GLOBAL EDITION

System Architecture

Strategy and Product Development for Complex Systems

Edward Crawley **Bruce Cameron** **Daniel Selva**

Foreword by Norman R. Augustine

ALWAYS LEARNING

PEARSON

SYSTEM ARCHITECTURE

Strategy and Product Development for Complex Systems

Global Edition

Edward Crawley Bruce Cameron Daniel Selva

PEARSON

Boston Columbus Indianapolis New York San Francisco Hoboken
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Vice President and Editorial Director, ECS: *Marcia J. Horton*
Executive Editor: *Holly Stark*
Senior Acquisitions Editor, Global Editions: *Priyanka Ahuja*
Field Marketing Manager: *Demetrius Hall*
Senior Product Marketing Manager: *Bram van Kempen*
Marketing Assistant: *Jon Bryant*
Senior Managing Editor: *Scott Disanno*
Global HE Director of Vendor Sourcing and Procurement: *Diane Hynes*
Director of Operations: *Nick Sklitsis*
Operations Specialist: *Maura Zaldivar-Garcia*
Senior Manufacturing Controller, Global Editions: *Trudy Kimber*
Cover Designer: *Lumina Datamatics*
Production Project Manager: *Rose Kernan*
Project Editor, Global Editions: *K.K. Neelakantan*
Program Manager: *Erin Ault*
Manager, Rights and Permissions: *Rachel Youdelman*
Media Production Manager, Global Editions: *Vikram Kumar*
Associate Project Manager, Rights and Permissions: *Timothy Nicholls*
Full-Service Project Management: *George Jacob, Integra*
Cover Image: © *Boskalis*

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-11084-8
ISBN 13: 978-1-292-11084-4

Typeset in 10/12, Times LT Sts by Integra
Printed in Slovakia by Neografia

Contents

Foreword

Norman R. Augustine

A particularly promising trend that has been taking place in healthcare is the marriage of biomedical research with engineering practices. A friend of mine, an engineer, recently described to me a meeting that took place at one of America's most prestigious universities between the faculties of the engineering department and the cardiology department exploring just such opportunities. Having decided to focus on constructing a practicable mechanical human heart, the head of cardiology began his presentation with a description of the properties of the human heart. Almost immediately an engineer interrupted, asking "Does it have to be in your chest? Could it be, say, in your thigh where it would be easier to reach?" No one in the room had ever considered that possibility. Nonetheless, the presentation continued. Soon another interruption occurred; this time it was another engineer asking, "Instead of just one heart could you have three or four small hearts integrated in a distributed system?" No one had thought of that either.

System Architecture, so insightfully presented in this book by three of the field's most highly regarded leaders, is about asking—and—answering just such questions. In my own career I have encountered system architecture questions in fields ranging from engineering to business to government. When established practices of the field of system architecture are applied, far superior outcomes seem to result.

Applying such practices has not always been the case. Early in my career I recall asking various of my colleagues who were working "together" on a guided missile program why they had chosen a particular design approach for their specific element of the product. One replied, "Because it is the lowest weight." Another assured me that his part would have the lowest radar cross-section. Still another answered because her component would be less costly. And yet another had focused on minimizing volume. And so it went.

What was missing? The answer is a *system architect*.

This shortcoming is too often encountered, usually in more subtle ways. Consider the case of the Near-Sonic Transport aircraft that was in the early stages of development a few years ago. A marketing survey had indicated that airline passengers want to get to their destinations faster. To an aerodynamicist (my own early field), if one wishes to avoid the penalties of supersonic flight, that translates into more closely approaching Mach One, creeping up on the drag curve into a regime wherein fuel consumption abruptly increases. This was, in fact, the underlying concept of the Near-Sonic Transport.

But when viewed from a system architecture perspective, the appropriate question is not how to fly faster; rather, it is how to minimize the time to get from one's home, to the airport, check-in, pass through security, board the aircraft, fly, collect baggage and travel to one's final destination. Placed in this context, an even more fundamental question arises: "How much will a passenger pay to save five or ten minutes of flying time?" The answer turns out to be, "not much"—and the Near-Sonic Transport aircraft thus met its early, and deserved, demise. There are clearly better

---

Norman R. Augustine has served in industry as chairman and CEO of Lockheed Martin Corporation, in government as Under Secretary of the Army, in academia as a member of the engineering faculty of Princeton University and as a trustee of MIT, Princeton, and Johns Hopkins and as a regent of the University System of Maryland's 12 institutions.

opportunities in which to invest if one's objective is to help passengers reach their *destinations* more rapidly. The failing in this case was to not recognize that one was dealing with a problem of *system architecture*... not simply a problem of aerodynamics and aircraft design.

My own definition of a "system" evolved over years of experience. It is "two or more elements that interact with one another." The authors of this book wisely add that the resultant functionality must exceed the sum of functionalities of the individual elements. Thus simple in concept, the complexity of most real-world systems is enormous. In fact, the equation describing the number of possible states a system of several elements (that interact in the simplest of all manners) has been aptly named, "The Monster!" And when a system includes humans, as many systems do, the challenge of system architecting becomes all the more immense due to the presence of unpredictability. But these are the kind of systems that one encounters, and are the kind of systems that the authors show how to deconstruct and address.

One such system that I had the occasion to analyze concerned provisioning the (human occupied) U.S. station at the Earth's South Pole. Setting the specific objective of the evaluation in itself required care... as is often the case. Was it to minimize expected cost? Or to minimize worst-case cost in the face of uncertainty, say, due to weather? Or perhaps to minimize "regret"—that is, when supplies are not delivered at all? Or...?

In the case of this particular system there are a number of elements that must interface with one-another: cargo ships, ice breakers, aircraft of various types, ice piers for off-loading, storage facilities, traverse vehicles, communications... and, underlying all decisions, was the ever-present danger of single-point failure modes creeping into the architecture.

