted.

In the distributed algorithm, a control element constructs a message that indicates the name of the critical section that it wants to access. This message is sent to all other control elements. When another control element receives

the access-request message, it will perform one of the following three actions based upon its current state:

1. If the control element is not in the requested critical section and does not want to enter the requested critical section, it will send an "OK" message back to the requesting control element.
2. If the control element is in the critical section, it will queue the request for access.
3. If the control element is not in the requested critical section but wants to enter the requested critical section, it will compare the timestamp on the requesting message to the timestamp on its own request message. The control element honors the message with the older timestamp. If the received message has an older timestamp, then the control element sends an "OK" message to the requesting control element. If the received message has a younger timestamp, then the control element queues the request message and waits for action on its access request.

The distributed algorithm can satisfy the safety, liveness, and fairness properties for the unmanned space exploration system of systems. Only one control element can gain access to the critical section at any given time. All requests will eventually be fulfilled. Requests are satisfied in the order each request is made. The downside of this algorithm is that each control element on the network is a potential point of failure. That is, if a control element is lost or failed, it will not respond to access requests from other control elements. Consequently, this "silence" will be interpreted as a denial of access by control elements that have submitted access requests to the critical section. Additionally, every control element in the network is a potential bottleneck given the number of messages generated for each access request.

15.2.2.2 Dependability

In general, we do not have the luxury of beginning a space exploration system of systems development from scratch. We must work with the components that are in development and components that are in operational use. We cannot begin totally anew, so we must find other methods to apply to this common development situation for a system of systems.

Oftentimes, engineers focus intently on developing the desired functionality and fail to sufficiently address dependability. Before we embark on our discussion of dependability, we offer our definitions of a dependable system and a trustworthy system:

- A *dependable system* is one that provides the appropriate levels of correctness and robustness in accomplishing its mission while demonstrating the appropriate levels of availability, consistency, reliability, safety, and recoverability.

- A *trustworthy system* is one that provides the appropriate levels of correctness and robustness in accomplishing its mission while demonstrating the appropriate levels of availability, consistency, reliability, safety, and recoverability to the degree that justifies a user's confidence that the system will behave as expected.

With respect to dependable and trustworthy systems, we define the following properties in the context of a dependable system of systems for space exploration:

- *Availability*: The probability that a system is operating correctly and is ready to perform its desired functions
- *Consistency*: The property that invariants will always hold true in the system
- *Correctness*: A characteristic of a system that precisely exhibits predictable behavior at all times as defined by the system specifications
- *Reliability*: The property that a system can operate continuously without experiencing a failure
- *Robustness*: A characteristic of a system that is failure and fault tolerant
- *Safety*: The property of avoiding a catastrophic outcome given a system fails to operate correctly
- *Recoverability*: The ease for which a failed system can be restored to operational use

Other properties can be used to describe a dependable system; however, we selected the above seven properties, as these seven properties should be the minimum set of properties for a dependable system of systems for space exploration.

Engineers must find new development methods for producing a dependable system of systems that exhibits predictable behavior and fault tolerance during runtime. As suggested in numerous publications, our current development techniques fail to support system developers in producing a system of systems with predictable behavior. Almost exclusively, engineers rely on testing prior to fielding the completed product to assess system behavior. Rather than discovering system behavior at the end of the development phase, engineers might apply techniques that support the design and realization of desired system behavior from the earliest phases of concept development, architecture, and specification.

In space exploration, NASA must have confidence that the spacecraft will correctly launch, ascend into space, travel to its space objective, complete mission objectives, and return safely to Earth in the situation of manned space exploration regardless of the conditions in the operational environment. That is, space exploration system of systems must be trustworthy systems. Besides the impact of mission failure, public confidence in NASA's ability to

lead space exploration could erode. Certainly, the impact of mission failure is always of interest in Congress, which sets annual funding levels for NASA.

In space operations, the computations for launch, guidance, navigation, and control must be correct and robust; that is, the space exploration system of systems should demonstrate correctness in that it does the right thing all the time and it is available all the time to complete the space mission objectives. Additionally, the space exploration system of systems should demonstrate robustness in that it handles unexpected states in a manner that minimizes performance degradation, data corruption, and incorrect output.

In the consideration of dependable software in space exploration system of systems, engineers should consider the development of architecture, component frameworks, contract interfaces, specifications, and distributed system design. Our seven properties of a dependable system apply to each of these areas; however, each area has unique considerations that engineers should consider, as previously discussed.

Dependability is equally as important as functionality. Dependability is an independent entity and should not be considered in the trade space for functionality. A rocket that functions as designed but demonstrates a 50% malfunction and abort rate is not a useful rocket. Dependability must be built in from the earliest phases of the system-of-systems development life-cycle—not bolted on at the end of the development.

15.3 Conclusion

Evolving space exploration system of systems may well prove to be the most challenging endeavors in the history of humankind. However, if NASA is to successfully fly Orion to the International Space Station, return to the Moon, travel to Mars, and collect data from the far reaches of our solar system, the NASA engineers must solve the issues of developing a system of systems. With respect to evolving space exploration systems of systems, we have discussed architecture, component frameworks, components, contract interfaces, specifications, distributed systems, and dependability. There are other issues in evolving a system of systems, but solving the problems previously discussed will dramatically increase the mission success of space exploration.

chapter sixteen

Airport operations a system-of-systems approach

*Saeid Nahavandi, Doug Creighton,
Michael Johnstone, and Vu T. Le*

Contents

16.1	Introduction to airport operations	404
16.2	An introduction to system-of-systems concepts	404
16.3	Airport operations and analysis	405
16.3.1	Airport security	405
16.3.2	Passengers (PAX)	406
16.3.3	Passenger baggage	406
16.3.4	Air cargo	406
16.3.5	Additional security concerns	406
16.4	Airport operations as a system of systems	407
16.4.1	Operational independence	407
16.4.2	Managerial independence	407
16.4.3	Geographical distribution	407
16.4.4	Emergent behavior	407
16.5	Rapid model architecture for airport operations	407
16.5.1	System inputs	408
16.5.2	Data collection	409
16.5.3	Model development phases	412
16.5.4	Model validation and verification	412
16.5.5	Subsystem analysis	414
16.5.5.1	Data analysis tool	414
16.5.5.2	Input generator	416
16.5.5.3	Automated flow control	417
16.5.6	Overall system performance	418
16.6	Conclusion	418
16.7	Future challenges in airport operations	419
	References	419

16.1 Introduction to airport operations

Analysis of airport operations is commonly performed in isolation, sharing only simple information such as flight schedules. With the increased concern over security in our airports, a new approach is required whereby all aspects of the airport operations are considered. A system-of-systems methodology is proposed and demonstrated through example. This method provides the decision maker with an improved understanding of the implication of policy decisions, resource allocations, and infrastructure investment strategies, through the capture of emergent behaviors and interdependencies. New tools and methodologies are required for model development and analysis. These tools and methods are presented in this paper.

16.2 An introduction to system-of-systems concepts

What is a system of systems (SoS)? This question is made clearer by describing attributes that are frequently associated with systems of systems. A system of systems is composed of a collection of systems that are able to function individually or independently. There is no requirement between the separate systems to enable them to exist; however, when the individual systems are brought together, the gains from the system of systems is greater than the sum of the gains from the individual systems. This attribute rules out complex systems, such as the human body. Although it can be argued that removal of one airport business function might cause the entire sector to collapse, it could still function in some form. This differs from the human body where, for example, without the nervous system the circulatory system would cease to function.

The second attribute is managerial independence, inferring that the individual systems are managed separately, with differing individuals responsible for the strategic direction. This attribute effectively rules out separate divisions from the one company due to the overriding CEO of the company defining a global company strategic direction.

The third attribute used to describe a system of systems is that there may be differences in the geographic location of the individual systems. To visualize this attribute, contemplate a battlefield where information from ground troops about enemy formations can be relayed to support aircraft. Here information from diverse locations is compiled to give a better understanding of the entire battlefield.

The final common attribute is that systems of systems display emergent behavior. This attribute infers that functions or behaviors not evident in the individual systems can be performed by the system of systems as the system of systems adapts and evolves over time. Due to these unique attributes, there lies an opportunity to develop methods to study systems of systems and predict their eventual behavior as the individual systems change, causing changes in the larger system of systems.

16.3 Airport operations and analysis

The term airport operations initially triggers the view of passengers being transported by aircraft. Further thought would identify activities that directly or indirectly affect passenger operations, such as baggage handling systems (BHS), aircraft maintenance, and passenger security. In fact, airport operations consist of numerous aspects: concourses, runways, parking, airlines, cargo terminal operators, fuel depots, retail, cleaning, catering, and many interacting people including travelers, service providers, and visitors. The facilities are distributed and fall under multiple legal jurisdictions in regard to occupational health and safety, customs, quarantine, and security. For the airport to function, these numerous systems must work together.

Currently decision making in this domain space is focused on individual systems. The challenge of delivering improved nationwide air transportation security, while maintaining performance and continuing growth, demands a new approach. In addition, information flow and data management are a critical issue, where trust plays a key role in defining interactions of organizations.

Of great concern recently has been the need for rapid implementation of security measures in place at airports to protect passengers, staff, and aircraft [1]. While individual systems are being improved to increase security, such as the use of EDS (explosive detection systems) in baggage systems, a holistic view is required for this complex airport environment. By analyzing the operations as a system of systems, a more thorough understanding of what is currently going into airport security can be achieved, and the real and perceived levels of risk can be reduced. Also, a layered approach to security can be applied to the systems of systems to improve the safety of passengers, staff, and aircraft. The financial cost of implementing a security policy in an airport can be reduced through the study of the individual systems within the airport and their interactions.

System of systems methodologies are required to rapidly model, analyze, and optimize air transportation systems. In any critical real-world system there is and must be a compromise between increased risk and increased flexibility and productivity. By approaching such problem spaces from a system of systems perspective, we are in the best position to find the right balance.

16.3.1 Airport security

Airports have become a focus point of government security policy making since the September 11 attacks. Current research has focused on detection of contraband items before being loaded on aircraft [2] and the allocation of X-ray machines to passenger services, trading off between cost, effectiveness, and deterrence [3]. To gain an understanding of all the relevant security threats, they will be discussed next.

16.3.2 Passengers (PAX)

The first security threats we shall consider are those generated by passengers. Passengers pose three situations that are required to be dealt with, carry-on baggage, the individuals themselves, and checked baggage, which will be discussed in the following section. Passengers would now be well aware of the security measures in place to protect from any threats. Metal detectors are used to screen individuals, with more complex individual screening in development [4] and X-ray machines used to check carry-on luggage. Behind-the-scenes profiling can identify “high-risk” passengers, and a range of explosives detection technologies are being investigated for their effectiveness. Effectiveness can be achieved both through their perceived value (functioning as a deterrent) as well as the actual detection performance.

16.3.3 Passenger baggage

The second security threat is that of checked baggage. At passenger check-in, bags are labeled with a bar code to identify the passenger, bag, and flight details. Before the bags are delivered onto the plane, they must undergo screening by X-ray machine or EDS. The BHS uses a layered security approach in order to accommodate large volumes of bags. The first layer involves examination by high-capacity X-ray machines. Any bags not able to be cleared are escalated to a combination of human image inspection, human manual inspection, or a more detailed X-ray machine. The layered approach ensures suspect items are identified while maintaining a high throughput.

16.3.4 Air cargo

A third security threat deals with the air cargo operations of an airport. Freight for an airline is managed by a cargo terminal operator (CTO). CTOs receive freight from a variety of sources including freight forwarders (FF), postal services, and occasionally members of the public. The CTO utilizes the cargo space of a flight, consolidating freight where appropriate and balancing available space between passenger baggage and freight cargo. As air cargo provides a valuable addition to a country’s economy, it is essential it is kept secure while not imposing so strict a policy that the operations will grind to a halt.

16.3.5 Additional security concerns

Further security areas include, but are not limited to, physical access to sites within the airport, staffing, maintenance, and aircraft supplies.

16.4 Airport operations as a system of systems

Earlier in this chapter, attributes that describe a system of systems were described: operation and managerial independence, geographic isolation, and emergent behavior. These attributes can now be used to describe airport operations as a system of systems.

16.4.1 Operational independence

Passenger services are theoretically able to operate without freight and mail services and vice versa; however, the combination of both services provides a more favorable economic situation.

16.4.2 Managerial independence

Services within an airport are provided by many independent companies. In some cases the same service (e.g., CTO) is provided by several companies competing for an airlines business. This distribution of service providers is in line with the managerial independence for a system of systems.

16.4.3 Geographical distribution

While not separated by large distances, airport services are typically treated as isolated from the perspective of the analyst. Passenger screening and baggage services exist within the concourse, while cargo operations take place within the airport grounds.

16.4.4 Emergent behavior

Through studying airport operations as a system of systems, the emergent behavior is discovered. Through the sharing of common resources, more efficient operations can be achieved within the individual subsystems.

16.5 Rapid model architecture for airport operations

Simulation is recognized as an important tool in the study of airport operations [5]. Simulation modeling provides the tools to answer complex problems that are presented in real-world scenarios. The knowledge gained through the completion of a simulation study provides essential information for any long-term decision-making process.

If traditional methods were used to model complex environments such as an airport, the time to generate the model would be enormous. In the past, simulation models have been used as an add-on to the design process, simply verifying a particular design. By using rapid modeling techniques the time to generate and analyze the simulation models can be greatly reduced.

A result of this is that the simulation model is able to provide useful information during the design process, improving the design and reducing the possibility of effort being put into inappropriate designs. An overview of some techniques is given below.

16.5.1 System inputs

The major inputs to the system are shown in Figure 16.1. These inputs have been identified as key inputs to the system.

Flight schedules define the arrivals and departures of all flights to and from the airport. These schedules allow for the scheduling of other resources within the system.

Security levels define the minimum levels of security that must be achieved by the system. This may imply 100% checked baggage screening, random and continuous, or targeted screening through flight or customer profiling.

Screening machines from different vendors have varied performance levels and processing rates. The time taken by different models to capture and analyze images, as well as the machines' reject rates, are different. This information needs to be captured in order to develop accurate models.

The arrival rate for passengers and cargo needs to be captured during the data collection phase simulation project and used in the simulation model. From a modeling perspective, passenger, passenger baggage, and cargo loads can be estimated from the flight schedule. Taking into consideration variables such as daily, weekly, and annual cycles, destinations, fare types, or aircraft type, it is possibly to generate profiles to model the input variability. Current modeling methodologies assume that passenger and cargo loads are somewhat determined by the corresponding flight schedules. An SoS approach requires these to be treated as part of the modeling itself, rather than just an input to the model.

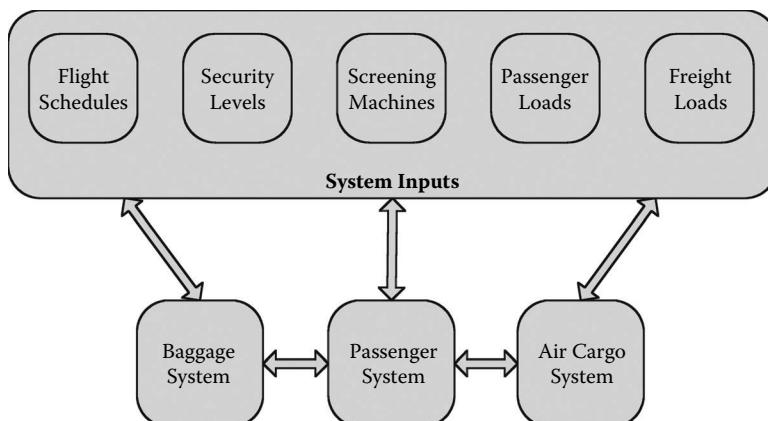


Figure 16.1 System inputs to an airport model.

