Contents

Front matter vii

Part 7: SE Implementation Examples 3
Cover
• Improve the ways in which Part 1 (SEBoK Introduction) provides a starting point for different SEBoK users to find and navigate knowledge relevant to them. This will include consideration of some of the SEBoK Use Cases which were not expanded in previous releases, and possible new case studies covering application domains such as Defense, Health Care or Transport.

• Review Part 2 (Foundations of Systems Engineering) with help from the International Society for the Systems Sciences (ISSS) to better understand the relationships between Systems Science (glossary) and Systems Thinking (glossary) as applied to engineered systems. We hope this will lead to an improved integration of systems principles, concepts, patterns and models into the other systems engineering focused knowledge areas across the SEBoK.

• Look for broader views on the key practices of Part 3 (Systems Engineering and Management) to feed back into the ongoing co evolution of key standards. In particular make more direct reference to the continuing evolution of Agile life cycle thinking and bring in more knowledge sources from the model based SE (MBSE) community.

• Expand our coverage of knowledge on systems engineering application and practices. In particular look for ways to bring in more knowledge on how systems engineering practices such as architecting, life cycle tailoring and model based systems engineering are applied in other domains.

• Identify the other groups, both within the systems engineering community and beyond, with interest in the topics of Part 5 (Enabling Systems Engineering) and Part 6 Related Disciplines and form stronger relationships with them. For example we are working with the IEEE Computer Society about the relationship between SE and Software Engineering.

We continue to work towards ensuring that our coverage of existing systems engineering knowledge is complete and to push the boundaries of that knowledge into new approaches and domains. I also want to strengthen further our links to all members of the systems engineering community through things like the SEBoK Sandbox. If you are interested in any of the activity discussed above or if you have other topics which we should be considering please contact me or the appropriate member of the Editorial Board directly or use one of the available feedback mechanisms.

We have continued to gather review comments and content suggestions from as wide a variety of individuals as possible to make the SEBoK a truly community-led product. Thank you to all those who have already joined this effort and I continue to look forward to working with many of you on future SEBoK releases.

Thank you,

Rick Adcock, Editor-in-Chief
BKCASE Editorial Board
r.d.adcock@cranfield.ac.uk
BKCASE Governance and Editorial Board

BKCASE Governing Board

The three SEBoK steward organizations – the International Council on Systems Engineering (INCOSE), the Institute of Electrical and Electronics Engineers Computer Society (IEEE-CS), and the Systems Engineering Research Center (SERC) provide the funding and resources needed to sustain and evolve the SEBoK and make it available as a free and open resource to all. The stewards appoint the BKCASE Governing Board to be their primary agents to oversee and guide the SEBoK and its companion BKCASE product, GRCSE.

The BKCASE Governing Board includes:

- **INCOSE**
  - Art Pyster (Governing Board Chair), Paul Frenz

- **IEEE Computer Society**
  - Andy Chen, John Keppler

- **SERC**
  - Jon Wade, Cihan Dagli

Past INCOSE governors Bill Miller, Kevin Forsberg, David Newbern, David Walden, Courtney Wright, Dave Olwell, Ken Nidiffer, Richard Fairley and Massood Towhidnejad. The governors would also like to acknowledge John Keppler, IEEE Computer Society, who has been instrumental in helping the Governors to work within the IEEE CS structure.

The stewards appoint the BKCASE Editor in Chief to manage the SEBoK and GRCSE and oversee the Editorial Board.

Editorial Board

The SEBoK Editorial Board is chaired by an Editor in Chief, supported by a group of Associate Editors.

**BKCASE Editor in Chief**

Richard D. Adcock  
*Cranfield University (UK)*  
richard.adcock@incose.org [1]

Responsible for the appointment of SEBoK Editors and for the overall content and coherence of the SEBoK.

Each Editor has his/her area(s) of responsibility, or shared responsibility, highlighted in the table below.
Ariela Sofer  
*George Mason University (USA)*  
asofer@gmu.edu [2]  
Responsible for Part 1

**SEBoK Part 2: Systems**

**Cihan Dagli**  
*Missouri University of Science & Technology (USA)*  
dagli@mst.edu [3]  
Responsible for the Systems Approach Applied to Engineered Systems knowledge areas

**Dov Dori**  
*Massachusetts Institute of Technology (USA) and Technion Israel Institute of Technology (Israel)*  
dori@mit.edu [4]  
Responsible for the Representing Systems with Models knowledge area

**Duane Hybertson**  
*MITRE (USA)*  
Jointly responsible for the Systems Fundamentals, Systems Science and Systems Thinking knowledge areas