In the business world one of the more complex problems faced in my career was whether—and how—all or major parts of seventeen different companies could be combined to create the Lockheed Martin Corporation. Each of the "elements" had its strengths and its weaknesses; each involved large numbers of humans, each with their own goals, capabilities, and limitations; and critical to the decision, the whole had to have significantly greater functionality than the sum of the parts. If the latter were not the case, there would be no reason to pay the financial premium that is implicit in most mergers and acquisitions.

Sadly, in engaging complex questions of this type there is no simple mathematical formula that will reveal the "right" answer. However, the discipline of systems thinking proves to be an invaluable tool in assessing exposure, opportunities, parametric sensitivities, and more. In the above case, most people judge that the answer came out "right"—which, incidentally, contrasts with nearly 80 percent of similar undertakings.

One of the authors of this book and I, along with a group of colleagues, had the occasion to propose to the President of the United States a human spaceflight plan for America for the next few decades. In this instance perhaps the most difficult challenge was to define a useful *mission*, as opposed to the (non-trivial) task of defining an appropriate hardware configuration. Fortunately, such issues are amenable to solution through system thinking.

As the authors point out in the material that follows, the process of establishing the architecture of systems is both a science and an art. But, as is so elegantly portrayed herein, there is a Darwinian phenomenon wherein systems embodying the mistakes of the past do not survive; whereas those that embody sound architectures generally do survive—and even prosper.

That, of course, is what architecting complex systems is all about.

Preface

We wrote this book to capture a powerful idea. The idea of the "architecture of a system" is growing in recognition. It appears in diverse fields including the architecture of a power grid or the architecture of a mobile payment system. It connotes the DNA of the system, and the basis for competitive advantage. There are over 100,000 professionals with the title system architect today, and many more practicing the role of the architect under different titles.

Powerful ideas often have nebulous boundaries. We observed that many of our co-workers, clients, students had a shared recognition of system architecture issues, but used the term in very different scopes. The term is often used to differentiate between existing systems, as in "the architecture of these two mountain bikes is different."

What exactly constitutes the architecture of a system is often a subject of great debate. In some fields, the term is used for a singular decision that differentiates two types of systems at a high level, as in "packet-switched architecture" vs. "circuit-switched architecture." In other fields, the term is used to describe a whole implementation, save for some smaller details, as in "our software as a service architecture."

Our goal was to capture the power of the idea of architecture, and to sharpen the boundaries. Much of the power of idea originates with the potential to trade among several architectures early, to look downstream and identify which constraints and opportunities will be central to value. It isn't possible to trade among early ideas if the architecture encompasses all details, nor is it a meaningful exercise if important drivers of value are missing.

We wrote this book to build on the idea that the architect is a specialist, not a generalist, as proposed by Eberhardt Rechtin. Our intent is to showcase the analysis and methodologies of system architecture, and to develop the 'science' of system architecture. This text is less prescriptive in places than the discipline of product design, as the systems tackled are more complex. Where the product development community has a stronger focus on design, our focus centers more on emergence—the magic of functions coming together to produce a coherent whole.

We've imbued this book with our past experience. We've been fortunate to be involved in the early development of a number of complex systems in communications, transportation, mobile advertising, finance, robotics, and medical devices, ranging in complexity from farm equipment to the International Space Station.

Additionally, we have included case studies from the experience of other system architects, in disciplines ranging from hybrid cars to commercial aircraft. Our intent was that this book can only advance system architecture if it works from challenges faced by system architects today.

We wrote this book for two core audiences—professional architects and engineering students. System architecture as an idea grew out of practitioners' wisdom and attempts to codify the challenges of developing new architecture. One core audience is senior professionals who are faced with architectural decisions. The field encompasses a variety of professionals in senior technical and managerial roles in technical industries—software, electronics, industrial goods, aerospace, automotive, and consumer goods.

This book is also focused on engineering students as a core audience. This text grew out of the graduate course we have taught at MIT for the past 15 years, where we've been fortunate to educate many leaders in the private sector and government. The lens of architecture helps us understand how a system operates today, but moreover, we believe that it is a necessary competency to learn in the management of technical organizations.

This page intentionally left blank

Acknowledgments

We'd like to thank the many people that made this book possible. First and foremost, our thanks to Bill Simmons, Vic Tang, Steve Imrich, Carlos Gorbea, and Peter Davison who contributed sections from their expertise, and who all provided comments on early drafts. We're indebted to Norm Augustine, who in addition to contributing the foreword, shaped our thinking on the topic.

Our reviewers Chris Magee, Warren Seering, Eun Suk Suh, Carlos Morales, Michael Yukish, and Ernst Fricke helped us deliver crisp messages and helped identify where we had missed key ideas. We also received a number of anonymous reviews, whose feedback improved the book. Dov Dori has been an invaluable partner as the developer of the OPM.

Pat Hale supported the development of the curriculum at MIT, and provided feedback on an early draft. The 63 students of the MIT System Design and Management Class of 2011 reviewed each chapter in detail and provided mountains of suggestions. In particular, our thanks to Erik Garcia, Marwan Hussein, Allen Donnelly, Greg Wilmer, Matt Strother, David Petrucci, Suzanne Livingstone, Michael Livingstone, and Kevin Somerville. Ellen Finnie Duranceau at MIT Libraries helped us choose a publisher wisely.

Our graduate students over the years have helped shape the book's content – much of their work appears here in one form or another. In addition to those mentioned above, we'd like to thank Morgan Dwyer, Marc Sanchez, Jonathan Battat, Ben Koo, Andreas Hein, and Ryan Boas.

We would like to thank Eun Suk Suh for his contributions to the Global Edition as well.

The staff at Pearson made our book a reality—Holly Stark, Rose Kernan, Erin Ault, Scott Disanno, and Bram van Kempen. Thanks for all your hard work.

Finally, to our wives Ana, Tess, and Karen, thanks for your patience as we labored on weekends and during vacations, enduring the risk that this project become a "forever book."