16.5.2 Data collection

Models require accurate data before they will produce meaningful results. When modeling multiple systems, the task of data collection can become enormous due to the number of constraints, exceptions to standard procedure, and interdependencies. Compounding the challenges resulting from the sheer volumes of information is the fact that major business sectors are still predominantly paper based. Any reporting typically reports the average performance statistics and does not capture the process variability critical to understand the impact of changes to systems or procedures.

Therefore, automatic data collection techniques become highly desirable. Data can be sourced electronically from weighing machines, control systems, booking systems, and many more places. In some instances, like human operator decision process and execution times, automatic data collection is not available, and alternative methods are required. In these cases a simple interface to a software database is an extremely efficient way to collect the data. An interface is shown in Figure 16.2. Data are collected for various tasks as the buttons on the form are selected. As an operator performs a task, the corresponding button on the form is selected; this will allow subsequent buttons to be selected depending on the status of the task. All button sequences and timings are recorded.

Once the data has been captured and stored in the database, either through the software illustrated in Figure 16.2, or through an import from another electronic format, a visual user interface can be used to query the database. An example of a user interface to query the database is shown in Figure 16.3.

The interface allows for the selection of different tables and data ranges to specify the data for analysis. By selecting analysis types on the left of the interface, a series of graphs are generated for viewing on the right.

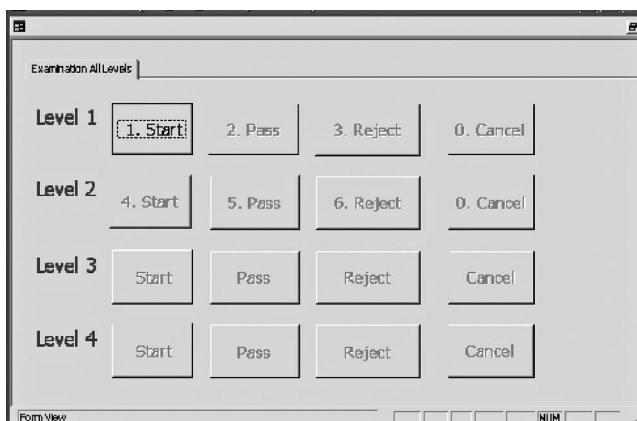


Figure 16.2 Software Interface to database.

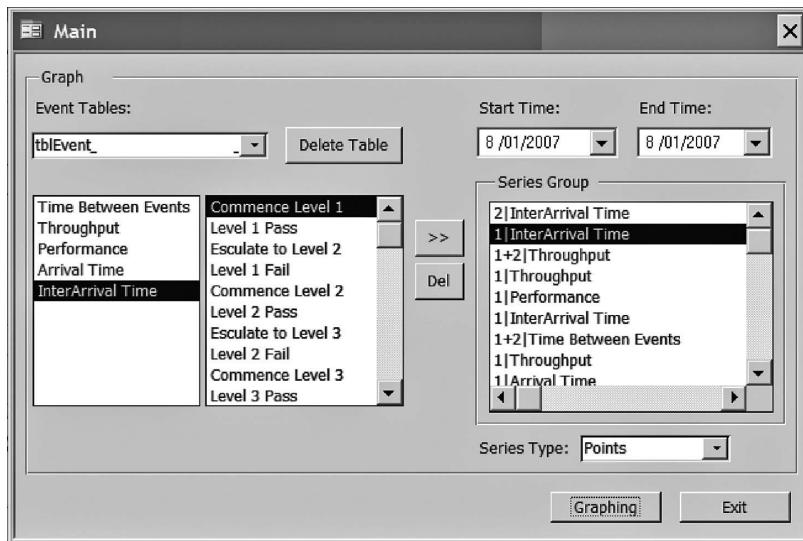


Figure 16.3 User interface for database query.

A combination of automated and graphical analysis techniques are required to facilitate the arduous task of data cleaning, to ensure the integrity of data, and analysis. The analysis is required for model input, model validation, and model analysis data sets.

When the user clicks [Graphing], the program will draw the data in a plotting tool that allows visual inspection of data. A typical graph is shown in [Figure 16.4](#).

The graphical view of the data provides a data extraction method for erroneous data. For instance, during the experimental test trial, at the beginning of the data collection process, the data obtained is not reliable. This is due to many factors including training the operator to use the software and hardware, changing the configuration of the system, determining a different area to monitor, or too few data items due to the time of the day. Each of these factors would affect the result from the collected data; therefore these data points need to be excluded from the input data before it can be statistically analyzed and inserted into the simulation system.

A user can toggle between “zoom” and “selection” mode while viewing the graph. When a user chooses selection mode, points are able to be selected, as shown in [Figure 16.5](#).

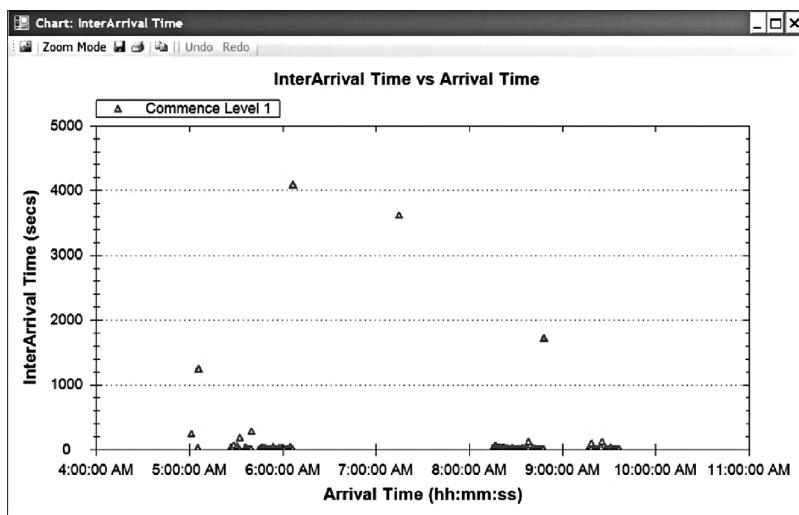


Figure 16.4 Data analysis graphical output.

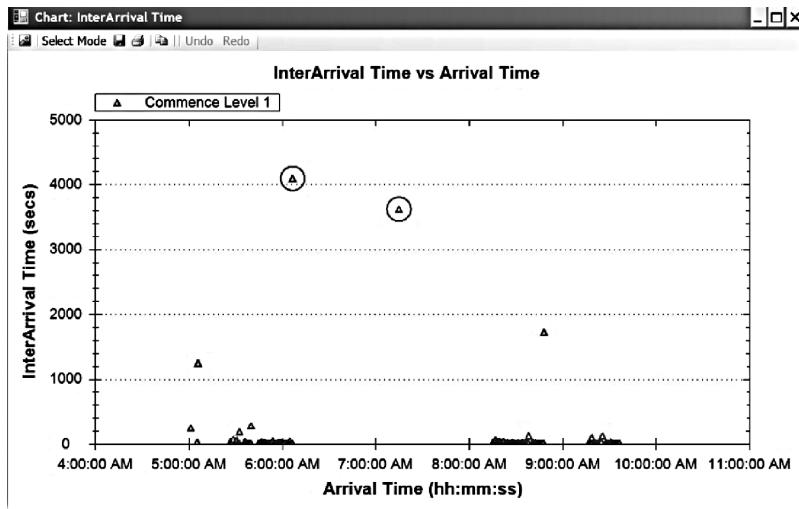


Figure 16.5 Multiple point selection.

With the data selected, it can be removed from the graph, as shown in [Figure 16.6](#).

Once the data is cleaned, it is available for use in the model. An overview of the data cleaning and analysis tool is shown in [Figure 16.7](#).

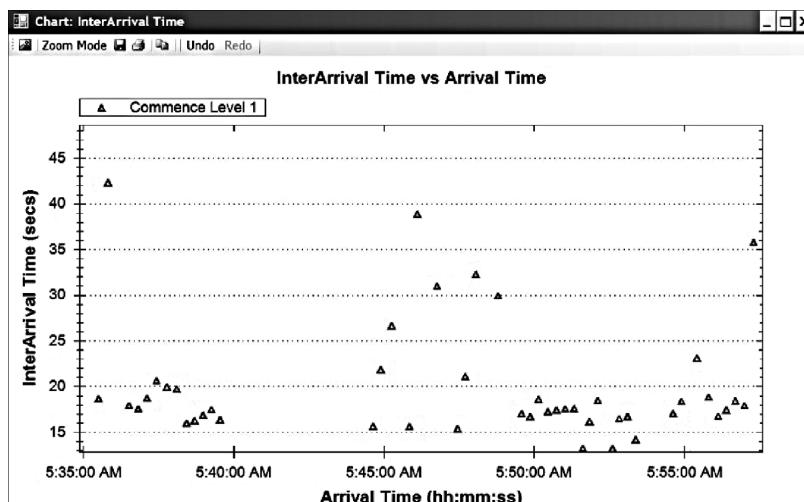


Figure 16.6 Dataset with erroneous data cleaned.

16.5.3 Model development phases

Development of the model can be split into two sections, a high-level model to establish relationships and verify data and detailed models to use for verification and analysis.

The high-level model is used for verification of logical flows within the system and is used to establish relationships within the submodels. A high-level model provides a more suitable tool with which to go to various management levels from different companies to confirm plant operations and flows. It also gives credence to requests for detailed information from the company. The information requested, such as arrival times of goods over a large time frame, can require a large effort on behalf of the company to produce. The high-level model can help convince the company of the need for the data and show the benefits that the simulation process will generate.

The detailed models of each system are used to verify the data received from the system being modeled and to analyze the system. They require information on arrival rates and volumes, process flows and timings, deadlines, staffing levels, and all the variables commonly associated with a simulation analysis. The detailed models require validation, as discussed in Section 16.5.4, and analysis, which is covered in Section 16.5.5.

16.5.4 Model validation and verification

The role of model validation is to show that the detailed model is a true representation of the actual process. Ideally, this should be done with the experts of the system being modeled. Data that was supplied by the company in question

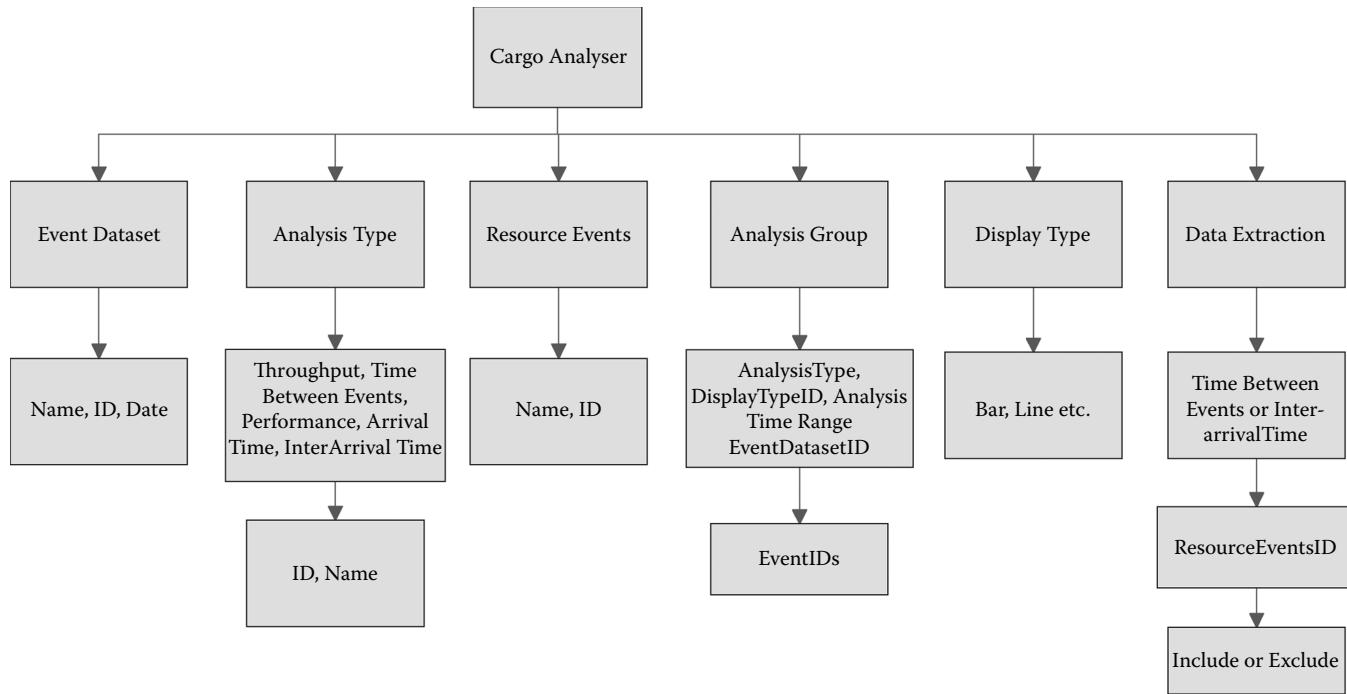


Figure 16.7 An overview of the data cleaning and analysis tool.

should be compared with the data being entered into the model, i.e., arrival rates and volumes, and any assumptions that have been made clearly stated. The model should be run with animation to give more weight to the data that the model produces. Once the process flows are verified, data produced from the model can be presented and any discrepancies discussed. Once the model is recognized as a true representation of the real system, new scenarios can be run to answer “what if” questions about the process and to tune real-world operational policies.

16.5.5 Subsystem analysis

Techniques for detailed analysis of subsystems are presented in this section. As the analysis will focus on the flow of items, either people, bags, or cargo, the models are designed to capture events as the items pass through waypoints in the system. This allows for the analysis of the path taken through the model. Travel times, bottlenecks, utilizations, arrivals, and exits are examples of the data that can be analyzed. An overview of a tool to analyze the data generated by the models is presented in Section 16.5.5.1, while an automated input generator is described in Section 16.5.5.2. Flow control through the system is described in Section 16.5.5.3.

16.5.5.1 Data analysis tool

A hierarchical view of the data analysis tool is shown in [Figure 16.8](#). The analyzer is implemented as a separate executable from the simulation model, but has the ability to modify input data files used by the models. This enables the analyzer to run independently and update scenarios with knowledge gained through the analysis of past model runs. The analyzer is designed to make the analysis of new results simple, logical, and with the minimal effort.

The analyzer works by describing a criterion of analysis, defined in the analysis method; it could be throughput, travel time, or amount of items in a certain section of the system within a time interval. This criterion is basically a search criterion specifying when an item passes certain nodes or does not go through a certain node. It could describe an item that goes through node one before going to node two and so on.

Each defined analysis criterion is then assigned to an analysis collection. The analysis collection will hold a number of analysis criteria. In terms of a system of systems, we have a set of criteria from the analysis method system that is embedded inside an analysis collection system. The analysis collection also contains the analysis range and interval definition to fully describe an analysis collection. Each of the criteria within a collection can then be assigned to an individual output run, so that multiple output runs can be analyzed and compared against each other. In order to display the result, the display format is defined and appended to the analysis collection to present the analyzed result.