**Dov Dori**  
*Massachusetts Institute of Technology (USA) and Technion Israel Institute of Technology (Israel)*  
dori@mit.edu [4]  
Responsible for the Representing Systems with Models knowledge area

**Janet Singer** (USA)  
Jointly responsible for the Systems Fundamentals, Systems Science and Systems Thinking knowledge areas

**Peter Tuddenham** College of Exploration (USA)

**SEBoK Part 3: Systems Engineering and Management**

**Barry Boehm**  
*University of Southern California (USA)*  
boehm@usc.edu [5]  
Jointly responsible for the Systems Engineering Management and Life Cycle Models knowledge areas

**Kevin Forsberg**  
*OGR Systems*  
Jointly responsible for the Systems Engineering Management and Life Cycle Models knowledge areas

**Gregory Parnell**  
*University of Arkansas (USA)*  
gparnell@uark.edu [6]  
Responsible for Systems Engineering Management knowledge area.

**Phyllis Marbach**  
*Incose LA (USA)*

**Garry Roedler**  
*Lockheed Martin (USA)*  
garry.j.roedler@lmco.com [7]  
Responsible for the Concept Definition and System Definition knowledge areas.

**Ken Zemrowski**  
*TASC*  
kenneth.zemrowski@incose.org [8]  
Responsible for the Systems Engineering Standards knowledge area.

**SEBoK Part 4: Applications of Systems Engineering**

**Judith Dahmann**  
*MITRE Corporation (USA)*  
jdahmann@mitre.org [9]  
Jointly responsible for Product Systems Engineering and Systems of Systems (SoS) knowledge areas

**James Martin**  
The Aerospace Corporation  
james.martin@incose.org [11]  
Responsible for the Enterprise Systems Engineering knowledge area.

**Michael Henshaw**  
*Loughborough University (UK)*  
M.J.d.Henshaw@lboro.ac.uk [10]  
Jointly responsible for Product Systems Engineering and Systems of Systems (SoS) knowledge areas
SEBoK Part 5: Enabling Systems Engineering

Emma Sparks
Cranfield University
Jointly responsible for the Enabling Individuals and Enabling Teams knowledge area

Rick Hefner
California Institute of Technology

Tim Ferris Cranfield University

Bernardo Delicado
MBDA / INCOSE

SEBoK Part 6 Related Disciplines

Alice Squires
Washington State University (USA)
alice.squires@wsu.edu [12]

Paul Phister
MANIAC Consulting
phisterp@juno.com [13]

SEBoK Part 7 Systems Engineering Implementation Examples

Dick Fairley
Software & Systems Engineering Associates (USA)
Responsible for Part 7: Systems Engineering Implementation Examples, which includes Case Studies and Vignettes

Graduate Reference Curriculum for Systems Engineering (GRCSE)

David H. Olwell
Naval Postgraduate School (USA)
Senior Editor for GRCSE.

Editorial Board Support

The Assistant Editor provide general editorial support across all topics and assist with both content improvement and production issues.

BKCASE Assistant Editor

Claus Ballegaard Nielsen
Cranfield University (UK)
c.nielsen@cranfield.ac.uk [14]

Interested in Editing?

The Editor in Chief is looking for additional editors to support the evolution of the SEBoK. Editors are responsible for maintaining and updating one to two knowledge areas, including recruiting and working with authors, ensuring the incorporation of community feedback, and maintaining the quality of SEBoK content. We are specifically interested in support for the following knowledge areas:

- System Deployment and Use
- Product and Service Life Management
- Enabling Businesses and Enterprises
- Systems Engineering and Software Engineering
- Procurement and Acquisition
- Systems Engineering and Specialty Engineering
If you are interested in being considered for participation on the Editorial Board, please visit the BKCASE website http://www.bkcase.org/join-us/or contact the BKCASE Staff directly at bkcase.incose.ieeecs@gmail.com [15].

SEBoK v. 1.8 released 27 March 2017

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [16].

References

[1] mailto:richard.adcock@incose.org
[2] mailto:asofer@gmu.edu
[3] mailto:dagli@mst.edu
[4] mailto:dori@mit.edu
[5] mailto:boehm@usc.edu
[6] mailto:gparnell@uark.edu
[7] mailto:garry.j.roedler@lmco.com
[8] mailto:kenneth.zemrowski@incose.org
[9] mailto:jdahmann@mitre.org
[10] mailto:M.J.d.Henshaw@lboro.ac.uk
[11] mailto:judes.martin@incose.org
[12] mailto:alice.squires@wsu.edu
[13] mailto:phisterp@juno.com
[14] mailto:c.nielsen@cranfield.ac.uk
[15] mailto:bkcase.incose.ieeecs@gmail.com
Acknowledgements and Release History

This article describes the contributors to the current version of the SEBoK. For information on contributors to past versions of the SEBoK, please follow the links under "SEBoK Release History" below. To learn more about the updates to the SEBoK for v. 1.8, please see the Letter from the Editor.