Edward Crawley Bruce Cameron Daniel Selva

Cambridge, MA

About the Authors

Edward F. Crawley

Edward Crawley is the President of the Skolkovo Institute of Science and Technology (Skoltech) in Moscow, Russia, and a Professor of Aeronautics and Astronautics and Engineering Systems at MIT. He received an S.B. and an S.M. in Aeronautics and Astronautics and an Sc.D. in aerospace structures, all from MIT.

From 1996 to 2003, he was head of the Department of Aeronautics and Astronautics at MIT. He has served as founding co-director of an international collaboration on the reform of engineering education and was the lead author of *Rethinking Engineering Education: The CDIO Approach*. From 2003 to 2006, he was the Executive Director of the Cambridge-MIT Institute, a joint venture with Cambridge University funded by the British government and industry; the Institute's mission was to understand and generalize how universities can act effectively as engines of innovation and economic growth.

Dr. Crawley has founded a number of companies. ACX, a product development and manufacturing firm; BioScale, a company that develops biomolecular detectors; Dataxu, a company in Internet advertising placement; and Ekotrope, a company that supplies energy portfolio analysis to businesses. From 2003 to 2012, he served on the Board of Directors of Orbital Sciences Corporation (ORB).

Professor Crawley is a Fellow of the AIAA (American Institute of Aeronautics and Astronautics) and Royal Aeronautical Society (UK) and a member of the Royal Swedish Academy of Engineering Science, the Royal Academy of Engineering (UK), the Chinese Academy of Engineering, and the National Academy of Engineering (US).

Bruce G. Cameron

Bruce Cameron is the founder of Technology Strategy Partners (TSP), a consulting firm, and the Director of the System Architecture Lab at MIT. Dr. Cameron received his undergraduate degree from the University of Toronto, and graduate degrees from MIT.

As a Partner at TSP, Dr. Cameron consults on system architecture, product development, technology strategy, and investment evaluation. He has worked with more than 60 Fortune 500 firms in high tech, aerospace, transportation, and consumer goods, including BP, Dell, Nokia, Caterpillar, AMGEN, Verizon, and NASA.

Dr. Cameron teaches system architecture and technology strategy at the Sloan School of Management and in the School of Engineering at MIT. Previously at MIT, Dr. Cameron ran the MIT Commonality Study, which comprised over 30 firms spanning 8 years.

Previously, Dr. Cameron worked in high tech and banking, where he built advanced analytics for managing complex development programs. Earlier in his career, he was a system engineer at MDA Space Systems, and has built hardware currently in orbit. He is a past board member of the University of Toronto.

Daniel Selva

Daniel Selva is an Assistant Professor in Mechanical and Aerospace Engineering at Cornell. He has degrees in electrical engineering and aeronautical engineering from Polytechnic University of Catalonia (UPC), Supaero, and MIT.

Professor Selva's research focuses on applications of system architecture, knowledge engineering, and machine learning tools to early design activities. His work has been applied to the NASA Earth Science Decadal Survey, the Iridium GeoScan Program, and the NASA Tracking and Data Relay Satellite System (TDRSS), where he developed architectural analysis in support of system architects and executives. He is the recipient of Best Paper and Hottest Article awards.

Between 2004 and 2008, he worked for Arianespace in Kourou, French Guiana, as a member of the Ariane 5 Launch team, specializing in the On Board Data Handling, and Guidance, Navigation and Control. He has previously worked for Cambrian Innovation in the development of novel bioelectromechanical systems for use on orbit, and at Hewlett Packard on the monitoring of banking networks. He is a member of the Board of Advisors for NuOrion Partners, a wealth management firm.

This page intentionally left blank

Part 1
System Thinking

Part 1: System Thinking focuses on the opportunities presented in system architecture, namely, the opportunity to articulate the key decisions that define a system and to choose an architecture to match complex challenges.

Chapter 1: Introduction to System Architecture presents the idea of architecture with examples, identifies good architecture, and outlines the book. *Chapter 2: System Thinking* assembles the ideas necessary for system analysis. *Chapter 3: Thinking about Complex Systems* identifies the constituent modes of thinking we will use to analyze system architecture.

Chapter 1
Introduction to System Architecture

Architecture of Complex Systems

In June 1962, NASA made the decision to use a dedicated capsule to descend to the surface of the Moon from lunar orbit, rather than to descend to the surface with the Command/Service Module used to bring astronauts to lunar orbit. This decision implied that the dedicated capsule, later named the Lunar Module, would have to rendezvous in lunar orbit with their ride home and support a crew transfer between vehicles.

This decision was made in the first year of the Apollo program, seven years before the maneuver would be executed in lunar orbit. It was made before the majority of program staff was hired and before the design contracts were awarded. Yet the decision was formative; it eliminated many possible designs and gave the design teams a starting point. It guided the work of hundreds of thousands of engineers and an investment that in 1968 exceeded 4% of federal outlays.

We conceive, design, implement, and operate complex and sometimes unprecedented systems. The largest container ship today carries 18,000 containers, up from 480 containers in 1950. [1], [2] Cars built today routinely have 70 processors scattered through the vehicle, connected by as many as five separate buses running at 1 Mbit/s [3]—a far cry from early electronics buses used to communicate fuel injection at a mere 160 bit/s. Oil platforms costing $200 to 800 million [4] are developed and produced almost routinely; 39 were delivered between 2003 and 2009. [5]

These systems are not merely large and complex. They are sometimes configurable for each customer and are often very costly to deliver. Customers of consumer products expect unprecedented levels of customization and configurability. For example, BMW calculated that it offered 1.5 billion potential configurations to its customers in 2004. [6] Some complex systems are very costly to deliver. Norm Augustine points out that the unit cost of a fighter aircraft rose exponentially from 1910 through 1980, predicting that in 2053 the entire U.S. defense budget would procure exactly one aircraft. [7] Interestingly, Augustine's prediction has held up well for 30 years: In 2010 an F-22 raptor cost $160 million, or $350 million if the development costs are included. [8]

The Advantages of Good Architecture

Do these complex systems meet stakeholder needs and deliver value? Do they integrate easily, evolve flexibly, and operate simply and reliably?