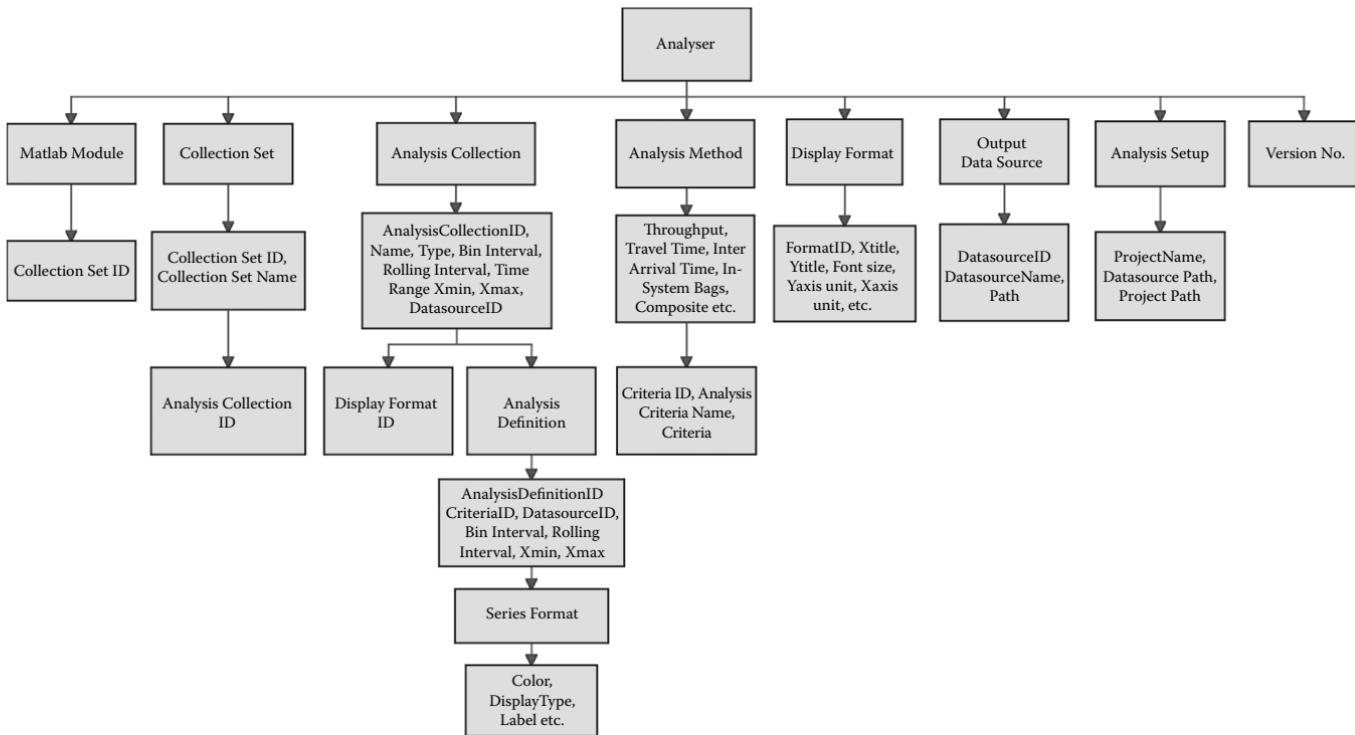


Figure 16.8 Analyzer overview.

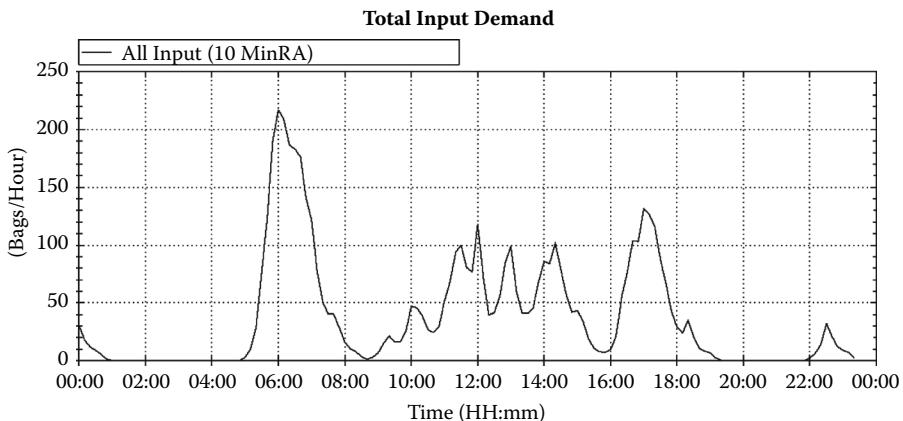


Figure 16.9 Analysis sample output.

The analysis collection set is a grouping method that allows multiple selections of analysis collections to be analyzed simultaneously. This is necessary in order to specify a group of subsystem resources or sections within a large system to analyze together. It could be a modification to the model, and we want to analyze it in its own analysis collection set. This would allow previously defined criteria and analysis collection sets to be used to analyze modified systems.

Output results from analysis of the collection set can then be further analyzed for statistical significance due to changes. This is done by integrating Matlab to perform statistical analysis. An example output from the analyzer is shown in [Figure 16.9](#). In this figure the inputs to the system are shown as a 10-minute rolling average over the day. It can be seen that there is a heavy peak at 6 a.m. and a later peak at 5 p.m., reflecting the busier time of day at an airport in terms of passenger demand.

16.5.5.2 *Input generator*

It is important to be able to test each subsystem and even sections of each subsystem with a variety of inputs in order to observe how the system will behave over the full range of possible input combinations. Rather than using the system inputs that were determined during the data collection, it is more efficient to generate the desired range of inputs using a highly configurable method.

The basis for the input generator is to provide a simple method to test specific sections of models using a range of input conditions. To achieve this, the input generator uses a sinusoidal waveform with parameters to specify the amplitude, frequency, and relevant time period.

Once the waveforms have been configured, they can be applied to sections of the model during runtime. For example, if we are interested in how a specific merge is performing in a BHS, we are able to specify flows inputting

into the main line and merge line, and using the data analysis tool, we can determine how the merge is performing.

16.5.5.3 Automated flow control

Flow control is an important aspect for systems that reflect networks. The baggage handling system (BHS) within the airport is a system that uses a set of rules for flow control to route bags from the check-ins to the exits within the system. Determining the flow control for the baggage system manually is a time-consuming task, and without testing it is not possible to define an optimal set of rules to route the bags through the system.

Before a bag can be routed to its system exit, it must have been identified by a barcode scanner and determined to not contain banned items via X-ray screening. This is the role of the baggage handling system. Ideally, the BHS will perform these operations on the bags and deliver them to the correct exit in the minimum possible time. A 3-D simulation model representation of a BHS is shown in Figure 16.10. This example represents a small section of the conveyor system showing automatic tag readers (ATR) and X-ray machines.

An algorithm has been developed to generate the set of flow control rules to control the conveyor system. The algorithm treats the system as a white box; it can see within the system, but has no a priori knowledge of the system and uses learned metrics to determine effective methods to route bags through the network. Through knowledge gained of bag status changing (due to either being scanned by ATRs or screened by X-ray machines) and system loads, the algorithm is able to generate a set of rules to provide flow control through the BHS. The set of rules are then implemented with the simulation model to determine their effectiveness and look for possible enhancements.



Figure 16.10 An example of a baggage handling system showing X-ray machines and tag readers.

16.5.6 Overall system performance

Performance of security systems in airports has previously been applied to baggage screening solutions [3] and a cost model approach in [6]. These methods consist of determining the effectiveness of particular security solutions and comparing the cost to the effectiveness of the solution. A similar approach can be taken when analyzing the airport from a system of systems perspective.

Determining the overall system performance begins with investigation of the key criteria used to define a cost function. If we look at the key inputs to the model from [Figure 16.1](#), the volumes of passengers and freight, as given by flight schedules, and their individual arrival rates are of interest. We are also interested in the required security level to be achieved and the number of available screening devices and their capabilities. These criteria can be grouped together in terms of cost. This cost-based approach will determine the expense involved in achieving varying security levels.

By running each detailed model, we are able to see what effects these criteria incur on the individual system. Each run of a detailed model requires data analysis which normally is a lengthy process. By automating this analysis using the data analysis tools described, we not only save time but can bring these results back into the higher-level system and feed the new data into the subsystems. Through this iterative behavior, various scenarios are able to be investigated, and some useful traits can become evident in the model. For example, the sharing of staff to operate the X-ray machines is possible, due to differences in the peak times between different freight operators and the baggage system. Now this can only occur if the operators are trained on all the models of X-ray machines in use. However, by reducing this pool of available X-ray machines, the training costs will be reduced, and the overall system costs to achieve a specific security level will also be reduced.

16.6 Conclusion

A system-of-systems modeling and analysis perspective of an airport has been presented. In order to analyze the airport, new tools and methodologies are required. The tools automate the data collection and analysis required for the detailed models, while the methodologies show how the subsystems can be combined into a high-level model to characterize the system inputs and the interactions between variables of the submodels.

The system-of-systems approach has helped to analyze the entire airport's approach to security. Rather than treating each aspect of the airport as a separate entity, by recognizing the need for a holistic approach, we have been able to gain significant insight into the most appropriate ways security can be implemented to achieve the highest level of security with the least amount of costs incurred.

16.7 Future challenges in airport operations

While we have mainly been considering the security side of airport operations, there are many more issues that can be addressed with a similar approach. Care must be taken with these systems, as they are closely connected and often are near to capacity [5]. Similar tools for data acquisition and analysis will make the task more manageable, and in the data collection phase a better understanding of the environment is often achieved.

Vast amounts of effort have been put into the analysis of separate or individual airport systems. In order to combine these efforts to achieve a more realistic model of the world, a common approach to problems could be used, an approach similar to the system-of-systems approach presented here.

The challenge exists to model emergent behavior further. Although modeled here to some degree, better tools and methods are required to model this important feature of system of systems. With improved tools and techniques, the emergent behavior can be observed and analyzed to provide further understanding of the very complex system under investigation.

References

1. Leone, K. and R. Liub. 2005. The key design parameters of checked baggage security screening systems in airports. *Journal of Air Transport Management* 11: 69–78.
2. Dawson, J. 2002. National labs focus on tools against terrorism in wake of airliner and anthrax attacks. *Physics Today* 55(1): 4.
3. Kobza, J. E., T. J. Candalino, and S. H. Jacobson. 2004. Designing optimal aviation baggage screening systems using simulated annealing. *Computer and Operations Research* 31(10):1753–1767.
4. Dirner, C. 2007. TSA rolls out new screening technology. ABC News. <http://www.abcnews.go.com/Technology/story?id=3716917&page=1>.
5. Wieland, F. and A. Pritchett. 2007. Looking into the future of air transportation modeling and simulation: a grand challenge. *Simulation* 83:373–384.
6. Jacobson, S. H., T. Karnani, J. E. Kobza, and L. Ritchie. 2006. A cost-benefit analysis of alternative device configurations for aviation checked baggage security screening. *Risk Analysis* 26(2):297–310.

chapter seventeen

Knowledge amplification by structured expert randomization—KASERs in SoS design

Stuart H. Rubin

Contents

17.1	Introduction	422
17.2	Problem statement	423
17.3	Background	423
17.4	Randomization	428
17.5	Expert compilers	429
17.6	The knowledge acquisition bottleneck	430
17.6.1	A theoretical perspective	431
17.6.2	Church's thesis	431
17.6.3	Pseudo-code	431
17.7	Knowledge-based language design	434
17.7.1	An example	434
17.8	Expert vs. conventional compilers	435
17.9	Expert optimizing compilers	437
17.10	Knowledge reuse	437
17.11	An information-theoretic basis	439
17.12	Knowledge amplification	439
17.13	Mining for rules	440
17.13.1	Approach	441
17.14	Refrigeration system design example	443
17.15	Conclusion	449
	Acknowledgments	449
	References	450

17.1 Introduction

The U.S. Army needs robotic combat vehicles that can autonomously navigate the battlefield and carry out planned missions that necessarily embody unplanned details. On one end of the spectrum lie the simple insect-like robots that have been popularized by Brooks at MIT [1]. Their simple behaviors can be evolved much in the same manner as a simple program can be created through the use of chance alone. Of course, more complex behaviors cannot be tractably evolved, because the search space here grows exponentially. What is needed are heuristics to guide the evolutionary process. We can, of course, program search strategies and have indeed programmed robots to perform a myriad of complex functions—from the robot Manny’s (U.S. Army) ability to walk the battlefield to unmanned aerial vehicles (UAVs). What is needed is a means to program more complex and reliable functionality for constant cost. That is, a system of systems (SoS) is needed. For example, one can program a robotic vehicle to sense and avoid an obstacle on the right, say. But then, what is the cost of programming the same robot to sense and avoid an obstacle on the left? It should be less, and is to some extent if object-oriented component-based programs are written. The problem here, though, is that the cost is front loaded. The programmer needs to know *a priori* most of the domain symmetries if he or she is to capture them in the form of objects. A better solution is to do for symbolic programming what fuzzy logic did for Boolean logic [2,3]. That is, we need the programmer to be able to constrain the robot’s behavior in the form generalized actions. Then, instances of these generalizations constitute the desired program. Even search-control heuristics can be acquired through the exercise of this paradigm. Instead of programming the robot to do a specific action, we program the robot to (heuristically) search a space of actions for one or more that is consistent with environmental constraints. The writing of such programs is easier for the human, and the constraints that instantiate them serve to render the search space tractable for the machine.

The advocated *fuzzy programming* approach also has ties with case-based reasoning (CBR) [4]. Using traditional CBR, a program or case is needed for navigating an obstacle on the right and another for navigating an obstacle on the left as explained above. If a proper representational formalism is supplied, then these two cases can be fused into one. This formalism will allow environmental constraints to elicit one or the other navigational behaviors. It also allows for the tractable evolution of behaviors not explicitly programmed for as emergent behaviors (e.g., the simple behaviors of going forward or backward). The development of a practical representational formalism for the Army’s Future Combat System (FCS) is addressed below. The development of a more general theory of formalization holds promise for the partial mechanization of creativity and invention [5]. The spin-off benefits of this line of work thus hold even greater promise and will serve to leverage the initial investment, albeit, for the longer term [6].

17.2 Problem statement

Can a computational intelligence be designed, which allows the knowledge engineer to focus on what he/she does best—namely, the specification of general knowledge and algorithms, while off-loading the details to the machine (i.e., those details that the machine is best equipped to handle, namely, the instantiation of general knowledge and algorithms)? Moreover, can such a computational intelligence serve the U.S. Army's FCS program through the evolution of autonomous robotic behaviors and strategies? The solution of this problem will entail the symbolic evolution of dynamic heuristics for constraining effective search.

17.3 Background

This chapter was motivated by the failure of the Japanese Fifth Generation Project [7] and embodies a prescription for success. Looking back, in the mid-1980s the then Japanese Ministry for Industry and Trade (MITI) boldly promised the world that they would build an intelligent architecture that was more advanced than any to date. Over \$250,000,000 (in 1980 U.S. currency) was invested in this project. MITI proposed to build a massively parallel architecture that realized the predicate calculus (Prolog) for intelligent processing [7]. The reasons behind the failure of that project are now clear: First, much of what we term common-sense reasoning is not amenable to deductive reasoning (i.e., it is open under deduction). Second, Prolog's back-cut mechanism did not incorporate any heuristics to delimit the combinatorial explosion of resolution.

In 2002, Japan fired up an ultrafast vectored supercomputer that is approximately ten times faster than its fastest U.S. competitor [6]. The advantages attendant to making computers even faster are shared with those of acquiring smarter heuristics. We are just beginning to gain the computing power to understand what is going on in systems with thousands or millions of variables; even the fastest machines are just now revealing the promise of what is to come. In addition to computational robotics, heuristic methods will serve the gamut from climatic modeling to molecular biology and, of course, in the simulation of nuclear fusion, to name just three. According to Tristram [6], "The United States won't be at the cutting edge of simulation—the "third pillar" of scientific discovery—if the performance of its computers lags."

War fighters need state-of-the-art technologies that provide support in the analysis, assimilation, and dissemination of real and simulated digitized battle-space information. In particular, analytic tools are needed for logistical planning (e.g., optimization and resource allocation problems). The tractable discovery of near-optimal paths using concurrent constraint-based programs will enable the computation of mission plans of ever-greater complexity.