Governance

The SEBoK is shaped by the BKCASE Editorial Board and is overseen by the BKCASE Governing Board. A complete list of members for each of these bodies can be found on the BKCASE Governance and Editorial Board page.

Content and Feature Updates for 1.8

This is a minor update, that features a new Healthcare SE Knowledge Area (KA) in Part 4: Applications of Systems Engineering introducing four new articles on healthcare. The focus on MBSE has also been expanded with two new articles, an MBSE case study added to Part 7 and an article describing the use of SysML for modelling Systems Engineering Core Concepts. Part 7 has been expanded with an additional case study on the Northwest Hydro System. The article on Technical Leadership in Part 5 has been updated with new material, and Part 6 has also seen major updates of the Reliability, Availability, and Maintainability and Resilience Engineering articles.

Finally, work has been done on describing the organization of knowledge within the SEBoK with the aim of providing readers with better guidance through the SEBoK structure. The Structure of the SEBoK has been update to provide a better overview of the SEBoK and a simplified version of the central model created for the SEBoK structure has been added to the SEBoK main page and to the introductions of each individual part of the SEBoK.

For more information about this release please refer to Version 1.8.

SEBoK Release History

There have been 16 releases of the SEBoK to date, collected into 6 main releases.

Main Releases

- Version 1.0 - The first version intended for broad use.
- Version 1.1 - A minor update that made modest content improvements.
- Version 1.2 - A minor update, including two new articles and revision of several existing articles.
- Version 1.3 - A minor update, including three new case studies, a new use case, updates to several existing articles, and updates to references.
- Version 1.4 - A minor update, including changes related to ISO/IEC/IEEE 15288:2015 standard, three new case studies and updates to a number of articles.
- Version 1.5 - A minor update, including a restructure and extension of the Software Engineering Knowledge Area, two new case studies, and a number of corrections of typographical errors and updates of outdated references throughout the SEBoK.
- Version 1.6 - A minor update, including a reorganization of Part 1 SEBoK Introduction, a new article on the Transition towards Model Based Systems Engineering and a new article giving an overview of Healthcare Systems Engineering, a restructure of the Systems Engineering and Specialty Engineering KA.
- Version 1.7 - A minor update, including a new Healthcare SE Knowledge Area (KA), expansion of the MBSE area with two new articles, Technical Leadership and Reliability, Availability, and Maintainability and a new case study on the Northwest Hydro System.
Acknowledgements and Release History

- Version 1.8 Current Version
  Click on the links above to read more information about each release.

SEBoK Releases

18 development releases preceded this production release:

1. Version 0.25 on September 15, 2010
2. Version 0.5 on September 19, 2011
3. Version 0.75 on March 15, 2012
4. Version 1.0 on September 14, 2012
5. Version 1.0.1 on November 30, 2012
6. Version 1.1 on April 26, 2013
7. Version 1.1.1 on June 14, 2013
11. Version 1.3.1 on December 5, 2014
12. Version 1.3.2. on April 14, 2015
14. Version 1.5 on December 7, 2015
15. Version 1.5.1 on December 18, 2015
17. Version 1.7 on October 27, 2016
18. Version 1.8 on March 27, 2017

Version 0.25 was released as a PDF document for limited review. A total of 3135 comments were received on this document from 114 reviewers across 17 countries. The author team studied these comments with particular interest in feedback about content and about diversity within the community.

In January 2011, the authors agreed to move from a document-based SEBoK to a wiki-based SEBoK, and beginning with v. 0.5, the SEBoK has been available at www.sebokwiki.org. Making the transition to a wiki provided three benefits:

1. easy worldwide access to the SEBoK;
2. more methods for search and navigation; and
3. a forum for community feedback alongside content that remains stable between versions.

For additional information, see the article on Acknowledgements and Release History.

Wiki Team

The wiki team is responsible for maintenance of the wiki infrastructure as well as technical review of all materials prior to publication.

- Claus Ballegaard Nielsen, Cranfield University.

The wiki is currently supported by Ike Hecht from WikiWorks.

SEBoK v. 1.8 released 27 March 2017
SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [16].