Well architected systems do!

FIGURE 1.1 Complex systems: The heavy-lift ship MV *Blue Marlin* transporting the 36,000 metric ton drilling platform SSV Victoria. (Source: Dockwise/Rex Features/Associated Press)

The simplest notion of architecture we will use is that *architecture* is an abstract description of the entities of a system and the relationship between those entities. In systems built by humans, this architecture can be represented as a set of decisions.

The premise of this text is that our systems are more likely to be successful if we are careful about identifying and making the decisions that establish the architecture of a system. This text is an attempt to encode experience and analysis about early system decisions and to recognize that these choices share common themes. Over the past 30 years, analysis and computational effort have opened a broad tradespace of options, and in many areas, that tradespace grew faster than our ability to understand it. The field of system architecture grew out of practitioners' attempts to capture expert wisdom from past designs and to structure a broader understanding of potential future designs.

The market context in which our products and systems compete does not offer any comfort. Consider Boeing's decision to "bet the company" on the development of the 787 aircraft and the associated composite technology. Boeing is half of a global duopoly for large passenger aircraft, yet in its core business, rather than spreading risk across many small programs, the firm turns on a single product's emergent success or failure. The global market for mobile devices is larger and more competitive still. Although it can be argued that the product risk is more diversified (that is, an individual product development investment is a smaller fraction of firm revenues) in the mobile sector, witness the declines of former giants BlackBerry and Ericsson. To capture market share, systems must innovate on the product offering, incorporate novel technologies, and address multiple markets. To compete on tight margins, they must be designed to optimize manufacturing cost, delivered through multi-tiered supply chains. We will argue that good architectural decisions made by firms can create competitive advantage in difficult markets, but bad decisions can hobble large developments from the outset.

Every system built by humans has an architecture. Products such as mobile phone software, cars, and semiconductor capital equipment are defined by a few key decisions that are made early in each program's lifecycle. For example, early decisions in automotive development, such as the mounting of the engine, drive a host of downstream decisions. Choosing to mount an engine transversely in a car has implications for the modularization of the engine, gearbox, and drivetrain, as well as for the suspension and the passenger compartment. The architecture of a system conveys a great deal about how the product is organized.

In the design of complex systems, many of these early architectural decisions are made without full knowledge of the system's eventual scope. These early decisions have enormous impact on the eventual design. They constrain the envelope of performance, they restrict potential manufacturing sites, they make it possible or impossible for suppliers to capture after-market revenue share, and so forth. As an example of gathering downstream information for upstream consumption, the width of John Deere's crop sprayers is constrained to be less than the column separation at the manufacturing site. In this case the width constraint is obvious to the development team and was not uncertain or hidden, but it is one of the main variables in the productivity equation for a crop sprayer.

The central assertion of this text is that these early decisions can be analyzed and treated. Despite uncertainty around scope, even without knowing the detailed design of components, the architecture of the system merits scrutiny. Architecting a system is a soft process, a composite of science and art; we harbor no fantasies that this can or should be a linear process that results in an optimal solution. Rather, we wrote this text to bring together what we've learned about the core ideas and practices that compose system architecture. Our central assertion is that structured creativity is better than unstructured creativity.

This focus on decisions enables system architects to directly trade the choices for each decision, rather than the underlying designs they represent, thus encouraging broader concept evaluation. At the same time, this decision language enables system architects to order decisions according to their leverage on the system performance, in recognition that system architectures are rarely chosen in one fell swoop; rather, they are iteratively defined by a series of choices.

The failed National Polar-Orbiting Environmental Satellite System (NPOESS) is an exemplar of architectural decisions handicapping a system. NPOESS[1] was created in 1994 from the merger of two existing operational weather satellite programs, one civilian (weather prediction) and one military (weather and cloud cover imagery). The rationale for the merger was not ill-founded; these two systems collecting related data presented a $1.3 billion cost consolidation opportunity. [9] Early in the merged program, a decision was made to include the superset of instruments capability from both historical programs. For example, the VIIRS (Visible Infrared Image Radiometer Suite) instrument was expected to combine the capabilities of three historical instruments.

The assumption underlying the program was that the functional complexity of the merged program would scale linearly with the sum of the two historical programs. This might have held, had the program derived needs and concepts from the heritage instruments. However, a second decision to list new functions independent of the system concept trapped the architectural performance in an

[1] The prevalence of challenges with government programs cited here reflects a bias: We have more information about government programs than about private programs. Our intent is to learn from the challenges, not to comment on public vs. private.

unreachable corner of its envelope. For example, the VIIRS instrument was to accomplish the tasks of three instruments with less mass and volume than a single historical instrument.

A series of early architectural decisions placed NPOESS on a long and troubled development path, attempting to create detailed designs that ignored fundamental system tensions. Further, a failure to appoint a system architect responsible for managing these trades during the early years of the program foreshadowed challenges to come. The program was canceled in 2010, $8.5 billion *over* the original $6.5 billion estimate. [10]

This text is not a formula or a manual for product development. Success is not assured. Experience suggests that getting the architecture wrong will sink the ship but that getting it "right" merely creates a platform on which the execution of the product can either flourish or flounder.

There are many aspects of this text that are applicable to all systems, whether built by humans, evolved by society, or naturally evolved. The analysis of architecture can be applied to built or evolved systems. For example, brain researchers are trying to unfold the architecture of the brain, urban planners deal with the architecture of cities, and political and other social scientists strive to understand the architecture of government and society. But we will focus predominantly on built systems.

Learning Objectives

This is a text on how to think, not what to think. Our intent is to help the reader develop a way to think about and create system architecture, not to provide a set of procedures. Experience suggests that the best architects have a remarkably common understanding of architecture and its methods, but the content they work with and the context in which they work vary widely.