Expert systems have for two decades promised to deliver a rule-based computational intelligence to the PC. This goal has been largely fulfilled,

and the required level of computational intelligence has been realized using far less computational power than the human mind is capable of. If we are someday offered hardware that is several orders of magnitude faster than that currently available, we say that our expert systems do not need this much raw computing power and cannot effectively utilize it. Therein lies a dichotomy. Intelligent systems must be able to use all the computing power available, given an arbitrary problem to solve. Otherwise, the intelligent system may be mathematically defined to be trivial because it does not utilize (heuristic) search.

Figure 17.1 depicts the opportunity for increasing the cost effectiveness in the design and realization of intelligent, or search-based, software. The point labeled "A" depicts the high cost associated with a random evolutionary strategy—that is, one where all of the design is off-loaded to a very high-end machine. The asymptote to the left of this point evidences that this strategy quickly grows to be intractable. The point labeled "B" depicts the high cost associated with current programming practice—that is, where the entire program design lies within the province of the knowledge engineer without the associated benefits afforded through the deployment of evolutionary tools to assist with development. The asymptote to the right of this point shows that this strategy quickly grows to be impractical. The point labeled "C" represents an optimum that varies with the operational domain. Here, the knowledge engineer does what he or she does best (i.e., high-level design work), and the computer does what it does best (i.e., a heuristic search for detailed specifications). This represents the optimum symbiosis of human and machine for the production of intelligent software. Notice that, if a more powerful machine becomes available, then the designer's task becomes proportionately reduced and vice versa. Again, nothing is underutilized.

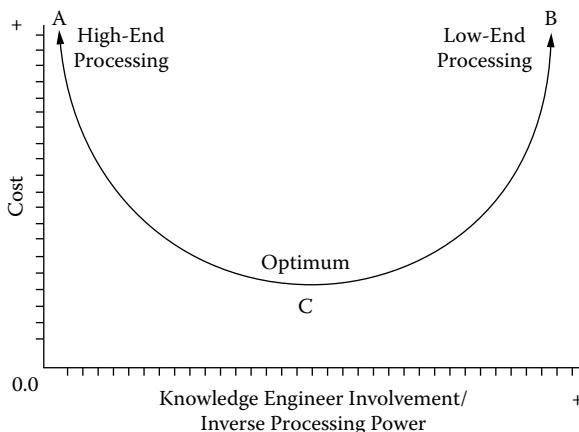


Figure 17.1 Cost effectiveness of intelligent software design.

Not every programming application or computational intelligence is a candidate for realization using what we term, the *heuristic calculus*. The criteria for the successful realization of this methodology are as follows.

- The operational domain needs to be amenable to hill climbing. For example, finding the correct set of numbers to a combination lock is not defined as hill climbing inasmuch as one cannot ascertain that one combination of numbers is closer to the true combination than another. On the other hand, the task of driving from city A to city B using an ordered set of intermediary cities such that the total distance driven is minimized exemplifies hill climbing. It also evidences that there are heuristic solutions to the NP-hard Traveling Salesman Problem (TSP). Note that the use of hill climbing does not preclude the optimization of recursively enumerated solutions, if any.
- The evolved design can be tested using sets of input/output pairs, or auxiliary expert systems to evaluate relative system performance. The advantage provided by including characterizing I/O examples in the process of program specification (and necessarily randomized testing by definition [8]) is not theoretical, but rather is similar to the advantages accorded the programmer working with, say, a fifth-generation language such as LISP or SCHEME in comparison with one working with a universal machine language. That is to say that, in practice, there is no comparison!
- The design is hierarchically decomposable into sets consisting of relatively small objects. Small objects facilitate tractable evolution (see below). Additionally, the problem domain should be reducible to similar, but simpler, problems to facilitate tractable heuristic evolution. For example, the game of chess can be reduced from an 8×8 board to a 6×6 board with far less reduction in the space of possible strategies. Indeed, it was customary to use a 6×6 board in the early days of computer chess when computer memory was at a premium. Again, computer time, in general, is always at a premium.
- The design allows for the definition of labeled *sets* of effective alternatives. For example, in the design of primitives for robotic movement, the set of allowable moves might be DIRECTION: {LEFT, RIGHT, FORWARD, BACKWARD}.
- A real-time model and/or simulation can be cost-effectively designed to provide feedback on the success or failure of an instance of the user design. Note that this model or simulation must be hardened against system crashes to allow for rapid evolution. The choice between the use of a model or simulation depends upon the relative costs/benefits of modeling and building a simulation. In particular, simulations have the advantage in that they can usually be run far faster—allowing for the evolution of better systems. However, models can usually provide more exacting and convincing demonstrations (e.g., using time-lapse photography). Clearly,

the use of both simulations and models is favored where practical. In Figure 17.2, we see that simulations can often outperform models—but only at a cost. The figure also shows that, to the left of breakeven, one should opt to use simulations and then switch over to models to the right of this point. The exact crossover point is domain dependent. This two-pronged approach will serve to maximize the region of interest (ROI).

There are two specific applications, within the context of the Army's FCS, that are amenable to the use of a heuristic calculus for computational intelligence. First are the applications involving programmed behaviors. In this context, RoboScout, a robotic reconnaissance vehicle developed by the U.S. Army, has three fundamental behaviors:

1. Obstacle detection and avoidance
2. Actions to be taken on enemy detection
3. Actions to be taken on enemy contact

RoboScout needs to respond to dynamic and to some degree unpredictable situations with one among several, if any, appropriate actions. It is not too difficult to specify situation-action pairings in general terms, but the “devil lies in the details.” For example, the knowledge engineer might write: IF detected THEN avoid harm. This general rule might be instantiated using the heuristic calculus, on the basis of experiment, to yield for example: IF laser detection registered THEN execute object for stealth retreat. Furthermore, while the need for the heuristic calculus is apparent for a single robot, the need is even greater in the case of multiple robots whose complex behavior requires communication and coordination.

The heuristic calculus is consistent with emergent behavior. Here, the attendant benefits associated with its use are greatly leveraged. Other tasks that can benefit from the heuristic calculus include critical axis of advance,

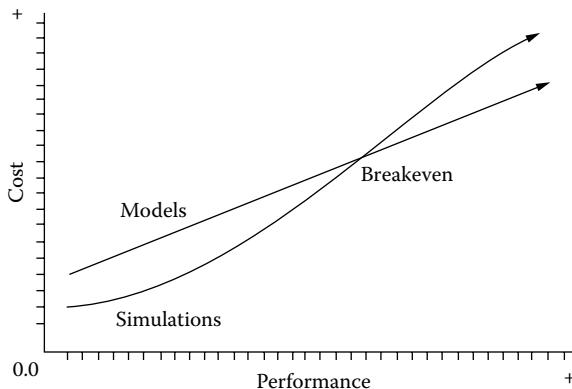


Figure 17.2 Cost effectiveness of intelligent software design.

surveillance ranges (direct fire, lethal), dead/shadow spaces that need additional reconnaissance, sensor selection for tasking based on their capabilities, sensor ganging, sensor target pairing, logistics (refueling), and zones to input a commander's intent for the mission.

Another U.S. Army robot, named Oberwatch, implements the generally described task of sensing without being sensed. It will traverse sand dunes and attempt to peek over the top to gather reconnaissance. Its mission cannot be planned in great detail, if for no other reason than because the sand dunes change their formation with changes in the prevailing winds. Here again, the knowledge engineer can supply generic domain knowledge and have the system of collaborative robots evolve behaviors that are consistent with changing conditions and mission parameters.

There is no inherent reason why the heuristic calculus cannot be similarly applied to a task such as developing sensor arrays, which best serve a defined mission. Indeed, the arrays can be evolved to be reconfigurable for several related missions. More generally, we say that the heuristic calculus can find application in computational creativity. This should come as no surprise, given that the heuristic calculus takes up where the predicate calculus is essentially incomplete. The development of computational creativity may be expected to have far-reaching implications for the development of network-centric warfare. This follows because, instead of just programming a computer to execute our immutable instructions, we are now moving to a position whereby we can state our domain knowledge and provide the computer with a means by which it can expand it by way of (heuristic) search. It may be expected that the profoundness of this paradigm shift will become evident in due course.

In particular, computational creativity serves to enable *strategy evolution*. It is consistent with a *naturalistic* approach to decision making, whereby under fast-paced, uncertain, and dynamic conditions, the Commander will act and react on the basis of prior experience, generating, monitoring, and modifying plans to meet the needs of the situation [9]. Furthermore, the Collective Intelligence Module (CIM) would employ the heuristic calculus and assist the Commander/Combat Service Support with planning and replanning processes (i.e., Course of Action [COA]), mission rehearsal, identification of enemy unit composition and intent, resource management, task expansion, task feasibility, task status, and task optimization.

A war game simulation can be written and different designs tested. Evolved strategies can be downloaded (i.e., similar to a cross compiler) for use in distinct application software. The end result of strategy evolution can be inserted in planning tools such as CAPES, FCS-C2, MC2, etc. Again, the use of the human and machine are optimized toward a common objective. The plan accounts for the potential dichotomy created by the efficient, rapid, and precise information processing of the robotic elements and the imaginative, experienced, and intuitive reasoning of the human elements of the Unit Cell [9].

17.4 Randomization

Randomization is a term first coined by Gregory C. Chaitin in a 1975 *Scientific American* article entitled, “Randomness and mathematical proof” [8]. This is a detailed concept and explains why certain sequences of numbers are said to be random—i.e., because a generator, which is more compact than the sequence itself cannot be found. Thus, if one were to write the sequence: 0101010101010101 . . . , then you see a pattern and conceive an algorithm that is a compaction (randomization) of that sequence. Here, the sequence is obviously not random. This example shows how the entropy of a software system is decreased if data is represented as a program wherever possible (e.g., Euler equations in lieu of tables of logarithms) [8]. If this is not possible, then the data is said to be relatively random.

Randomization implies that the computer does what it does best and the user that which he or she does best. Thus, the computer may be best utilized as a search engine, while the software engineer/programmer is uniquely valuable as a “conceptual engineer.” Such a human–machine symbiosis is found at the cutting edge of software engineering.

Think of any effective optimization as a transformation. When one transforms pseudo-code to program code, one is in effect optimizing the concept for execution on a digital computer. When one transforms bubble sort into quick sort, for example, again one is optimizing one function for a more efficient function under practical assumptions about the data and the computer. Optimization in the context of this chapter will be taken to mean randomization with respect to space and/or time. Thus, for example, bubble sort is temporally randomized (i.e., $n > 21$) into quick sort. Similarly, quick sort is spatially randomized into bubble sort, since the latter has a smaller core bit-image. Moreover, optimization where not trivial (e.g., peephole optimization in a compiler), needs to be knowledge based. That is, in any non-trivial search for a better program, one needs to apply heuristic knowledge to that search. Otherwise, the search may be intractable on any computing device. All of this ties into the importance of knowledge-based software engineering.

Randomization is not limited to humans or machines. For example, bumblebees learn to fly the hypotenuse of a triangle formed from previous legs of the same journey. Here, the reduced flight path is in effect a randomization.

The opposite of randomness, in Chaitin’s sense of the word, is symmetry. That is, if a pair of objects are not mutually random, then they are mutually symmetric. For example, two fruits such as an apple and an orange are mutually symmetric by many characterizations, but not by color; whereas, a fruit and a rock are mutually random by most definitions.

Fruits may be genetically engineered by making use of the genetic symmetries. By inserting different genes into a protein one gets a new genotype, which may result in a new phenotype, which may be viable and may be superior under certain conditions. These techniques have given rise to disease- and

drought-resistant crops and several new antibiotics. Rational drug design, as it is called, is rational only because it makes use of computational intelligence.

A network architecture, such as defined by the Web, can be employed to mine knowledge from information. Such knowledge can be applied to the construction of extensible intelligent search engines, which further serve to put new knowledge at one's fingertips. The point to be made is that mining is in effect randomization and here serves to make for an intelligent Internet. It can be done!

As a second example, closed-loop neural networks cannot be mathematically studied in detail as a consequence of the Incompleteness Theorem. Yet, even here there is randomization at work. Each neuron temporally and spatially sums around 20,000 inputs to yield one output signal, which is either excitatory or inhibitory. This n:1 mapping defines a randomization, which necessarily occurs in a system of randomizations driven by feedback. Experiments have shown that such systems converge. It should be noted that these systems are ideal candidates for photonic realization through the use of spatial light modulators (SLMs). This is because of the nature of the connections and the relaxed need for exactness or even speed in the system. Indeed, it may be said that the relaxation of the precision required by strictly digital computers may enable a revolution in computational intelligence. Of related interest is a recent televised newscast, which stated that researchers have succeeded in producing a primitive calculator using brain cells.

17.5 Expert compilers

Expert compilers or systems have control mechanisms (i.e., the inference engine), explanation facilities (i.e., the explanation subsystem), and at least one knowledge representation language with which to express the rules. The higher the level of the knowledge representation language, the easier it is to specify complex rules. It then follows that the rules, control mechanism, explanation subsystem, etc. can be bootstrapped (i.e., more rapidly programmed) using the now effective pseudo-code! The more complex the expert compiler becomes, the higher the level of pseudo-code that can be realized, which of course means that a more complex expert compiler can be created and so on. If one follows this chain of reasoning, then one will see that it realizes the ultimate CASE (computer-assisted software engineering) tool. It potentially contributes the most to the intertwined fields of artificial intelligence and software engineering.

Improve the capability to code, and you improve the expert compiler. Improve the expert compiler, and you improve the capability to code. This idea is achievable, at least in theory, through the randomization of software patterns in the form of a grammar and the use of an evolving grammar for assisting in the specification of symmetric rules. A two-level expert compiler is depicted in [Figure 17.3](#).

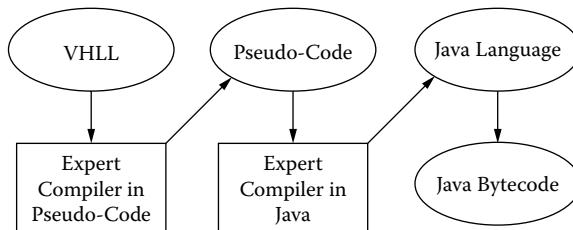


Figure 17.3 A two-level expert compiler.

Here, an expert compiler written in pseudo-code transforms a very high-level language (VHLL) to pseudo-code. The pseudo-code is then transformed by an expert compiler written in Java to the Java language, which is then converted to executable bytecode. There are two levels of bootstrapping here, because there are two levels of expert compilers. The use of multiple levels of expert compilers is a randomization. It represents a segmentation of knowledge bases from the standpoint of functionality, removing symmetry, and thus randomizing. The same pseudo-code compiler can be reused in the expert compilation of many different VHLLs. The Layout software CASE tool by Objects Inc. (Danvers, MA) was built through the realization of such a scheme.

Expert compilers are uniquely suited to the compilation of higher-level languages. This stems in part because they are readily maintained and updated—making for an extensible compiler, which in turn makes for an extensible language. Expert compilers can also be networked for the translation of heterogeneous languages (i.e., languages whose semantics are arduous to define). Such languages are extremely high level and suffer from ambiguity in their use. The network serves to bring different domain-specific expert compilers to bear on their translation.

17.6 *The knowledge acquisition bottleneck*

The knowledge acquisition bottleneck refers to the difficulty of capturing knowledge for use in the system. Whether the system is used for evaluating bank loans, intelligent tutoring, or prescribing medical treatments, the question remains: How do we obtain the knowledge used by the system (i.e., code it), and how do we verify it for mission-critical applications? For example, the medical rule, IF the patient is coughing THEN prescribe cough syrup, is usually true. However, an expert medical system having this limited medical knowledge would prescribe cough syrup to someone whose airway is obstructed during the course of a meal, for example! It is lacking in the more specific rule, IF the patient is coughing AND the patient was just eating THEN apply Dr. Heimlich's maneuver with certainty factor = 0.90.