References

Part 7: SE Implementation
Examples
References

Article Sources and Contributors 212
Image Sources, Licenses and Contributors 214
Related Disciplines

Part 6 of the Guide to the SE Body of Knowledge (SEBoK) presents knowledge that would be useful to systems engineers as they interact with these other fields and experts in those fields.

Systems engineering (SE), as a discipline, intersects with other disciplines across the practice of engineering and across the enterprise. The knowledge areas (KAs) contained in this part, and the topics under them, are not meant to comprise additional bodies of knowledge but, rather, to give an overview with emphasis on what a systems engineer needs to know, accompanied by pointers to that knowledge.

Knowledge Areas in Part 6

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 6 contains the following KAs:

- Systems Engineering and Software Engineering
- Systems Engineering and Project Management
- Systems Engineering and Industrial Engineering
- Systems Engineering and Specialty Engineering
References

Works Cited

None.

Primary References


Additional References

None.

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].

References

Systems Engineering and Software Engineering

Software is prominent in most modern systems architectures and is often the primary means for integrating complex system components. Software engineering and systems engineering are not merely related disciplines; they are intimately intertwined. (See Systems Engineering and Other Disciplines.) Good systems engineering is a key factor in enabling good software engineering.

The SEBoK explicitly recognizes and embraces the intertwining between systems engineering and software engineering, as well as defining the relationship between the SEBoK and the Guide to the Software Engineering Body of Knowledge (SWEBOK) (Bourque, and Fairley, 2014).

This knowledge area describes the nature of software, provides an overview of the SWEBOK, describes the concepts that are shared by systems engineers and software engineers, and indicates the similarities and difference in how software engineers and systems engineers apply these concepts and use common terminology. It also describes the nature of the relationships between software engineering and systems engineering and describes some of the methods, models and tools used by software engineers.

Topics
Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Software Engineering in the Systems Engineering Life Cycle
- The Nature of Software
- An Overview of the SWEBOK Guide
- Key Points a Systems Engineer Needs to Know about Software Engineering
- Software Engineering Features - Models, Methods, Tools, Standards, and Metrics

Discussion
Software engineers, like systems engineers,

- engage in analysis and design, allocation of requirements, oversight of component development, component integration, verification and validation, life cycle sustainment, and system retirement.
- work with or as a component specialists (for example, user interface, database, computation, and communication specialists) who construct or otherwise obtain the needed software components.
- adapt existing components and incorporate components supplied by customers and affiliated organizations.

These commonalities would make it appear that software engineering is merely an application of systems engineering, but this is only a superficial appearance. The differences between the two disciplines arise from two fundamental issues:

1. Differences in educational backgrounds (traditional engineering disciplines for SE and the computing disciplines for SWE) and work experiences that result in different approaches to problem solving, and
2. Different ways of applying shared concepts based on the contrasting natures of the software medium and the physical media of traditional engineering.

Table 1 itemizes some of the shared concepts that are applied in different ways by systems engineers and software engineers. Each discipline has made contributions to the other. Table 1 indicates the methods and techniques developed by systems engineers adapted for use by software engineers and, conversely, those that have been adapted for use by systems engineers.

Table 1. Adaptation of Methods Across SE and SWE *
The articles in this knowledge area give an overview of software and software engineering aimed at systems engineers. It also provides more details on the relationship between systems and software life cycles and some of the detailed tools used by software engineers. As systems become more dependent on software as a primary means of delivering stakeholder value the historical distinction between software and systems engineering may need to be challenged. This is a current area of joint discussion between the two communities which will affect the future knowledge in both SEBoK and SWEBok.

References

Works Cited


Primary References


Software Engineering in the Systems Engineering Life Cycle

This article describes how software engineering (SwE) life cycle processes integrate with the SE life cycle. A joint workshop organized by INCOSE, the Systems Engineering Research Center and the IEEE Computer Society was held to consider this relationship (Pyster et al, 2015). This workshop concluded that:

"Software is fundamental to the performance, features, and value of most modern engineering systems. It is not merely part of the system, but often shapes the system architecture; drives much of its complexity and emergent behavior; strains its verification; and drives much of the cost and schedule of its development. Given how significant an impact software has on system development and given how complex modern systems are, one would expect the relationship between the disciplines of systems engineering (SE) and software engineering (SWE) to be well defined. However, the relationship is, in fact, not well understood or articulated."