This text aims to help system architects to *structure and lead* the early, conceptual phases of the system development process, and to *support* the process throughout its development, deployment, operation, and evolution.

To these ends, this text provides guidance to help architects:

- Use system thinking in a product context and a system context
- Analyze and critique the architecture of existing systems
- Identify architectural decisions, and differentiate between architectural and non-architectural decisions
- Create the architecture of new or improved systems, and produce the deliverables of the architect
- Place the architecture in the context of value and competitive advantage for the product and the firm
- Drive the ambiguity from the upstream process by defining the context and boundaries of the system, interpreting needs, setting goals, and defining the externally delivered functions
- Create the concept for the system, consisting of internal function and form, while thinking holistically and out of the box when necessary
- Manage the evolution of system complexity and provide for future uncertainty so that goals are met and functions are delivered, while the system remains comprehensible to all during its design, implementation, operation, and evolution
- Challenge and critically evaluate current modes of architecting

- Identify the value of architecting, analyze the existing product development process of a firm, and locate the role of architecting in the product development process
- Develop the guiding principles for successful architecting

To accomplish these objectives, we present the principles, methods, and tools of system architecture. *Principles* are the underlying and long-enduring fundamentals that are always (or nearly always) valid. *Methods* are the ways of organizing approaches and tasks to achieve a concrete end; they should be solidly grounded on principles, and they are usually or often applicable. *Tools* are the contemporary ways to facilitate process; they are applicable sometimes.

One of our stated goals is for readers to develop their own principles of system architecture as they progress through the text. The architect should base decisions, methods, and tools on these principles.

> *"Principles are general rules and guidelines, intended to be enduring and seldom amended, that inform and support the way in which an organization sets about fulfilling its mission. In their turn, principles may be just one element in a structured set of ideas that collectively define and guide the organization, from values through to actions and results."*
>
> U.S. Air Force in establishing its "Headquarters Air Force Principles for Information Management," June 29, 1998

> *"Principles become modified in practice by facts."*
>
> James Fenimore Cooper, *The American Democrat*, 1838, Ch. 29

We have scattered our own principles throughout this text, but we encourage you to develop your own principles as you reflect on your own experience.

Organization of the Text

This text is organized into four parts.

Part 1: System Thinking (Chapters 1 to 3) introduces the principles of system thinking and then outlines the tools for managing complexity. These principles and tools are echoed through the remainder of the text. The notions are expressed in terms of running examples: an amplifier circuit, the circulatory system, a design team, and the solar system.

Part 2: Analysis of System Architecture (Chapters 4 to 8) is focused on the analysis of architecture. We provide an in-depth exploration of form in an effort to separate it from function, and then we deconstruct function. We introduce the ideas of solution-neutral function and concept, and we analyze the architecture of existing simple systems. Analysis can be applied to any system—both to those intentionally built by humans and to those that evolve, such as organizations, cities, or the brain. In many sections of Part 2, we begin with very simple systems. This is not intended as an insult to the reader's intelligence. Rather, we chose for analysis those systems that can be completely understood in their constituent parts, in order to hone the methods that we later scale up to complex systems. Working with simple systems eliminates the concern that the

product cannot be treated as a system because it is impossible to comprehend all of its constituent parts at one time.

Part 3: Creating System Architecture (Chapters 9 to 13) is focused on the creation of architecture through decision making. It traces the forward process of identifying needs through to choosing an architecture. Whereas Part 2 works backwards from architecture to solution-neutral function, Part 3 deals directly with the ambiguity of the upstream process of goal setting, when no legacy architecture is available. Part 3 is organized around three ideas: reducing ambiguity, applying creativity, and managing complexity.

Part 4: Architecture as Decisions (Chapters 14 to 16) explores the potential of a variety of computational methods and tools to help the architect reason through decisions. Parts 1 to 3 are deliberately focused on the architect as a decision maker. We layer analysis and frameworks on top of the domain expertise of the architect, but the architect performs the integration among the layers, weighing priorities and determining salience. Part 4 explores the idea of encoding architectural decisions as parameters in a model that attempts to capture the salient pieces of many layers or attributes. We will show that there are applications for which the complexity of the architecting problem may be usefully condensed in a model, but it is important to remember that no model can replace the architect—accordingly, we emphasize decision support. In our experience, this decision representation serves as a useful mental model for the tasks of architecting.

References

[1] "Economies of Scale Made Steel," *Economist,* 2011. http://ww3.economist.com/node/21538156

[2] http://www.maersk.com/innovation/leadingthroughinnovation/pages/buildingtheworldsbiggestship.aspx

[3] "Comparison of Event-Triggered and Time-Triggered Concepts with Regard to Distributed Control Systems," A. Albert, Robert Bosch GmbH Embedded World, 2004, Nürnberg.

[4] J.E. Bailey and C.W. Sullivan. 2009–2012, *Offshore Drilling Monthly* (Houston, TX: Jefferies and Company).

[5] "US Gulf Oil Profits Lure $16 Billion More Rigs by 2015," Bloomberg, 2013. http://www.bloomberg.com/news/2013-07-16/u-s-gulf-oil-profits-lure-16-billion-more-rigs-by-2015.html)

[6] E. Fricke and A.P. Schulz, "Design for Changeability (DfC): Principles to Enable Changes in Systems throughout Their Entire Lifecycle," *Systems Engineering* 8, no. 4 (2005).

[7] Norman R. Augustine, *Augustine's Laws,* AIAA, 1997.

[8] "The Cost of Weapons—Defense Spending in a Time of Austerity," *Economist,* 2010. http://www.economist.com/node/16886851

[9] D.A. Powner, "Polar-Orbiting Environmental Satellites: Agencies Must Act Quickly to Address Risks That Jeopardize the Continuity of Weather and Climate," Washington, D.C.: United States Government Accountability Office, 2010. Report No.: GAO-10-558.