We see from this one small example that knowledge acquisition needs to be relatively complete, lest there be potentially dire consequences. Suppose that you are a knowledge engineer whose task is to write rule bases—each containing

thousands of rules for mission-critical applications (e.g., flight-control systems, medical decision support systems, weapons systems, vehicle guidance systems, and many more). It should be clear that the task is too difficult for human beings, given the current state of the art in knowledge acquisition. It is just that this software is inherently difficult to test.

We will propose a radically new method for cracking the knowledge acquisition bottleneck, but before we do, let us see what the problem really is. First, it should be mentioned that we have evolved higher-level languages because they are more closely aligned with the way in which we think. This should come as no surprise. Indeed, the Windows or other GUI was developed as a metaphor for our spatial-temporal reasoning. After all, we cannot easily learn to read machine code, for we are not machines.

17.6.1 *A theoretical perspective*

Computability theory provides us with a fundamental understanding of the nature of the problem. Rule predicates are recursively enumerable, but not recursive. That is to say that the predicates can only be discovered by search and cannot be functionally characterized. Any search process intrinsically takes time, and that is the fundamental problem underlying the knowledge acquisition bottleneck.

Grammatical inference techniques can suggest predicates, but only through the implied use of a heuristic methodology. Otherwise, there would be a contradiction on the nonrecursiveness of the problem. Fortunately, heuristic search can be extremely powerful in practice and fails just in case the sought knowledge is random relative to that embodied by the grammar. We can live with this theoretical limitation. It turns out that the methodology is potentially extremely good at providing symmetric knowledge with little if any search. Again, that is what it offers us toward cracking the knowledge acquisition bottleneck.

17.6.2 *Church's thesis*

Around the turn of the twentieth century, a scientist named Alfonso Church purported his thesis (i.e., Church's thesis), which states that, if a concept can be clearly and unambiguously represented as an algorithm, then it can indeed be coded. You need not code the algorithm to prove that it can be coded—only if you intend to use it. Thus, we defer to Church's thesis to accelerate our learning.

17.6.3 *Pseudo-code*

To begin with, we need to find symmetry in the above examples and design the pseudo-code to capture that symmetry (i.e., orthogonal programming, which is easier to read and maintain). An example should serve to clarify

this. Notice that there are two types of loop constructs used—For and While. They are symmetric in meaning, but not in structure. Here is the schema:

Repeat forever

(**Initialize** variable to value) // Parenthesized expressions are optional.

Break if variable {<, >, =, <>} value;

(**Body**)

{**Increment, Decrement**} variable;

(**Body**);

End

Thus, the pseudo-code:

Repeat forever

Initialize index = 0; // There would be a rule that sees no decimal
and thus knows that it is an int.

Break if index >= numbers.length; // The expert compiler will transform the relation.

Increment index;

System.out.println (numbers [index]);

End

would be compiled into:

```
for (int index = 0; index < numbers.length; index++)  
    System.out.println (numbers [index]);
```

Now, compare this pseudo-code with similar pseudo-code, which compiles into a while loop:

Repeat forever

Break if position <= 0 | numbers[position-1] <= key; // The expert compiler can translate the logic.

numbers[position] = numbers[position-1];

Decrement position;

End

would be compiled into:

```
while (position > 0 && numbers [position-1] > key)  
{  
    numbers [position] = numbers [position-1];  
    position--;  
}
```

Now, it is clear that a human can do the compilations. Next, you should observe how an expert compiler can be made to do the same. [Table 17.1](#) provides two sample schema-based rules that determine if a *for* or a *while* loop is called for. Note that they are only based on the previous examples.

Table 17.1 A Sample For-Loop Rule

PSEUDO-CODE	EFFECTIVE CODE
Repeat forever Initialize variable1=value1; Break if variable2 R1 value2; Increment variable3; Body; End	for $(type$ $variable1=value1;$ $variable2$ $!R1$ $value2;$ $variable3++)$ Body;

Also, it is not shown here that rules will apply in sequence. Different schemes, including a tagging scheme, can be used to process the partially transformed code. The partial transformation serves as a context to enable subsequent transformations. By tagging the transformed code, one insures that it will not itself be subjected to subsequent transformations. The tags are all removed upon completion of the transformation (Table 17.2).

Table 17.2 A Sample While-Loop Rule

PSEUDO-CODE	EFFECTIVE CODE
Repeat forever Break if variable1 R1 $value1 \mid variable2 R2$ $value2;$ Body; Decrement $variable3;$ End	While $(variable1 !R1 value1$ $\&& variable2 !R2$ $value2)$ Body; $variable3--;$

Next, observe several items. First, the capability to pattern-match the rule antecedent schemas is facilitated by having a very high-level language. That is why it helps to bootstrap the expert compiler in at least two levels. The capability to compile pseudo-code facilitates writing the rules. Here, the pseudo-code should be designed to facilitate high-level textual comparisons and substitutions. Conversely, observe how difficult it would be to test if “variable1 = value1” in assembler. You also need a symbol table to track variable definitions. Having the proper pseudo-code to work with would facilitate this. Once facilitated, you could more easily write rules, leading to higher-level languages, which facilitate writing rules, and so on. Remember, a small gain in the level of the language will result in an enormous improvement in programmer productivity—not only in time saved writing code, but in time saved in debugging it and reusing it!

17.7 Knowledge-based language design

Ever stop and wonder how it is that we design everything from assembly languages to fifth-generation languages? You will find, on inspection, that domain knowledge is used. This knowledge includes a knowledge of the type of problems to be solved as well as software engineering issues—including readability, testability, reuse, and maintenance. It follows that methodology drivers can be constructed, which will, at a minimum, facilitate the software engineer in the process of language design. Such methodology drivers are to be driven by our inferential expert systems, which themselves are underpinned by higher-level languages, which in turn are supported by expert compilers! In the theoretical limit, such methodology drivers will design ever-better languages without human input—that input being replaced by pure chance.

There are two fundamental routes by which higher-level languages can be designed (e.g., pseudo-code). The first approach is termed, *bottom-up design* and entails finding and extracting symmetries from the target-level code (e.g., Java). The second, complementary approach is termed, *top-down design* and entails the development of a structured high-level (i.e., random) language for the target domain. Both approaches may be applied in any particular design. This is termed, *middle-out design*. For example, it may be found that the initial bottom-up design has resulted in subsets (e.g., "{}") that can be easily generalized (e.g., The set containing "{}", "()", and "[]" might be a generalization in some instances. Note that a limited number of errors here can be corrected through the acquisition of more specific rules.). Similarly, it may be found that the initial top-down design has resulted in language constructs that are too difficult, if not impossible, to translate. Again, a methodology driver can provide the software engineer with assistance here.

17.7.1 An example

To begin with, just as it is more efficient to reuse small software components or more general rules, so it is with designing an expert compiler. The principle of *problem decomposition* should be applied. That is, render many small steps in the process of transformation in lieu of several relatively large ones. This will actually make it all easier to do! Next, formulate the rules using transformative pseudo-code (e.g., bottom-up steps 1–6, or similar). Rules should translate concepts (e.g., swap, bracket, et al.) in preference to mere syntax (e.g., " $\{x\}$ " → " $[x]$ ", which is a patch). That is, the rules should translate objects that are components, or can be thought of as components. This serves to make them more reusable. Reorder the rules from most specific to least specific and iteratively apply them to the initial pseudo-code. When no more rules apply, the pseudo-code will have been transformed into

Java. If errors occur, then these errors will be localized to a single step (i.e., box transformation). This makes it easy to repair the transformation rule, or where necessary, add a more specific *patch* rule to the knowledge base. The resulting Java code should perform the indicated function when run in the lab. This same process is repeated for each additional program, written in pseudo-code, for which translation to Java is desired. The rule base should be run to exhaustion on the new pseudo-code. Then, if necessary, specific rules should be added to the rule base that complete the transformation to Java. Ideally, the resulting rule base will be (random-basis) tested against all known pseudo-coded programs to make sure that a transformation error has not been inadvertently introduced. Handle such errors with a more specific patch as before.

17.8 *Expert vs. conventional compilers*

So far, we have seen that an expert compiler merely substitutes for a conventional compiler. That is, it has done nothing that a conventional compiler could not do. Indeed, the first compilers were rule based, but were abandoned in favor of faster table-driven compilers. However, as will be seen, expert compilers facilitate update operations and, unlike conventional compilers, are far more amenable to concurrent distributed processing, since rules can be processed in parallel.

Expert compilers are no more universal than are their underpinning expert systems. This will not be a problem so long as we crack the knowledge acquisition bottleneck. In fact, human programmers serve as expert compilers. You would not necessarily hire a physician to diagnose why your automobile is “coughing.” Similarly, you would not necessarily give an algebraic specialist pseudo-code pertaining to computational chemistry and expect him or her to properly translate it without assistance or further training. The capability for knowledge acquisition is the key, and this in turn hinges on a capability for higher-level representation.

Conventional compilers will not work where knowledge is required to complete the translation. That is, a consequence of a capability for transforming any program is the absence of domain-specific knowledge. Expert compilers offer such universality, but akin to their human programmers, provide software engineers with captured domain-specific knowledge. If this knowledge is relatively inexpensive—as is the case when the processes of knowledge acquisition are not bottlenecked—then the programming task is randomized in proportion to the applicability of the knowledge. For example, suppose that it was desired to compile the instruction, “Take out the Windows trash.” Conventional compiler technology is not designed to do this because conventional compilers are not meant to apply knowledge to the translation task. On the other hand, expert compilers could bring knowledge

of the Windows operating system to bear on the desired transformation. The result of transformation is:

1. Find the Windows Trash Can.
2. Check if the Trash Can is empty.
3. If the Can is empty then End.
4. Empty the Trash Can.
5. End.

Here, a knowledge of the Trash Can enabled the user to express himself or herself without the usual details. In a very real sense, the knowledge base understands what a Trash Can is and how it works. This knowledge base is also extensible.

Grammatical inference can capture the symmetry needed to extend this knowledge base at minimal cost. The actual mechanics follow below. To the extent that there are differences, the symmetric definitions are populated by randomness. Here is a mostly symmetric compilation of the instruction, “Put x in the Windows trash.” See if you can pick out the inherent random differences:

1. Find the Windows Trash Can.
2. Check if the Trash Can is *full*.
3. If the Can is *full* then End.
4. Put x in the Trash Can.
5. End.

The expert compiler can be updated so as to empty a full can. Notice how the expert compiler itself is used to bootstrap the following definition.

1. Find the Windows Trash Can.
2. Check if the Trash Can is *full*.
3. If the Can is *full* then, “Take out the Windows trash.”
4. Put x in the Trash Can.
5. End.

The higher-level code is not only more readable, less error prone, and more rapidly programmed, but it also facilitates reuse and thus the construction of software components. For example, the high-level instruction, “Replace the Trash Can.” could readily be substituted for the similar instruction, “Take out the Windows trash.” Similarly, at a still higher-level, the instruction, “Put x in the Windows trash.” could be substituted for. Each successive level is a randomization of the preceding one.

17.9 Expert optimizing compilers

One difficulty that arises in the bootstrapping of higher-level languages concerns the efficiency of the resulting code. For example, in the previous bootstrap the code will search for the windows trash can twice. It will do so on processing statement one and again on processing statement one when translating the higher-level statement, "Take out the Windows trash." There are several observations to be made here. First, the cost of being maximally efficient is rarely justified. For example, Karnough maps are rarely used in the design of VLSI logic. Second, being maximally efficient on a serial processor is not justified if parallel and/or distributed computation cannot be fully utilized. The significance of this assertion is in proportion to the number of parallel processors. Finally, and most importantly, expert compilers can effect knowledge-based optimizations. Conventional optimizing compilers produce efficient code. Expert compilers can also do this, but in addition they can be applied to the generation of efficient *algorithms*. Indeed, in theory at least, expert optimizing compilers can be self-applied for scalability.

Notice that it is relatively easy to add rules to the expert compiler. This is important because, unlike conventional compilers, it allows you to grow your pseudo-code incrementally. By comparison, recall how difficult it would be to write a large program if one could not grow it incrementally—you could not debug it! You can also request the explanation subsystem to provide an explanation in natural language for any and all steps in a transformative sequence (e.g., Why did you apply that transformation rule here?). Compare and contrast this with the error messages generated by conventional third-generation compilers. The reader is referred to the article in *Computer Design*, 1986, by Geoffry Hinton entitled "Intelligent Tools Automate High-Level Design" for further background information on expert compilers.

17.10 Knowledge reuse

The rules for expert compilation are said to be cases (i.e., CBR). This is because they embody an enormous complexity of information that, while theoretically describable in the form of production rules, would be most difficult to acquire. For example, each of the above antecedent schemas captures the need for the sequence to be, "Repeat forever... End" before a match can even be considered. However, each antecedent schema is also a predicate. Thus, they can be combined in either the form:

IF antecedent schema A is matched OR antecedent
schema B is matched THEN effective code

or the form:

IF antecedent schema A is matched AND antecedent schema B is matched THEN effective code

Note that the first form can be captured by two distinct rules. Schema A could be of the form:

Repeat forever ... {Some combination of Body and Decrement | Increment variable} End

Schema B could be of the form:

... Body Decrement variable ...

Schema C could be of the form:

... Decrement variable Body ...

Schemas D and E would substitute Increment for Decrement in the previous two schemas. Observe now how schema A could be reused with the others:

IF antecedent schema A is matched AND antecedent schema B is matched THEN effective code 1

IF antecedent schema A is matched AND antecedent schema C is matched THEN effective code 2

IF antecedent schema A is matched AND antecedent schema D is matched THEN effective code 3

IF antecedent schema A is matched AND antecedent schema E is matched THEN effective code 4

This is software reuse with a vengeance, because it enables the construction of higher-level reuse tools and so on! Again, observe that the ease with which the knowledge engineer can write such rules is highly dependent on the level of language with which he or she can express themselves. Bootstrapping one expert compiler for the creation of another is clearly the theoretical solution, because it defines a randomization operation. Finally, consider how a grammatical approach could randomize the rule base (i.e., both left- and right-hand sides) and in so doing provide assistance in the selection of conjunctive predicates (i.e., antecedents and consequents). This will minimize the cost of building expert compilers in the limit, which will minimize the cost of building higher-level languages, which will minimize the cost of building higher-level expert compilers, and so on. That is to say that a little help in cracking the knowledge acquisition bottleneck goes a long way toward

improving programmer productivity! Forty years ago, few believed in the concept of a compiler. Now, it is a given. Similarly, less than twenty years from now, the expert compiler will be commonplace.

17.11 An information-theoretic basis

Again, in 1975, Gregory Chaitin laid the basis for much of the new interpretation of information theory with his paper published in *Scientific American* entitled “On Randomness and Mathematical Proof”—see <http://www.umcs.maine.edu/~chaitin/sciamer.html> [8]. This work showed, among related things, that a sequence of random numbers is *random* because it cannot be “compressed” into an effective generating function. That is, random number sequences serve as their own fixed point. Notice that pseudo-random numbers are not truly random, because the generated sequence has an algorithm for its fixed point (i.e., not the sequence itself). If something is not random, it is *symmetric*. The author remains convinced that Gödel’s Incompleteness Theorem along with Chaitin’s related work on information theory and Solomonoff’s work on grammatical inference hold the key to major progress in software engineering, artificial intelligence, and thus automated intelligent tutoring systems.

Intelligent tutoring applications will realize SoSs having the greatest macroeconomic utility because they truly enable societal advancement for uniform cost and are consistent with the American guarantee of equal access to quality education for all regardless of economic class.

17.12 Knowledge amplification

The reader may be wondering where this theory of randomness and mathematical proof fit into cracking the knowledge acquisition bottleneck. A conceptual approach is taken here, and the formal details will be developed subsequently. Suppose that you have a rule in the system to the effect that, IF the patient is bleeding AND the cut is recent THEN apply antiseptic with certainty factor = 0.99. An abstraction, or superclass of this rule is:

IF physical_irregularities AND recent_events_of_medical_interest THEN medical_cures with certainty_factor = X.