In this article we give some of the basic relationships between SwE and SE and discuss how these can be related to some of the SEBoK knowledge areas.
Systems Engineering and Software Engineering Life Cycles

The Guide to the Software Engineering Body of Knowledge (SWEBOK) (Bourque and Fairley 2014) describes the life cycle of a software product as:

- analysis and design,
- construction,
- testing,
- operation,
- maintenance, and eventually
- retirement or replacement.

This life cycle is common to most other mature engineering disciplines.

In Part 3 of the SEBoK, SE and Management, there is a discussion of SE life cycle models and life cycle processes. A Generic Life Cycle Model is described, and reproduced in Fig. 1 below. This is used to describe necessary stages in the life cycle of a typical engineered system.

![A Generic Life Cycle Model](SEBoK Original)

Part 3 defines a collection of generic SE life cycle processes which define the activities and information needed across the SE life cycle. These processes include activities which contribute across the whole life cycle, with peaks of focused activity in certain stages, see Applying Life Cycle Processes for details.

The following sections provide a brief discussion of how SwE life cycle processes fit into SE life cycle process models. In practice, the details of this relationship are a key part of how a system life cycle is planned and delivered. The relationship will be shaped by the operating domain practice and solution type. Some examples of this are provided in the Implementation Examples.

Systems Engineering and Software Engineering Standards


The SWEBOK references the equivalent ISO/IEC/IEEE Software Engineering Life Cycle Processes 12207 Standard (2008), which defines a very similar set of processes for software systems. Figure 2 shows the relationship between the Enabling, Acquisition, Project and Technical Systems and Software processes in both 15288 and 12207 and the software specific processes of 12207. This alignment is from the last updates of both 12207 and 15288 in 2008. The SE processes have been further updated in 15288:2015, see Systems Engineering and Management for details. This change has not yet been applied to 12207. An update of 12207 is planned for 2016, in which the alignment to 15288 will be reviewed. See Alignment and Comparison of the Standards for more discussion of the relationships between the standards.
Systems Engineering and Software Engineering Life Cycle Relationships

Pyster et al (2015) define two technical dimensions of engineered systems and of the engineering disciplines associated with them. The vertical dimensions of a system are those that modularize around technically focused engineering concerns involving specific elements of the system; the horizontal dimensions of a system involve cross-cutting concerns at the systems level. Examples of vertical concerns include quality attributes and performance effectiveness; and cost, schedule and risk of physical, organizational or human system elements associated with a particular technology domain. Examples of horizontal concerns include addressing evolving customer preferences that drive systems-level quality attributes, trade-off and optimization; resolving system architecture, decomposition and integration issues; implementing system development processes; and balancing system economics, cost, risk and schedule.

In complex systems projects, SE has a horizontal role while traditional engineering disciplines such as electrical, mechanical, and chemical engineering have vertical roles. To the extent that it is responsible for all aspects of the successful delivery of software related elements SwE can be considered as one of the vertical discipline. All of these traditional vertical disciplines will have some input to the horizontal dimension. However, the nature of software and its role in many complex systems makes SwE a critical discipline for many horizontal concerns. This is discussed further below.

The ISO/IEC/IEEE 12207 software engineering standard (2008) considers two situations:

- The life cycle of software product, containing minimal physical hardware, should use the software specific processes and a simple life cycle
- The life cycle of systems with a significant software content (sometimes called software intensive systems) should integrate the software processes into the SE life cycle

The second of these is the one relevant to the practice of SE and requires a significant horizontal contribution from SwE.

The relationship central to this is the way SwE Implementation Processes (see Fig 2) are used in the SE life cycle to support the implementation of software intensive system elements. This simple relationship must be seen in the context of the concurrency, iteration and recursion relationship between SE life cycle processes described in Applying Life Cycle Processes. This means that, in general, software requirements and architecture processes will be applied alongside system requirements and architecture processes; while software integration and test processes are applied alongside system integration, verification and validation processes. These interrelationships help with vertical software concerns, ensuring detailed software design and construction issues are considered at the system level. They also help with horizontal concerns, ensuring whole system issues are considered and are influenced by an...
understanding of software. See the Nature of Software for more details.

The ways these related processes work together will depend on the systems approach to solution synthesis used and how this influences the life cycle. If a top down approach is used, problem needs and system architecture will drive software implementation and realization. If a bottom up approach is used, the architecture of existing software will strongly influence both the system solution and the problem which can be considered. In Applying Life Cycle Processes a "middle-out" approach is described which combines these two ideas and is the most common way to develop systems. This approach needs a two-way relationship between SE and SwE technical processes.