[10] D.A. Powner, "Environmental Satellites: Polar-Orbiting Satellite Acquisition Faces Delays; Decisions Needed on Whether and How to Ensure Climate Data Continuity," Washington, D.C.: United States Government Accountability Office, 2008. Report No.: GAO-08-518.

Chapter 2
System Thinking

2.1 Introduction

System thinking is, quite simply, thinking about a question, circumstance, or problem explicitly as a system—a set of interrelated entities. System thinking is *not* thinking systematically. The objective of this chapter is to provide an overview and introduction to systems and system thinking.

System thinking can be used in a number of ways: to understand the behavior or performance of an existing system; to imagine what might be if a system were to be changed; to inform decisions or judgments that are of a system nature; and to support the design and synthesis of a system, which we call system architecture.

System thinking sits alongside other modes of reasoning, such as critical reasoning (evaluating the validity of claims), analytic reasoning (conducting an analysis from a set of laws or principles), and creative thinking, among others. Well-prepared thinkers use all of these modes of thought (cognition) and recognize when they are using each one (meta-cognition).

This chapter begins by defining what a system is and exploring the property of emergence that gives systems their power (Section 2.2). Subsequently, we examine four tasks that aid us in system thinking:

1. Identify the system, its form, and its function (Section 2.3)
2. Identify the entities of the system, their form and function, and the system boundary and context (Section 2.4)
3. Identify the relationships among the entities in the system and at the boundary, as well as their form and function (Section 2.5)
4. Identify the emergent properties of the system based on the function of the entities, and their functional interactions (Section 2.6)

These tasks will be explained sequentially, but real reasoning is rarely sequential and more often iterative. As discussed in Chapter 1, *methods* are the ways of organizing such tasks to achieve a concrete end. Methods are usually or often applicable. The *principles* on which the methods of system thinking are based are also presented in this chapter.

2.2 Systems and Emergence

Systems

Because system thinking is reasoning about a question, circumstance, or problem explicitly as a system, our starting point for system thinking should be a discussion of *systems.* Few words in the

modern English language are as widely applied or defined as the word "system." The definition that we use in this text is given in Box 2.1.

Box 2.1 Definition: *System*

A system is a set of entities and their relationships, whose functionality is greater than the sum of the individual entities.

The definition has two important parts:

1. A system is made up of entities that interact or are interrelated.
2. When the entities interact, there appears a function that is greater than, or other than, the functions of the individual entities.

At the core of all definitions of the word "system" is the first property listed here: the presence of *entities* and their *relationships.* Entities (also called parts, modules, routines, assemblies, etc.) are simply the chunks that make up the whole. The relationships can exist and be static (as in a connection) or dynamic and interactive (as in an exchange of goods).

Based on this part of the definition, what does *not* qualify as a system? If something is uniform in consistency throughout, it is not a system. For example, a brick (at a macroscopic level) is not a system, because it does not contain entities. However, a brick wall would qualify as a system, because it contains entities (many bricks and much mortar) and relationships (load exchange and geometry). Likewise, if a set of entities have no relationships (say, a person in Ukraine and a bag of rice in Asia), they do not constitute a system.

Notice how hard one must work to define things that are not systems! Someone might argue that at the right scale, a brick is a system: It is made of clay, which itself is a mixture of materials, and the materials have relationships such as sharing load and being in a geometric form (a parallelepiped). Likewise, a person in Ukraine could spend a euro to buy Asian rice, linking these entities into a trading system.

In fact, broadly construed, almost any set of entities can be interpreted as a system, and this is why the word is so commonly used. A closely related concept is the adjective "complex," which (in its original and primary sense) means having many entities and relationships. In some languages, the noun "complex" is used to mean a system, as it sometimes is in technical English (as in "Launch Complex 39A" at the Kennedy Space Center).

Two ideas that are often confused are the concepts system and product. A *product* is something that is, or has the potential to be, exchanged. Thus some products are not systems (rice) and some systems are not products (the solar system), but many of the things we build are both products (exchanged) and systems (many interrelated entities), so the two words have become mixed in common usage.

Another closely related concept is architecture, the subject of this text. In its simplest form, *architecture* can be defined as "an abstract description of the entities of a system and the relationships between those entities." [1] Clearly, the notion of a system (that exists and functions) and architecture (the description of the system) are intimately related.

TABLE 2.1 | Types of emergent functions

	Anticipated Emergence	Unanticipated Emergence
Desirable	Cars transport people Cars keep people warm/cool Cars entertain people	Cars create a sense of personal freedom in people
Undesirable	Cars burn hydrocarbons	Cars can kill people

Emergence

System thinking emphasizes the second property listed in the definition of a system: A system is a set of entities and their relationships, *whose functionality is greater than the sum of the individual entities.*

This emphasized phrase describes what is called *emergence,* and it is the power and the magic of systems. Emergence refers to what appears, materializes, or surfaces when a system operates. Obtaining the desired emergence is why we build systems. Understanding emergence is the goal—and the art—of system thinking.

What emerges when a system comes together? Most obviously and crucially, function emerges. *Function* is what a system does: its actions, outcomes, or outputs. In a designed system, we design so that the anticipated desirable primary function emerges (cars transport people). This primary function is often linked to the benefit produced by the system (we buy cars because they transport people). Anticipated but undesirable outcomes may also emerge (cars burn hydrocarbons). Sometimes, as a system comes together, unanticipated function emerges (cars provide a sense of personal freedom). This is a desirable unanticipated outcome. An undesirable unanticipated function can also emerge (cars can kill people). As suggested by Table 2.1, emergent function can be anticipated or unanticipated, and it can be desirable or undesirable. It is also clear that more than the primary desirable function can emerge from a system (cars can also keep us warm or cool, and cars can entertain people).

The essential aspect of systems is that some new functions emerge. Consider the two elements shown in Figure 2.1: sand and a funnel-shaped glass tube. Sand is a natural material and has no anticipated function. A funnel concentrates or channels a flow. However, when they are put together, a new function emerges: keeping time. How could we have ever expected that sand + funnel would produce a time-keeping device? And how did two mechanical elements, sand and shaped glass, produce an informational system that keeps track of the abstraction called "time"?