The first time that this rule is entered, it is random relative the existing base of knowledge. Superclassing this rule may or may not have random components—depending on whether or not the superclasses have been previously defined. Now, consider that we are writing the previously defined medical rule, IF the patient is coughing THEN prescribe cough syrup. The antecedent, “the patient is coughing” does not match the rule instance, “the patient is bleeding AND the cut is recent.” If it did, then that would be the most specific match that we would use. (Actually, it matches neither predicate in

this instance.) Next, let us try matching the antecedent, “the patient is coughing” against the superclass, “physical_irregularities.” Note that there can be an arbitrary number of superclass levels. Clearly, the former is an instance of this superclass. Given this match, the software informs the knowledge engineer that “recent_events_of_medical_interest” should be checked for the next conjunctive predicate (e.g., the patient was just eating). Of course, the user can say that this is not necessary (as in the case of the initial medical rule), but the point is that the user was provided with a most specific prompt to consider. Clearly, the quality of the knowledge base is raised as a result of the diligence of our hypothetical automated assistant. This is a key concept in understanding our subsequent formalization.

Observe that the resulting two rule instances are symmetric with respect to the defined superclass. By this we refer to the properties that make them symmetric (e.g., the size of the two objects, their color, their composition, etc.).

Observe too that the specification of a rule consequent can be guided by the same mechanism that guided the design of the rule antecedent. For example, if my consequent is a sequence of actions intended to get a robot to walk, then I could easily forget to poll one of many sensors at a critical moment in the locomotion. (This has actually happened to us in programming a turtle-like robot.)

17.13 Mining for rules

Data mining also fits into this unified theory of intelligence. Solomonoff has done much work in the area of grammatical inference. This is the induction of grammars from examples. It has been proven that for context-free grammars or higher, induction must be guided by heuristics. In essence, grammatical inference is the randomization of data. Given that we succeed in formalizing an expert system having automated knowledge acquisition facilities in the form of context-free grammars, it follows that data can be mined to create such expert systems by building on the work of Solomonoff. Note that, if the mining process is itself directed by expert systems, then it is called directed mining. Just as conventional expert systems, such as XpertRule (i.e., which we rate at the top of the line), have data mining tools available, so too will our strategic expert systems. They will be implementations of Kurt Gödel’s and Gregory Chaitin’s theories of randomness and Roy Solomonoff’s related work on grammatical inference. We naturally view this approach as exciting! The details will follow.

Remember, when you want to experience the thrill of an exciting project in the making, you have to be prepared for revision as new knowledge comes into play. After all, is that not the idea of software maintenance? Am I not now doing self-reference, which is the subject of Gödel’s work? Have I not now arrived at a fixed point, which tests your comprehension of the above? Well, at least you now see the kind of reasoning that you were spared—at least to this point.

17.13.1 Approach

Consider programming a robot to take out the garbage. It is not too difficult to imagine this robot emptying an empty can. Simply put, the program does not convey to the machine what humans term *commonsense knowledge*. Experience has shown that common sense is difficult to capture. Now suppose that the same robot could be programmed using the heuristic calculus. Such a design might appear as follows (i.e., in a very high-level readable format).

1. Do not waste time.
2. Empty garbage can into trash receptacle.

Clearly, a desired instance of this design would have the robot ignoring an empty garbage can. Such behavior could, in principle, be evolved.

There is no *silver bullet*. That is, there is no magic algorithm that once discovered will make all our machines smart. Rather, whatever our machines do, they must be programmed for. Thus, the key is not to directly attempt to make our machines smart, but rather, indirectly make them smart by making them easier to program (in the details). Furthermore, not only are higher-level programs easier to write—they are also easier to modify and reuse. Reuse has the added advantage in that latent bugs tend to surface in proportion to the degree of reuse—including the degree of diversity of reuse, whereupon they may be eradicated.

Proximity to a concept and a gentle shove are often all that is needed to make a major discovery—and that is the reason for the drive towards languages of ever-higher level.

Douglas Hofstadter [10]

Certainly, there have been several success stories in the design and realization of higher-level computer languages. The most notable successes have occurred in the realization of third-, fourth-, and fifth-generation, extensible, and object-oriented languages. We can see in retrospect that each of these advances hailed a revolution in software engineering practice. However, with the possible exception of the LISP family of languages (for which its inventor, Dr. John McCarthy received Japan's equivalent of our Nobel Prize in 1996), none of these software advances enabled a revolution in the practice of artificial intelligence. That distinction necessarily awaits a symbiosis of human and machine—one that is based on first principles. The following example will serve to illustrate the point.

Suppose that one desires to automatically synthesize a program for recursively computing the factorial function. However, further suppose that one's

memory is a little lax and about all that can be recalled with certainty are the following specifications.

1. $\{0!=1, 1!=1, 2!=2, 3!=6, 4!=24\}$
2. $\text{fact}(0) = 1;$
3. $\text{fact}(n) = n * \text{fact}(n+1) \mid n + \text{fact}(n+1) \mid n * \text{fact}(n-1) \mid n + \text{fact}(n-1) \mid \dots$

We may write the third specification as:

$$4. \text{fact}(n) = n \{*, +\} \text{fact}(n \{+, -\} 1),$$

or more generally as:

$$5. \text{fact}(n) = n \mathbf{R1} \text{fact}(n \mathbf{R2} 1),$$

where, **R1** and **R2** serve as semantic handles for the associated relational set operators.

Clearly, given the characterizing I/O constraints, the proper instance(s), if any (e.g., $\text{fact}(n) = n * \text{fact}(n - 1)$), will be automatically discovered. This example was chosen as well to illustrate the need for a stack overflow/timer protection mechanism. Also, notice the inherent tradeoff: The more concise (general) the search specification, the less time that it takes to write, and the longer it takes to search. Conversely, the more complex (specific) the search specification, the longer it takes to write, and the less time it takes to search. The cost-effectiveness of this tradeoff was depicted in [Figure 17.1](#).

The designer specification language needs to include primitives that allow one to specify what to evolve and what to hold constant at every step of the evolutionary process. Thus, the heuristic calculus expresses certain similarities with partial differential equations here. The complexity of solving these equations derives from the triangle inequality. Thus, if one module has n variables having m possible values each, then the search space is $O(m^{**n})$. Next, if one can constrain the problem into p modules—each having $\text{ceil}(n/p)$ variables and $\text{ceil}(m/p)$ possible values each, then the search space is reduced to $O(p*(m/p)^{**}(n/p))$ —for an exponential reduction of $O(10^{**}(n \log m - n/p \log m/p))$. The key point to be made is that the search space is under the dynamic control of an algorithm. This algorithm can specify (and learn) meta-heuristics that serve to divide and conquer the search process for what equates to the equivalent of an exponential speedup in computational power. Clearly, knowledge is power!

Ordering or otherwise delimiting the search space is to reiterate a heuristic process. When search constraints are specified top-down by the knowledge engineer, they represent embedded knowledge. When the constraints are satisfied bottom-up through execution, domain-specific knowledge is evolved. Heuristic knowledge is acquired through an attempt to compress or randomize [8] positive and negative examples into respective fuzzy forms that account for what works and what does not. Naturally, heuristic knowledge itself is subject to dynamic evolution as the system further experiences

what does and does not work. The result is that solutions are discovered, and their discovery serves to increase the pace at which better solutions can be discovered through a form of bootstrap.

The use of heuristics can sacrifice *admissibility* (i.e., the guarantee of finding an optimal solution if one exists), but gains several orders of magnitude in the capability to solve otherwise intractable problems. Often, heuristic power can be gained by sacrificing admissibility by using some function for h that is not a lower bound on h^* . An example pertaining to the use of nonadmissible heuristics can be found in Nilsson's 15-puzzle [11]. Here, $h(n) = P(n) + 3 S(n)$, where $P(n)$ is the sum of the distances that each tile is from "home" (ignoring intervening pieces), and $S(n)$ is a sequence score obtained by checking around the noncentral squares in turn, allotting two for every tile not followed by its proper successor and allotting zero for every other tile; a piece in the center scores one [11]. Other heuristics include, but are not limited to simulated annealing, exploiting domain symmetries, scaling from small to large, divide and conquer, problem reduction, as well as other transformational approaches.

The heuristic calculus will formalize processes for heuristic acquisition to guide the search for domain-specific heuristics. The solution will entail bootstrapping the definition of our programming methodology to allow for the programmed discovery of the search control heuristics. Again, the emphasis is on providing a programmable approach—not on any one learning algorithm per se (e.g., Minton's explanation-based learning [12]). The need for a programmed approach is a consequence of the Incompleteness Theorem [13].

17.14 Refrigeration system design example

The purpose served by this example is to demonstrate the utility of the KASER (knowledge amplification by structured expert randomization) [14] in learning to abstract design principles for SoSs and apply those principles to assist design engineers. The refrigeration example was chosen here for the sake of clarity, although it should be clear that many other examples will do as well.

To begin, let us diagram a simple Carnot-cycle refrigerator, depicted in [Figure 17.4](#). This has the simple predicate representation:

```
Next (Compressor, Heat Exchanger)
Next (Heat Exchanger, Evaporator)
Next (Evaporator, Refrigerator)
Next (Refrigerator, Compressor)
```

This then is a simple case upon which the user will improve. The model base can be segmented for efficiency, maximum cooling, etc., or it can be general as is presumed here. Let us further assume that the design engineer will opt

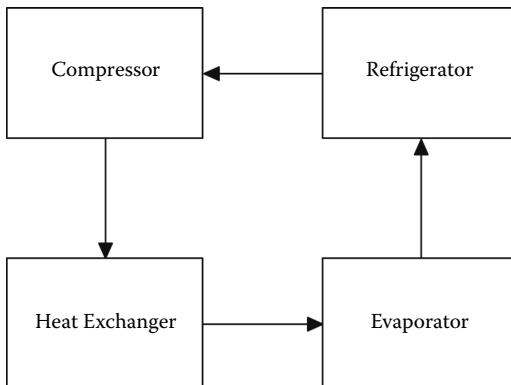


Figure 17.4 A Carnot-cycle refrigerator.

to maximize cooling here and will at first manually transform this design case into a two-stage compressor as [Figure 17.5](#) makes clear.

The predicate representation for this flowchart is:

Next (Compressor, Heat Exchanger)		
Next (Heat Exchanger, Evaporator)	Next (Heat Exchanger, Heat Exchanger)	
Next (Evaporator, Refrigerator)	Next (Evaporator, Freezer)	
Next (Refrigerator, Compressor)	Next (Freezer, Compressor)	
Equal (Refrigerator, Freezer)		

Observe that we have the following case transformation here:

A	A	B
Next (Compressor, Heat Exchanger)	Next (Compressor, Heat Exchanger)	
Next (Heat Exchanger, Evaporator)	Next (Heat Exchanger, Evaporator)	Next (Heat Exchanger, Heat Exchanger)
Next (Evaporator, Refrigerator)	→ Next (Evaporator, Refrigerator)	Next (Evaporator, Freezer)
Next (Refrigerator, Compressor)	Next (Refrigerator, Compressor)	Next (Freezer, Compressor)
		Equal (Refrigerator, Freezer)

This case can be generalized into a candidate version space of possible rules for abstract development (e.g., a thermoelectric refrigerator). However, here the right recursive rule, $A \rightarrow A B$ (see above, where A and B are KB segments, or granules) completely captures the design improvement. This immediately suggests $A B B$, $A B B B$, and so on as further candidate improvements. This is a creative design suggestion for a multistage freezer and is viable, subject to the availability of suitable refrigerants etc., which were not incorporated

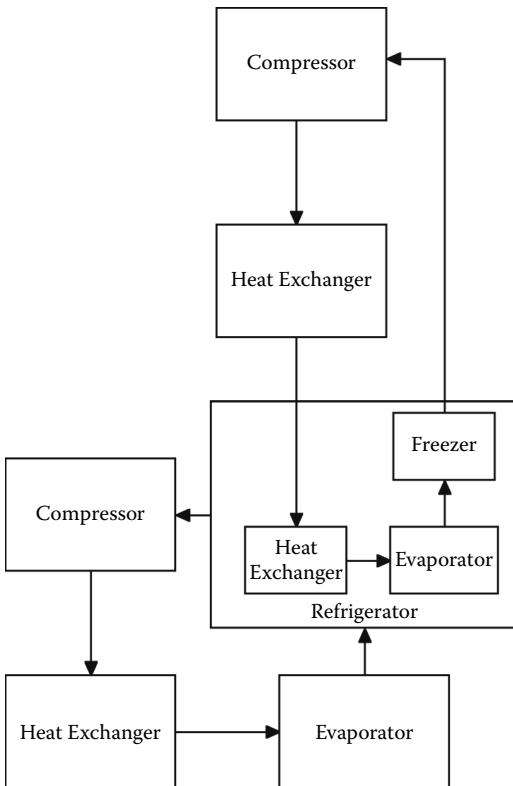


Figure 17.5 Transformation into a two-stage compressor.

into the model. A suitable graphics engine could then illustrate a multistage freezer. No model can incorporate all of practice.

Consider next a simple thermoelectric refrigerator, depicted in [Figure 17.6](#), designed as an improvement to our simple Carnot-cycle refrigerator:

This has the simple predicate representation:

C
Next (Thermopile, Refrigerator)
Next (Thermopile, Heat Exchanger)

Here, the problem is to create a version space of possible maps from A to C as a prelude to the automatic design of a multistage thermoelectric refrigerator. The idea is to automatically port knowledge from one related design to another. The rules in the version space will be automatically constrained by other cases in system memory, which may not be contradicted. In this manner, the system will automatically get smarter with use. At this point, here are two viable maps in the version space, where the second is a generalization of the first:

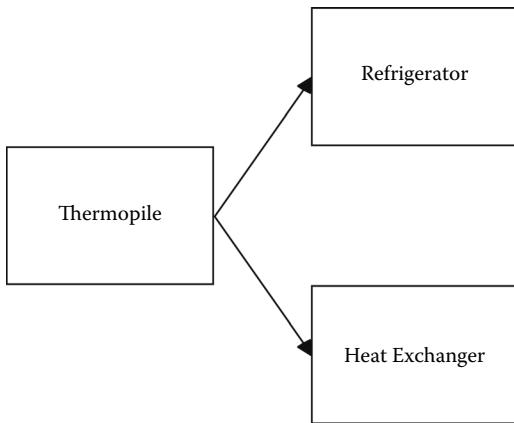


Figure 17.6 A simple thermoelectric refrigerator.

A	C
Next (Compressor, Heat Exchanger)	Next (Thermopile, Refrigerator)
Next (Heat Exchanger, Evaporator)	Next (Thermopile, Heat Exchanger)
Next (Evaporator, Refrigerator)	→
Next (Refrigerator, Compressor)	

A	C
Compressor	→ Thermopile
Evaporator	→ NIL
Next (X, NIL)	→ NIL
Next (NIL, Y)	→ NIL
Equal (Refrigerator, Thermopile) (Thermopile, Refrigerator)	

Now, consider applying this generalization to the design of a multistage thermoelectric refrigerator. That is, $A \rightarrow C B'$:

A	C	B'
Next (Compressor, Heat Exchanger)	Next (Thermopile, Heat Exchanger)	
Next (Heat Exchanger, Evaporator)		Next (Heat Exchanger, Heat Exchanger)
Next (Evaporator, Refrigerator)	→ NIL	
Next (Refrigerator, Compressor)	Next (Refrigerator, Thermopile)	Next (Freezer, Thermopile)
		Equal (Refrigerator, Freezer)

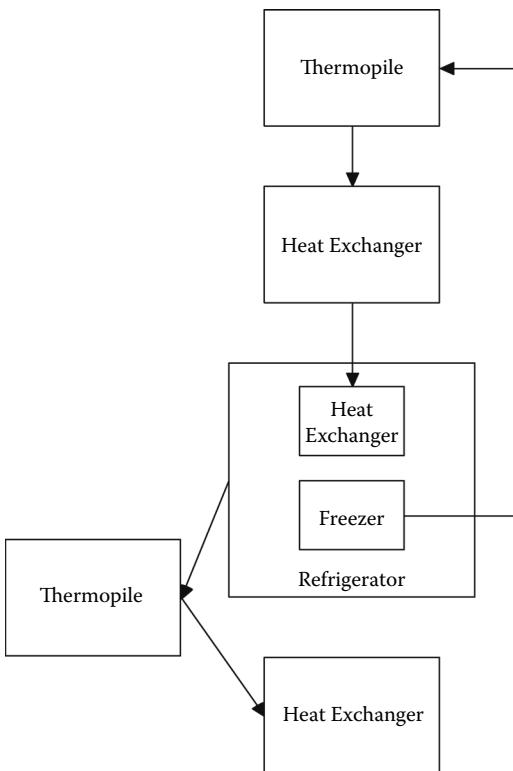


Figure 17.7 A two-stage thermoelectric freezer.