The SW Support Processes may also play these vertical and horizontal roles. Part 3 contains knowledge areas on both System Deployment and Use which includes operation, maintenance and logistics; and Systems Engineering Management which covers the project processes shown in Figure 2. SwE support processes focus on the successful vertical deployment and use of software system elements and the management needed to achieve this. They also support their equivalent horizontal SE processes in contributing to the success of the whole system life cycle. The Software Reuse Processes have a particularly important role to play in deployment and use and Product and Service Life Management processes. The later considers Service Life Extension, Capability Updates, Upgrades, and Modernization and system Disposal and Retirement. All of these horizontal software engineering activities rely on the associated SE activities having a sufficient understanding of the strengths and limitations of software and SwE, see Key Points a Systems Engineer Needs to Know about Software Engineering.

The Life Cycle Models knowledge area also defines how Vee and Iterative life cycle models provide a framework to tailor the generic life cycle and process definitions to different types of system development. Both models, with some modification, apply equally to the development of products and services containing software. Thus, the simple relationships between SE and SwE processes will form the basis for tailoring to suite project needs within a selected life cycle model.

Software and Systems Challenges

Pyster et al. (2015) define three classes of software intensive systems distinguished by the primary sources of novelty, functionality, complexity and risk in their conception, development, operation and evolution. These are briefly described below:

- **Physical Systems** operate on and generate matter or energy. While they often utilize computation and software technologies as components, those components are not dominant in the horizontal dimension of engineering. Rather, in such systems, they are defined as discrete system elements and viewed and handled as vertical concerns.

- **Computational Systems** include those in which computational behavior and, ipso facto, software are dominant at the systems level. The primary purpose of these systems is to operate on and produce data and information. While these systems always include physical and human elements, these are not the predominant challenges in system development, operation and evolution.

- **Cyber-Physical Systems** are a complex combination of computational and physical dimensions. Such systems are innovative, functionally complex and risky in both their cyber and physical dimensions. They pose major horizontal engineering challenges across the board. In cyber-physical systems, cyber and physical elements collaborate in complex ways to deliver expected system behavior.

Some of the challenges of physical and computational systems are well known and can be seen in many SE and SwE case studies. For example, physical system life cycles often make key decisions about the system architecture or hardware implementation which limit the subsequent development of software architecture and designs. This can lead to software which is inefficient and difficult or expensive to change. Problems which arise later in the life of such systems may be dealt with by changing software, or human, elements. This is sometimes done in a way which does not fully consider SwE design and testing practices. Similarly, computational systems may be dominated by the software architecture, without sufficient care taken to consider the best solutions for enabling hardware or people. In
particular, operator interfaces, training and support may not be considered leading to the need for expensive organizational fixes once they are in use. Many computational systems in the past have been developed without a clear view of the user need they contribute to, or the other systems they must work with to do so. These and other related issues point to a need for system and software engineers with a better understanding of each other’s disciplines. Pyster et al. considers how SE and SwE education might be better integrated to help achieve this aim.

Examples of cyber-physical systems increasingly abound – smart automobiles, power grids, robotic manufacturing systems, defense and international security systems, supply-chain systems, the so-called internet of things, etc. In these systems there is no clear distinction between software elements and the whole system solution. The use of software in these systems is central to the physical outcome and software is often the integrating element which brings physical elements and people together. These ideas are closely align with the Service System Engineering approach described in Part 4.

SEBoK Part 3 includes a Business and Mission Analysis process which is based on the equivalent process in the updated ISO/IEC/IEEE 15288 (2015). This process enables SE to be involved in the selection and bounding of the problem situation which forms the starting point for an engineered system life cycle. For cyber physical systems an understanding of the nature of software is needed in the formulation of the problem, since this is often fundamentally driven by the use of software to create complex adaptive solution concepts. This close coupling of software, physical and human system elements across the system of interest continues throughout the system life cycle making it necessary to consider all three in most horizontal system level decision.

The life cycle of cyber physical systems cannot be easily partitioned into SE and SwE achieving their own outcomes, but working together on horizontal system issues. It will require a much more closely integrated approach, requiring systems and software engineers with a complementary set of competencies, and changes how the two disciplines are seen in both team and organizational structures. See Enabling Systems Engineering.

References

Works Cited


Primary References

Additional References

SEBoK Discussion
Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox[1].
The nature of the software medium has many consequences for systems engineering (SE) of software-intensive systems. Fred Brooks has famously observed that four properties of software, taken together, differentiate it from other kinds of engineering artifacts (Brooks 1995). These four properties are

1. complexity,
2. conformity,
3. changeability
4. invisibility.