In addition to function, *performance* emerges. Performance is how well a system operates or executes its function(s). It is an attribute of the function of the system. How quickly does the car transport people? How accurately does the hourglass keep time? These are issues of performance. Take as an example the human system shown in Figure 2.2, a soccer (or football) team. The function of all soccer teams is the same: the team members must work together to score more goals than the opponent. However, some soccer teams have better performance than others — they win more games. The team portrayed in Figure 2.2 was arguably the highest-performing team in the world in 2014 — the German national team that won the 2014 World Cup.

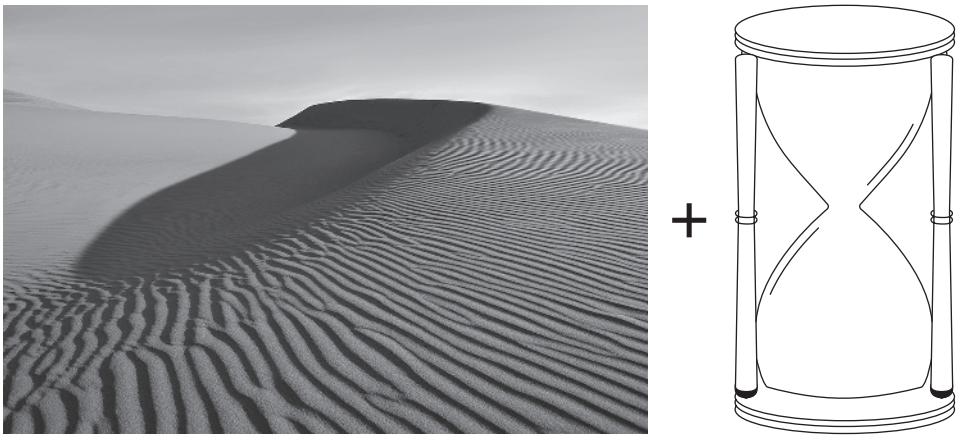

FIGURE 2.1 Emergent function from sand and a funnel: Time keeping. (Source: LOOK Die Bildagentur der Fotografen GmbH/Alamy)

The first principle of system architecture deals with emergence (Box 2.2). Principles are long-enduring truths that are always, or nearly always, applicable. The principles we introduce will generally begin with quotations illustrating how great systems thinkers have expressed the principle. These quotations suggest the timelessness and universality of the principle. Each principle also includes a descriptive part and a prescriptive part (which guide our actions), as well as some further discussion.

There are other attributes of operation that emerge from a system, such as reliability, maintainability, operability, safety, and robustness. These are often called the "ilities." In contrast with functional and performance emergence, which tend to create value immediately, the emergent value created by these "ilities" tends to emerge over the lifecycle of the system. How safely does a car transport people? How reliably does the hourglass keep time? How robustly does the German national soccer team win? How robustly or reliably will the software run? When a car

FIGURE 2.2 Emergent performance: The German soccer team in the 2014 World Cup. (Source: wareham.nl (sport)/Alamy)

Box 2.2 Principle of Emergence

"A system is not the sum of its parts, but the product of the interactions of those parts."

Russell Ackoff

"The whole is more than the sum of the parts."

Aristotle, *Metaphysics*

As the entities of a system are brought together, their interaction will cause function, behavior, performance, and other intrinsic properties to emerge. Consider and attempt to predict the anticipated and unanticipated emergent properties of the system.

- The interaction of entities leads to emergence. Emergence refers to what appears, materializes, or surfaces when a system operates. It is this emergence that can give systems added value.
- As a consequence of emergence, change propagates in unpredictable ways.
- It is difficult to predict how a change in one entity will influence the emergent properties.
- System success occurs when the anticipated properties emerge. System failure occurs when the anticipated emergent properties fail to appear or when unanticipated undesirable emergent properties appear.

breaks down at the side of the road, is it a mechanical "ility" problem or an embedded software "ility" problem?

The final class of emergence is so important that it merits a separate discussion: severe unanticipated and undesirable emergence. We usually call this an *emergency* (from the same word root as emergence!). Cars can lose traction and spin or roll. A soccer team could develop conflicts and lose its effectiveness on the day of an important match. Pictured in Figure 2.3 is a

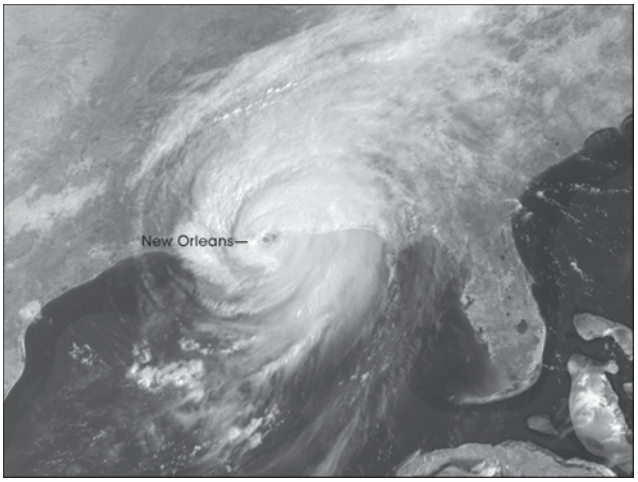

FIGURE 2.3 Emergency as emergence: Hurricane Katrina. (Source: Image courtesy GOES Project Science Office/NASA)

natural example of emergence: Hurricane Katrina as it bore down on New Orleans. The devastation from this system was enormous.

These emergent properties associated with function, performance, the "ilities," and the absence of emergencies are closely related to the *value* that is created by a system. Value is *benefit at cost.* We build systems to deliver the benefit (the worth, importance, or utility as judged by a subjective observer).