The initial equivalent depiction of this two-stage thermoelectric freezer is depicted in Figure 17.7.

This design is not quite correct, though, due to a random variation. That is, the translation from fluid mechanics to thermoelectrics is not perfectly symmetric. We observe that, while it makes sense to cool a compressed gas in stages to conserve energy, this is not practical to do using thermocouples. Thus, we need to add the domain-specific (context-sensitive) transformation rule (discovered by the KASER algorithm automatically):

$$\{\text{Next (Thermopile, Heat Exchanger), Next (Heat Exchanger, Heat Exchanger)}\} \rightarrow \{\text{Next (Thermopile, Heat Exchanger)}\}.$$

The corresponding flowchart is depicted in [Figure 17.8](#). Notice that this rule captures this essential difference in thermoelectric systems design for broadly applicable reuse (and further specialization).

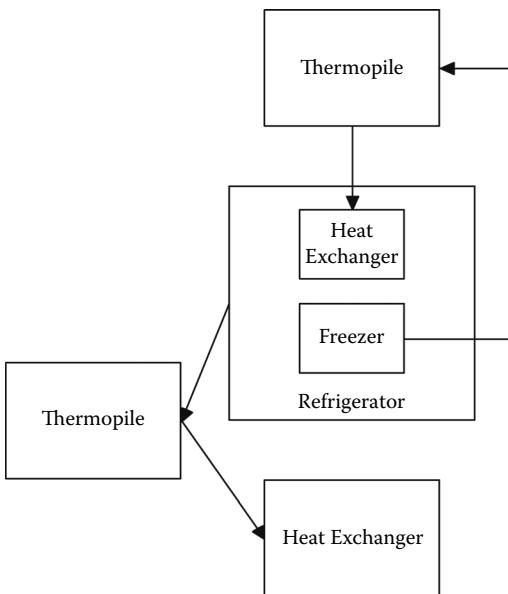


Figure 17.8 A transformatively corrected two-stage thermoelectric freezer.

Notice that this rule would not fire for the case of compressors. If we had designed the thermoelectric refrigerator first and now wanted to transform our solution to a gas refrigerator, then we would have the rule:

{Next (Thermopile, Heat Exchanger)} \rightarrow {Next (Compressor, Heat Exchanger), Next (Heat Exchanger, Evaporator), Next (Evaporator, Refrigerator)}, where {Next (Heat Exchanger, Evaporator)} \rightarrow {Next (Heat Exchanger, Evaporator), Next (Heat Exchanger, Heat Exchanger)}.

Observe that right recursion will not be a problem.

This simple example does not include (or preclude) the use of informative connectives (e.g., a dotted line indicating that the Heat Exchanger and Freezer must not be too close to each other, and the like). Just like the directed arrow translates into the “Next” predicate, the labeled line segment here might be translated into the “Distant” predicate. Furthermore, each nonprimitive box is hierarchically defined. Of course, decision boxes and similar constructs (e.g., to capture concurrency, as in Concurrent (Apply Front Brakes, Apply Rear Brakes)) may augment our block diagrams for use in more complex designs. Also, facilities may eventually be needed to support development by simultaneous users. Moreover, so far all generalizations have been made in the first-order predicate calculus through the simulated application of the KASER language translation algorithm [14]. This algorithm makes it unnecessary to translate into a second-order, or predicate calculus on account of

its iterative context-sensitive rewrite rules, which unlike rules in the predicate calculus are acquired bottom-up. Finally, fuzziness in system design is captured by an allowance for nondeterministic (probabilistic) rewrite rules. For example, the predicate relation, Equal (Refrigerator, Freezer) can induce nondeterminism into the design process.

17.15 Conclusion

Kurt Gödel stunned the mathematical world with his Incompleteness Theorem in 1931 [13]. His proof demonstrated that Hilbert's plan for a completely systematic mathematics cannot be fulfilled. That is, his theorem proved that it would not be possible to formalize most of science and the arts. Instead, processes of discovery would inexorably be tied to chance and lie outside the realm of mechanical realization. However, the Japanese Fifth Generation project flew directly in opposition to this! Its failure could have been foreseen. Basically, the predicate calculus would need to be formulated on a heuristic basis so as to respect the Incompleteness Theorem. In particular, a heuristic calculus for the Army's Future Combat System (FCS) [9], [15] was described using a cost-benefit approach.

The heuristic calculus takes up where the Japanese Fifth Generation project left off [7]. It holds the promise of enabling the development of complex software solutions for intelligent applications (i.e., those inherently based on effective search), such as those pertaining to the U.S. Army's FCS (RoboScout, Oberwatch, et al.). Unlike the predicate calculus, the heuristic calculus provides for computational creativity in keeping with the dictates of the Incompleteness Theorem. The heuristic calculus can serve where the application domain allows for solutions to be hill-climbed and for which a simulation and/or model can be cost-effectively constructed. It represents a novel approach to intelligent software design. In summary, the heuristic calculus serves to weld the human and the machine in a symbiotic fashion. Human and computational resources alike are fully utilized in a most cost-effective and synergistic manner.

Acknowledgments

The material, which served as the basis for this chapter was prepared while the author was at Ft. Monmouth, NJ. The author wishes to gratefully acknowledge the assistance provided by the following individuals in alphabetical order: Rich Borman, Simon Heimfeld, Dr. David Hislop, Robert Lawrence, Stephen Levy, Dr. John Niemela, Steve Oshel, Doug Peters, and Randolph Reitmeyer. This work was produced by a U.S. government employee as part of his official duties and is not subject to copyright. It is approved for public release with an unlimited distribution.

References

1. Steels, L. and R. Brooks, eds. 1995. *The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents*. Lawrence Erlbaum Assoc., Mahwah, NJ.
2. Zadeh, L. A. 1996. Fuzzy logic = computing with words. *IEEE Trans. Fuzzy Syst.* 4(2):103–111.
3. Rubin, S. H. 1999. Computing with words. *IEEE Trans. Syst. Man, Cybern.* 29(4):518–524.
4. Pal, S. K., T. S. Dillon, and D. S. Yeung, eds. 2000. *Soft Computing in Case-Based Reasoning*. Springer-Verlag, London.
5. Koza, J. R., M. A. Keane, and M. J. Streeter. 2003. Evolving inventions. *Scientific American* 288(2):52–59.
6. Tristram, C. 2003. Supercomputing resurrected. *MIT Technology Review* 106(1):52–60.
7. Feigenbaum, E. A. and P. McCorduck. 1983. *The Fifth Generation*. Addison-Wesley Publishing Co., Reading, MA.
8. Chaitin, G. J. 1975. Randomness and mathematical proof. *Scientific American* 232(5):47–52.
9. Pronti, J., S. Molnar, D. Wilson, et al. 2002. Future Combat System (FCS) C2 Architecture Study. *DARPA Interim Report*, Document No. 008, Jan. 2002.
10. Hofstadter, D. R. 1979. *Gödel, Escher, Bach: An Eternal Golden Braid*. Basic Books Inc., New York.
11. Nilsson, N. J. 1980. *Principles of Artificial Intelligence*. Morgan Kaufmann Publishers Inc., Mountain View, CA.
12. Minton, S. 1988. *Learning Search Control Knowledge: An Explanation Based Approach*. Kluwer International Series in Engineering and Computer Science, vol. 61, Kluwer Academic Publishers, New York.
13. Uspenskii, V. A. 1987. *Gödel's Incompleteness Theorem*, Translated from Russian. Ves Mir Publishers, Moscow.
14. Rubin, S. H., S. N. J. Murthy, M. H. Smith, and L. Trajkovic. 2004. KASER: knowledge amplification by structured expert randomization. *IEEE Transactions on Systems, Man, and Cybernetics—Part B: Cybernetics* 34(6):2317–2329.
15. Ackerman, R. K. 2002. Army builds future combat systems around information technologies. *Signal* 57(3):39–42, Nov. 2002.

chapter eighteen

System-of-systems standards

Mark A. Johnson

Contents

18.1	Contemporary standards	451
18.1.1	Definitions	452
18.1.2	Usage	452
18.1.3	Typical contents	453
18.1.4	Evolution	454
18.1.5	Development	454
18.1.6	Application	455
18.1.7	Management	456
18.2	System of systems standards	457
18.2.1	System of systems-specific considerations	458
18.2.2	Pathfinder examples of SoS initiatives and their use of standards	459
18.2.3	Potential system of systems standards	460
18.3	Final comments	460
	References	461

18.1 Contemporary standards

Almost each arena of human activity uses recognized standards or guidelines for personal, business, government, or other entity use. There exist a multitude of organizations and standards. Visiting the Los Alamos National Laboratory Research Library, in New Mexico, one finds that the library provides information on over 350,000 standards from more than 450 organizations [2]. A “standard” as found in a dictionary has many meanings, all of which may have some relevance to systems of systems, their development, and realization. The following paragraphs provide a look at some definitions of “standard(s)” and some general meanings and specifics concerning their usage, development, evolution, application, and management.

18.1.1 Definitions

“Standard” can be used as a noun or an adjective. As a noun, one definition of a standard is that of an object serving as a banner or emblem to serve as a rallying entity for battle. Another is that of a flag with a heraldic, organizational, or corporate symbol. In many realms of human endeavor it is thought of “as something established by authority, custom, or general consent as a model or example” [3]. A standard can be a safety guideline, norm, yardstick, benchmark, measure, criterion, guide, touchstone, model, pattern, example, or exemplar. A standard may represent a principle, ideal, code of behavior, code of honor; or morals, scruples, or ethics [3,4]. In addition, it may relate to a standard of work (quality, level, grade, caliber, merit) excellence [4].

As an adjective, standard may represent the standard way of doing “it”: normal, usual, typical, stock, common, ordinary, customary, conventional, wonted, established, settled, set, fixed, traditional, and prevailing. Or it may be used to identify a piece of work on a subject as definitive, established, classic, recognized, accepted, authoritative, most reliable, exhaustive [4].

18.1.2 Usage

The term standard has both general and specific meanings within the realm of contemporary standards. In general, a standard represents a document produced by a standards body, organization, or entity. A specific standard represents either a part of a document or the whole document that is required, recommended, or permitted to be used as practices, processes, formats, contents, etc.

There are three principal types or levels of standards and one alternative [5]. They are:

1. Standard;
 - a. “shall” = is required to,
 - i. Standards define mandatory practices, work products, formats, contents, etc.
 - ii. The clauses are considered “pass” or “fail” for measuring compliance.
2. Recommended or Expected Practice;
 - a. “should” = is recommended that:
 - i. Recommended practices contain suggested practices, processes, products, formats, contents, etc.
3. Guidelines;
 - a. “may” = is permitted to—statement of possible course of action,
 - b. “can” = is able to—statement of possibility or capability,
 - i. Guides may (or can),
 1. Define alternatives and discuss tradeoffs for different methods to satisfy a standard.

2. Offer guidelines for implementing.
 3. Recommend tailoring or document “typical” tailoring.
 4. Define strategy for application of related standards.
 5. Map between similar standards (and/or standards bodies).
 6. Translate terminology for different domains.
 7. Provide other supportive information.
4. Alternative terms: used (with, or in place of, the above terms)
 - a. Normative = standard, *shall*
 - b. Informative = not considered actually part of the “requirements”
 - i. Usually front matter and sometimes an introduction is defined as informative.
 - ii. Additionally, annexes, or other sections providing for understanding, may also be identified as informative.
 - c. Some entities also add an intermediate designation, “expected” where:
 - i. Some practice is required.
 - ii. A common solution is defined.
 - iii. An alternative solution may be substituted.

Much of the terminology concerning usage, such as shall, should, or may, have come to be understood to have the meanings associated with their usage from Old English, where “shall” has come to be associated with authority or command and is required, “should” is associated with expediency and ought to be done, and “may” is used to express possibility, opportunity, or permission [6]. The usage and origination of the standard often predicates its contents.

18.1.3 Typical contents

Typical contents of a standard are presented in this section. A given standard may or may not contain each of the following contents. In some standards there may even be more topics and content than is presented. From this perspective there is not an all encompassing “standard” for a standard, although the various standards organizations each use a template of the contents and how they are portrayed and are to be accomplished. Thus, the following content delineation is taken from the content found in one standard developed under the direction of a recognized standards organization and may not reflect the content required or needed by another organization or entity.

An introduction usually begins a standard wherein are contained the development and evolution background, compliance (voluntary or mandatory), document organization, and acknowledgments. Next, the scope of the standard is addressed to include a purpose statement, coverage, limitations, and in some cases, the intended applications. In addition, some standards include a section of “normative references” that identify other standards or documents which are referenced or used as part of the standard. There may also

be a section on “informative standards” used to expand upon a requirement and improve understanding of its application and/or context. Acronyms, terminology, and relevant definitions are given next. The body of the standard is presented next. The body is usually followed by some example applications of the standard to demonstrate key concepts and contexts. As standards evolve, the typical contents of a standard have and continue to evolve.

18.1.4 Evolution

Standards usually evolve from one or more entities developing a way of defining, doing, or characterizing something. These become understood or directed descriptions, methods, or measures of something. These descriptions, methods, or measures are then applied to each something that comes along that is similar to the original something in some way or another. This may lead to a new standard or an extension of the existing one if/when the new something is not similar enough to the old something. This leads to the management of the descriptions, methods, or measures to ensure the correct ones are applied correctly to the old something or revised to accommodate a new something.

The evolution of standards by “harmonization” is what happens when there become several standards from differing standards entities, addressing the same or similar areas, that need to be brought into accord with one another [7]. The evolution of one or more standards may become imperative due to changing organizations, industries, technologies, methods, or national and/or international developments.

18.1.5 Development

Standards are usually developed to eliminate understandings between entities. These entities are typically manufacturers and vendors, users and consumers, governments or laboratories. Standards are either market driven and voluntary or directed within a government or organization. Standards may be particular to a single industry, government, or organization, or may affect many or even all industries, governments, or organizations. Thus some standards or portions of them may be general and may be broadly applied or used, while others may be specific to a given process, product, or activity. Yet, there are steps in the development process that are typically taken to realize a standard, beginning with the agreement or decision that one is needed.

A six-step development process as outlined by the International Organization for Standardization (ISO) is as follows:

- Stage 1: Proposal
- Stage 2: Preparatory
- Stage 3: Committee
- Stage 4: Enquiry

- Stage 5: Approval, and
- Stage 6: Publication [8]

These six steps represent a formal approach and are the minimum necessary to realize a viable standard. For ISO, a new standard must be proposed from within its membership and formally initiated by a relevant technical committee or subcommittee. For organizations or entities that are not so formal and do not require membership to be involved, the standard may be proposed by interested stakeholders within a given field, product area, government, business, or other entity. Yet, crucial to this stage is that the stakeholders agree to the need for a relevant standard and commit to participating in the process of its development.