Brooks states:

Software entities are more complex for their size than perhaps any other human construct because no two parts are alike (at least above the statement level). If they are, we make the two similar parts into a subroutine — open or closed. In this respect, software systems differ profoundly from computers, buildings, or automobiles, where repeated elements abound. (Brooks 1995, p 82)

**Complexity**

The complexity of software arises from the large number of unique interacting parts in a software system. The parts are unique because they are encapsulated as functions, subroutines, or objects, and invoked as needed rather than being replicated. Software parts have several different kinds of interactions, including serial and concurrent invocations, state transitions, data couplings, and interfaces to databases and external systems.

Depiction of a software entity often requires several different design representations to portray the numerous static structures, dynamic couplings, and modes of interaction that exist in computer software. Complexity within the parts, and in the connections among parts requires that changes undergo substantial design rigor and regression testing. Software provides functionality for components that are embedded, distributed and data centric. Software can implement simple control aop s well as complex algorithms and heuristics.

Complexity can hide defects that may not be discovered easily, thus requiring significant additional and unplanned rework.

**Conformity**

Software, unlike a physical product, has no underlying natural principles which it must conform to, such as Newton’s laws of motion. However, software must conform to exacting specifications in the representation of each of its parts, in the interfaces to other internal parts, and in the connections to the environment in which it operates. A missing semicolon or other syntactic error can be detected by a compiler. But, a defect in the program logic or a timing error may be difficult to detect when encountered during operation.on.

Unlike software, tolerance among the interfaces of physical entities is the foundation of manufacturing and assembly. No two physical parts that are joined together have, or are required to have, exact matches. There are no corresponding tolerances in the interfaces among software entities or between software entities and their environments. There are no interface specifications for software stating that a parameter can be an integer plus or minus 2%. Interfaces among software parts must agree exactly in numbers, types of parameters and kinds of couplings.

Lack of conformity can cause problems when an existing software component cannot be reused as planned because it does not conform to the needs of the product under development. Lack of conformity might not be discovered until late in a project, thus necessitating the development and integration of an acceptable component to replace the one that cannot be reused. This requires an unplanned allocation of resources (usually) and can delay project completion.
**Changeability**

Software coordinates the operation of physical components and provides most of the functionality in software-intensive systems. Because software is the most malleable (easily changed) element in a software-intensive system, it is the most frequently changed element. This is particularly true during the late stages of a development project and during system sustainment. However, this does not mean that software is easy to change. Complexity and the need for conformity can make changing software an extremely difficult task. Changing one part of a software system often results in undesired side effects in other parts of the system, requiring more changes before the software can operate at maximum efficiency.

**Invisibility**

Software is said to be invisible because it has no physical properties. While the effects of executing software on a digital computer are observable, software itself cannot be seen, tasted, smelled, touched, or heard. Software is an intangible entity because our five human senses are incapable of directly sensing it.

Work products such as requirements specifications, design documents, source code and object code are representations of software, but they are not the software. At the most elemental level, software resides in the magnetization and current flow in an enormous number of electronic elements within a digital device. Because software has no physical presence, software engineers must use different representations at different levels of abstraction in an attempt to visualize the inherently invisible entity.

**Uniqueness**

One other point about the nature of software that Brooks alludes to, but does not explicitly call out is the uniqueness of software. Software and software projects are unique for the following reasons:

- Software has no physical properties;
- Software is the product of intellect-intensive teamwork;
- Productivity of software developers varies more widely than the productivity of other engineering disciplines;
- Estimation and planning for software projects is characterized by a high degree of uncertainty, which can be at best partially mitigated by best practices;
- Risk management for software projects is predominantly process-oriented;
- Software alone is useless, as it is always a part of a larger system; and
- Software is the most frequently changed element of software intensive systems.

**References**

**Works Cited**


**Primary References**


An Overview of the SWEBOK Guide

Systems engineers are fortunate that the software community has developed its own body of knowledge. The introduction to Version 3 of the Guide to the Software Engineering Body of Knowledge states:

*The purpose of the Guide is to describe the portion of the Body of Knowledge that is generally accepted, to organize that portion, and to provide topical access to it.* (Bourque and Fairley 2014)

**SWEBOK Guide Version 3**

The purposes of SWEBOK V3 are as follows:

- to characterize the contents of the software engineering discipline;
- to promote a consistent view of software engineering worldwide;
- to clarify the place of, and set the boundary of, software engineering with respect to other disciplines;
- to provide a foundation for training materials and curriculum development; and
- to provide a basis for certification and licensing of software engineers.