In summary:

- A system is a set of entities and their relationships, whose functionality is greater than the sum of the individual entities.
- Almost anything can be considered a system, because almost everything contains entities and relationships.
- Emergence occurs when the functionality of the system is greater than the sum of the functionalities of the individual entities considered separately.
- Understanding emergence is the goal—and the art—of system thinking.
- Function, performance, and the "ilities" emerge as systems operate. These are closely linked to benefit and value, as is the absence of emergencies.

2.3 Task 1: Identify the System, Its Form, and Its Function

Form and Function

Systems simultaneously have the characteristics of *form* and *function.* Form is what the system *is.* Function is what the system *does.* To aid in developing an understanding of form and function in systems and system thinking, we will use four running examples: an amplifier, a design team, the circulatory system, and the solar system. Figures 2.4 through 2.7 show simple illustrations or schematics of these four systems. Note that the examples are chosen to include built and evolved systems, as well as informational, organizational, mechanical, and natural systems.

Each of these systems clearly has a *form.* Form is what a system *is*; it is the physical or informational embodiment that exists or has the potential to exist. Form has shape, configuration, arrangement, or layout. Over some period of time, form is static and perseverant (even

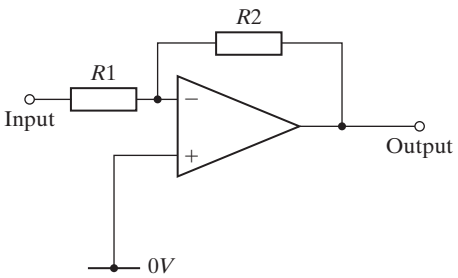

FIGURE 2.4 Amplifier circuit as a system. An operational amplifier and other electronic components that amplify signals.

FIGURE 2.5 Design team (Team X) as a system. Three people whose job it is to come up with a new device design. (Source: Edyta Pawlowska/Fotolia)

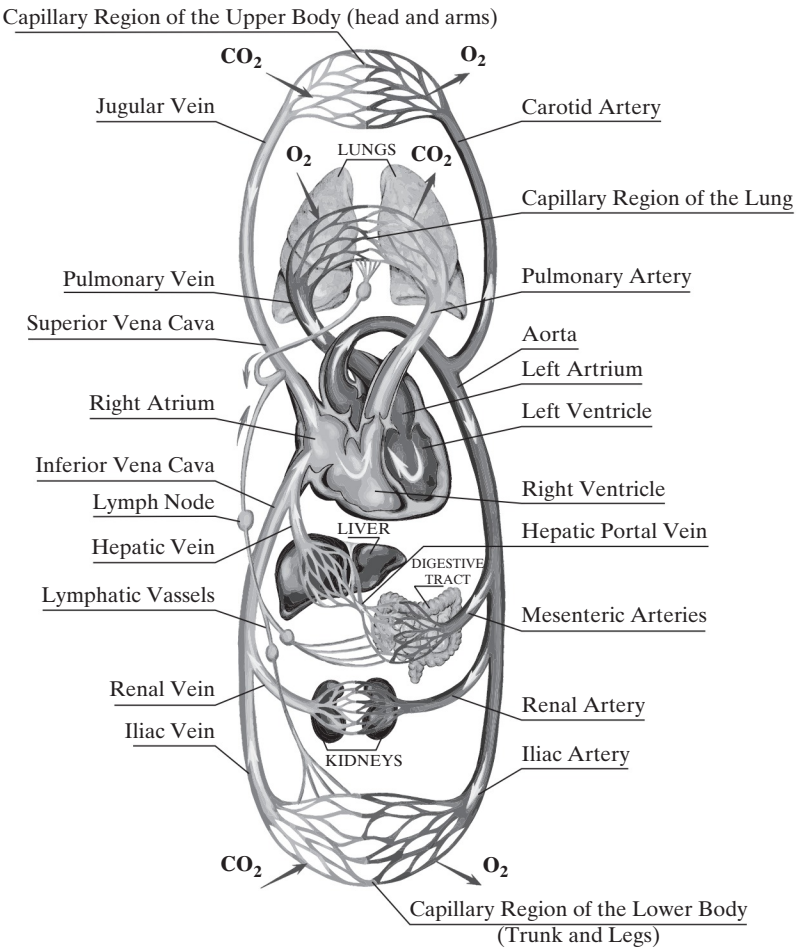

FIGURE 2.6 Circulatory system. The heart, lungs, and capillaries that supply oxygen to tissue and the organs. (Source: Stihii/Shutterstock)

FIGURE 2.7 Solar system. Our sun, and the planets and smaller bodies that orbit it. (Source: JACOPIN/BSIP/Science Source)

though form can be altered, created, or destroyed). Form is the thing that is built; the creator of the system builds, writes, paints, composes, or manufactures it. Form is not function, but form is necessary to deliver function.

Function is what a system *does*; it is the activities, operations, and transformations that cause, create, or contribute to performance. Function is the action for which a thing exists or is employed. Function is not form, but function requires an instrument of form. Emergence occurs in the functional domain. Function, performance, the "ilities," and emergencies are all issues of functionality. Function is more abstract than form, and because it is about transitions, it is more difficult to diagram than form.

Function consists of a *process* and an *operand.* The *process* is the part of function that is pure action or transformation, and thus it is the part that changes the state of the operand. The *operand* is the thing whose state is changed by that process. Function is inherently transient; it involves change in the state of the operand (creation, destruction, or alteration of some aspect of status of the operand). In organizations, function is sometimes referred to as role or responsibilities.

We are now prepared to state Task 1 of System Thinking (Box 2.3).

Box 2.3 Methods: **Task 1 of System Thinking**

Identify the system, its form, and its function.

Now we can apply this first task to our four running examples and identify their form and function, as summarized in Table 2.2.

For each of the built systems, there is an instrument of form, a process, and a value-related operand, whose change in state is the reason for the existence of the system. For the amplifier circuit, the **output signal** is the value-related operand. There may be more than one operand of