Once the decision is made to pursue a standard, a body of “experts” chosen by the stakeholders, along with a project lead or chairman, embarks upon collecting and organizing information for the standard as well as drafting an initial version of it. The preparatory stage is where brainstorming and drafting take place to develop the “expert” solution to the problem the stakeholders agreed needed a standard to resolve. The committee stage then becomes where the “expert” working group goes back to the stakeholders with the proposed solution for the standard. This may, and usually does, result in an iterative process where redirections or clarifications from the stakeholders cause the working group to revisit parts of the drafted standard until the solution satisfies the stakeholders. Once the stakeholders are satisfied, the standard is usually approved and then published.

For the more formal organizations, such as the ISO, once the relevant technical committee or subcommittee is in agreement with the draft standard, it is sent out to all ISO organizational entities for review and comment. There are times when this can be important, especially when the standard being drafted may impact other industries, governments, laboratories, users, etc. This will be an important aspect in the development of standards for system of systems, whether or not the standard is being developed under the umbrella of the ISO.

18.1.6 Application

“Standardization has always been about ensuring interoperability: a fundamental objective of all stakeholders be they policy-makers, industrial players or users” [7]. Standards are organizational, market, professional, industry, national, or international. In the future they may also become inter/intra planetary or stellar. However, the reasons for applying standards are as numerous as the number of standards themselves (e.g., each standard is developed for a purpose or reason). An organizational example may be the development of a standard as a guide for members to use when contemplating a computer hardware purchase. Markets may adopt a standard due to the interest in it by the consumers or the availability of the devices upon which it

resides or operates (e.g., VHS versus Betamax). Professionals may instantiate “best practices” from lessons learned over time as standards delineating the accepted method or approach to accomplishing professional activities. An industry may develop and instantiate a standard to ensure interoperability and acceptance of products, processes, or other industrial activities. One current example is the World Wide Web (WWW) domain naming conventions or standard. A nation may implement national standards such as laws and building codes. When a standard is adopted and used by one or more nations or countries, it becomes an international standard. Two important families of standards are the International Standards Organization (ISO) 9000 and 14000. The ISO 9000 family covers quality management, and the 14000 series covers environmental management [9].

ISO 9001, published in 2000, is one of the premier international standards. The standard, which gives the requirements for **quality management systems**, is now firmly established as the globally implemented standard for providing assurance about the ability to satisfy quality requirements and to enhance customer satisfaction in supplier-customer relationships [9]. In the market place, being able to stipulate that your organization or group is certified and registered to be standard compliant is often perceived as having increased credibility. With ISO 9001:2000 certification and registration the perception is that your products are assured to meet a certain level of quality due to the quality of your management. A group or entity receives certification from an accreditation body that is certified as competent to evaluate and issue certifications to other entities. Registration is the recording of the certification.

Some specific application areas in which standards are implemented internationally are food safety, information security, automotive, medical devices, supply chain security, health care, petroleum and gas, and software and system engineering. Due to the fact that some standards are used in the development and deployment of spacecraft and rovers to Mars and other bodies in the solar system, perhaps we can say that the standards used are interplanetary standards. The development, construction, launching, assemble of components, operation, and management of the International Space Station is very dependent upon an array of internationally accepted standards due to the various countries involved in its realization, operation, and management.

18.1.7 Management

Standards management is highly dependent upon the standards organization or other entity responsible for initially developing and/or implementing the standard. As identified in the chapter introduction, from one organizational library that maintains access to standards for its members use, the library identifies over 450 standards organizations managing over 350,000 standards [2]. These standards management organizations are from industry, professions, national, and international entities.

To provide a flavor of the differing standards and standards management organizations which develop and manage them, an abbreviated listing is given in acronym form: AA, AASHTO, AATCC, ABMA, ABS, ACI, AECMA, AES, AFNOR, AGA, AGMA, AIA, AIAA, AIChE, AIIM, AMT, ANS, ANSI, API, AREMA, ARI, ARINC, ASA, ASABE, ASCE, ASHRAE, ASSE-SAFE, ASSE, BPVC, ASME, ASQ, ASSE, ASTM, ATIS, AWS, AWWA, BPVC, BHMA, BOCA, BSI, BSMI, CAA, CEA, CEN, CENELEC, CEPT, CGSB, CIE, CSAA, CSAI, CTI, DELPHII, DELTA, DIN, DIN-ENG, DNV, EC, ECMA, EEMUA, EIA, ECA, ETSI, EUROCAE, FM, FMVSS, FORD, GA, GEIA, GM, GMB, GPA, HOLDEN, IAEA, ICAO, ICBO, ICC, ICEA, IEC, IEEE, IESNA, IETF, IMAPS, INCITS, NAVISTAR, IPC, ISUZU, ISA, ISEA, ISO, ITU, JAA, JAG, DEERE, JSA, JTC1, MODUK, MSS, NACE, NATO, NTS, NEMA, NFPAFLUID, NISO, NIST, NSF, OPEL, OSHA, PAPTAC, PFI, PIA, PPI, RTCA, RWMA, SAE, SBCCI, SCTE, SES, SMACNA, SMPTE, SNZ, SSPC, TAPPI, TIA, UL, and ULC.

Typically, most organizations periodically review standards to ensure they still serve the purpose for which they were originally developed. Changes in processes, technologies, markets, or agreements among nations may drive the need for either revision or withdrawal of the standard. There are also times when a standard is taken over by another standards body or merged with one or more other standards.

“Standards harmonization” is what happens when there become one or more standards from differing standards entities, addressing the same or similar areas of interest, that need to be brought into accord with one another [7]. The culmination of the harmonization process is the production of one or more standards covering the areas of interest. There are times when during “harmonization” another standards organization may take over the management of a given standard. In the development and implementation of standards for a system of systems, it may become a standard practice to perform a harmonization-like process where a standard is developed that provides an umbrella coverage of all of the standards accepted for use in the SoS.

18.2 *System of systems standards*

Engineering and acquisition are the arenas in which most current work on system of systems is being performed. Looking at how standards and standardization are being used in engineering, one runs into the concept of a “universally agreed-upon set of guidelines for interoperability” [7]. These guidelines provide four levels of standardization: compatibility, interchangeability, commonality, and reference [7]. These four levels of standardization are relevant in an SoS environment, since they create compatibility, similarity, measurement, and symbol and ontological standardization. As we mature the various disciplines involved in the development, fielding, and use of a system of systems, we shall need to be able to develop standards to ensure these four levels of the SoS standardization are met. In addition, more levels of interoperability and standardization may become apparent.

Growth of the information technologies, nanotechnologies, open systems, globalization (flat Earth syndrome or world without borders) is and will continue to drive the need for new standards. As these become part of large-scale, complex, systems of systems, the need to update, harmonize, or develop new system of systems standards is inevitable. As far as the definitions, usage, and content are concerned, there will be minimal impact from a system-of-systems perspective. Principal areas in which differences and additions will emerge will be in their evolution, development, application, and management.

18.2.1 System of systems-specific considerations

Standards, as with any other component of the system-of-systems journey, must be considered in the early stages. Where appropriate or adequate standards and/or guidelines are unavailable, new ones should be developed. Due to the desire for adaptability in all areas of system of systems, these should, at a minimum, be developed as standards that are "open" to any entity participating or impacted by the SoS. Adaptability is necessary in a system of systems, since the membership or configuration is or can be dynamic, and the relationships among all of the systems in the SoS may not always be known. The key to enabling the ability to be adaptable is interoperability from semantic, syntactic, and organizational perspectives/considerations [10].

To evolve current standards to those needed for the realization of system of systems, current standards and their progress are used as the starting point, since current standards will probably form the basis of system-of-systems standards. It may be that standards for a given system of systems will be a collage of current standards adopted to enable realization of the given system of systems. This is currently the case for efforts such as the United States Army Future Combat System of Systems or the Global Earth Observation System of Systems (GEOSS).

Development of system-of-systems standards is different from the development of a single organizational, industry, national, or international standard in that many systems, organizations, industries, nations, groups, relationships, and perspectives are involved. In the GEOSS project, a new architecture-centric, model-based systems engineering process developed by Boeing emphasizes concurrent development of the system architecture model and the system specification [11]. It may be that this process is the beginning of the development of a system-of-systems standard concerning SoS engineering processes.

Similar arguments concerning the difference of system-of-systems standards development are valid for their application. Managing SoS standards is probably going to embrace the same processes currently used with contemporary standards. It may be that variations in SoS standards management will be due to the various organizations responsible for the SoS standards being used for the given SoS endeavor. It is also probable that a

system-of-systems standards management standard may eventually evolve and be developed.

18.2.2 *Pathfinder examples of SoS initiatives and their use of standards*

The United States Army is developing the Future Combat System (FCS). The network component of the system of systems enables the various associated Families of Systems (FoS) to operate cohesively. The FCS network consists of five layers to provide seamless data delivery among the SoS. Standards, transport, services, applications, and sensors and platforms are the five layers. The standards layer provides the governance with which the other layers are shaped and formed, thus forming the foundation of the FCS network [12]. The standards categories included are doctrine, spectrum, governance, architecture, engineering, and policy.

The FCS network will conform to Defense Information Systems Agency (DISA) standards that ensure end-to-end interoperability testing and network testing. In addition, the FCS network will conform to the standards documentation to ensure that the net-centric attributes are in place to move into the net-centric environment as part of the service-oriented architecture (SOA) in the Global Information Grid (another SoS development) [12]. Conformance to the DISA standards is a principal cornerstone for ensuring interoperability of future FoS that become part of the FCS.

Global Earth Observation System of Systems (GEOSS) monitors climates, crops, forests, deserts, and their changes/effects and potential impact on humankind. It is a global project encompassing over 60 nations [11]. At the beginning of the GEOSS project, interoperability and standards adoption and methods of accommodation were developed through a standards and interoperability forum. During the development phases of the GEOSS, component, services, and interoperability standards were registered. These became GEOSS-registered standards. The rationale is that, when components conform to the same data descriptions and transport standards and are well defined in the GEOSS standards registry, then interoperability is easily achieved. However, when two or more GEOSS components do not share common standards, or where data, transport, and service definitions are not adequate, special interoperability arrangements are made [13]. Perhaps the disconnects will lead to one or more system-of-systems standards through extension and harmonization of the disparate one being accommodated during the initial stages of the project.

Looking forward at an eventual interplanetary Internet, engineers at the NASA Jet Propulsion Laboratory are working on standards for space communications. These would take into account the large distances and time delays inherent in space communications components and systems. Also, the Internet currently supports “only 4.3 billion unique addresses” and is discussing

launching a new format that would accommodate “340 trillion trillion trillion addresses.” [14]. Developing a planetary, interplanetary, and interstellar communications infrastructure will require the use and application of what is envisioned for the system-of-systems discipline and SoS standards.

18.2.3 Potential system of systems standards

As with contemporary standards, system-of-systems standards will be both generic and specific. Many of them will probably mimic current systems-oriented standards yet incorporate extensions that incorporate system-of-systems perspectives and needs. One need that is important in the systems realm is adaptability. Adaptability will become even more important in the system-of-systems realm due to the uncertainties, technologies, systems, stakeholders, organizations, and other entities that may be a part of or involved with the future system of systems throughout its lifetime and probable evolution.

One of the keys in ensuring the adaptability of a system of systems is interoperability from semantic, syntactic, and organizational considerations. One effort to provide a semantic and syntactic interoperability standard is the development by the United States Department of Defense (DoD), Command, Control, Computers, Communications, Intelligence, Surveillance, and Reconnaissance (C4ISR) organization of the “Levels of Information System Interoperability”[15].

A potential tool and standard for use in system-of-systems management is the DoD Architecture Framework (DoDAF). It uses a series of system or system-of-systems architectural views to define or characterize levels of system or system-of-systems development. A related tool and potential standard is the Object Modeling Groups (OMG), Unified Modeling Language (UML). Developed initially as a software algorithm development tool embracing the object-oriented approaches to analysis, development, and programming, UML has evolved into an often used tool for outlining, depicting, and evaluating processes, interactions among entities, organizational dynamics, and other inter and intra relationships. Additionally, an understanding by many investigators of large-scale, complex, system of systems is that modeling and simulation are going to play a critical role in SoS design, development, application, and management. Thus, standard approaches to modeling and simulating systems, processes, interactions, and other system-of-systems activities are prime areas in which standards for SoS may evolve.

18.3 Final comments

As tools develop and more system of systems initiatives are brought into being, standard approaches to initiating, developing, realizing, and managing a system of systems will be invented and become invested into our culture. At the Second IEEE, International Conference on System of Systems in 2007,

an International Consortium on System of Systems (ICSOS) was proposed and began forming. The consortium consists of partners from academia, industry, national academies of many nations, governments (including military components), and other international partners. The ICSOS is intended to act as a neutral, nonprofit broker in translating the needs of industries, governments, and others to the field of SoS researchers and to foster teaming arrangements from interested parties to respond to SoS challenges. An area already identified as needing attention is the area of SoS standards.

The need for system-of-systems standards is evident from the SoS literature, where definitions and perspectives are presented with great variability. These variations lead to difficulties in advancing and understanding the SoS discipline. Standards are used to facilitate common understandings and approaches by deriving uniform agreements to definitions and approaches. System-of-systems standards will help to unify and advance the SoS discipline for all disciplines requiring interoperability and standardization among disparate systems.

One of the initiatives that will come out of the ICSOS effort is the instantiation of a technical committee for SoS standards and sub/technical committees to address discipline subarea standards. The processes for developing and instantiating SoS standards will follow many of today's processes and approaches with some additional considerations as described in the GEOSS description with its "special arrangements." These considerations shall encompass the inclusion of practically all arenas of human activities that may impact the SoS development, application, and management or vice versa.

References

1. Fung, V., W. Fung, and Y. Wind. 2008. *Competing in a Flat World: Building Enterprises for a Borderless World*. Wharton School Publishing, Upper Saddle River, NJ.
2. IHS Specs and Standards—LANL Research Library. 2007. <http://library.lanl.gov/inforcs/stand/ihs/index.htm>.
3. Merriam-Webster's Collegiate Dictionary. 2004. Merriam-Webster Inc., Springfield, MA.
4. Merriam-Webster's Collegiate Thesaurus. 2004. Merriam-Webster Inc., Springfield, MA.
5. Bowen, G. M. 2007. Systems & Software Engineering Standards, Introduction to Books of Knowledge (BOKs) and Standards. <http://members.aol.com/kai-zensepg/standard.htm>.
6. The Random House College Dictionary. 1975. Random House, Inc. New York.
7. European Telecommunications Standards Institute. Standards, Open Standards and Interoperability. http://www.etsi.org/SOS_Interoperability/context.htm.
8. International Organization for Standardization. 2008. International Organization for Standardization. 2008. The International Organization for Standardization. <http://www.iso.org/iso/home.htm>.
9. International Organization for Standardization. 2008. Management standards. http://www.iso.org/iso/iso_catalogue/management_standards.htm.

10. Office of the Under Secretary of Defense. 2006. *Systems of Systems: Systems Engineering Guide*, Version 0.9, Office of the Undersecretary of Defense (Acquisition, Technology, and Logistics).
11. Pearlman, J. 2006. GEOSS—Global Earth Observation System of Systems, Key-note Presentation, 2006 IEEE International SoS Conference, Los Angeles, CA.
12. <http://www.army.mil/fcs/network.html>.
13. Khalsa, S. J. S. 2007. Report on the GEOSS Interoperability Process Pilot Project, Presentation at the IEEE GEOSS Workshop.
14. http://www.spacemart.com/reports/Internet_preparing_to_go_into_outer_space.htm, 10/17/2007.
15. C4ISR Interoperability Working Group. 1998. Levels of Information System Interoperability. U.S. Department of Defense, Washington DC.