SWEBOK V3 contains 15 knowledge areas (KAs). Each KA includes an introduction, a descriptive breakdown of topics and sub-topics, recommended references, references for further reading, and a matrix matching reference material with each topic. An appendix provides a list of standards most relevant to each KA. An overview of the individual KAs presented in the guide is provided in the next two sections.
Knowledge Areas Characterizing the Practice of Software Engineering

Software Requirements
The Software Requirements KA is concerned with the elicitation, negotiation, analysis, specification, and validation of software requirements. It is widely acknowledged within the software industry that software engineering projects are critically vulnerable when these activities are performed poorly. Software requirements express the needs and constraints placed on a software product that contribute to the solution of some real-world problems.

Software Design
Design is defined as both *the process of defining the architecture, components, interfaces, and other characteristics of a system or component and the result of [that] process* (IEEE 1991). The Software Design KA covers the design process and the resulting product. The software design process is the software engineering life cycle activity in which software requirements are analyzed in order to produce a description of the software's internal structure and its behavior that will serve as the basis for its construction. A software design (the result) must describe the software architecture -- that is, how software is decomposed and organized into components and the interfaces between those components. It must also describes the components at a level of detail that enables their construction.

Software Construction
Software construction refers to the detailed creation of working software through a combination of detailed design, coding, unit testing, integration testing, debugging, and verification. The Software Construction KA includes topics related to the development of software programs that will satisfy their requirements and design constraints. This KA covers software construction fundamentals; managing software construction; construction technologies; practical considerations; and software construction tools.

Software Testing
Testing is an activity performed to evaluate product quality and to improve it by identifying defects. Software testing involves dynamic verification of the behavior of a program against expected behavior on a finite set of test cases. These test cases are selected from the (usually very large) execution domain. The Software Testing KA includes the fundamentals of software testing; testing techniques; human-computer user interface testing and evaluation; test-related measures; and practical considerations.

Software Maintenance
Software maintenance involves enhancing existing capabilities, adapting software to operate in new and modified operating environments, and correcting defects. These categories are referred to as perfective, adaptive, and corrective software maintenance. The Software Maintenance KA includes fundamentals of software maintenance (nature of and need for maintenance, categories of maintenance, maintenance costs); key issues in software maintenance (technical issues, management issues, maintenance cost estimation, measurement of software maintenance); the maintenance process; software maintenance techniques (program comprehension, re-engineering, reverse engineering, refactoring, software retirement); disaster recovery techniques, and software maintenance tools.
Software Configuration Management

The configuration of a system is the functional and/or physical characteristics of hardware, firmware, software, or a combination of these. It can also be considered as a collection of specific versions of hardware, firmware, or software items combined according to specific build procedures to serve a particular purpose. Software configuration management (SCM) is thus the discipline of identifying the configuration of a system at distinct points in time for the purposes of systematically controlling changes to the configuration, as well as maintaining the integrity and traceability of the configuration throughout the software life cycle. The Software Configuration Management KA covers management of the SCM process; software configuration identification, control, status accounting, auditing; software release management and delivery; and software configuration management tools.

Software Engineering Management

Software engineering management involves planning, coordinating, measuring, reporting, and controlling a project or program to ensure that development and maintenance of the software is systematic, disciplined, and quantified. The Software Engineering Management KA covers initiation and scope definition (determining and negotiating requirements, feasibility analysis, and review and revision of requirements); software project planning (process planning, estimation of effort, cost, and schedule, resource allocation, risk analysis, planning for quality); software project enactment (measuring, reporting, and controlling; acquisition and supplier contract management); product acceptance; review and analysis of project performance; project closure; and software management tools.

Software Engineering Process

The Software Engineering KA is concerned with definition, implementation, assessment, measurement, management, and improvement of software life cycle processes. Topics covered include process implementation and change (process infrastructure, models for process implementation and change, and software process management); process definition (software life cycle models and processes, notations for process definition, process adaptation, and process automation); process assessment models and methods; measurement (process measurement, products measurement, measurement techniques, and quality of measurement results); and software process tools.

Software Engineering Models and Methods

The Software Engineering Models and Methods KA addresses methods that encompass multiple life cycle stages; methods specific to particular life cycle stages are covered by other KAs. Topics covered include modeling (principles and properties of software engineering models; syntax vs. semantics vs. invariants; preconditions, post-conditions, and invariants); types of models (information, structural, and behavioral models); analysis (analyzing for correctness, completeness, consistency, quality and interactions; traceability; and tradeoff analysis); and software development methods (heuristic methods, formal methods, prototyping methods, and agile methods).

Software Quality

Software quality is a pervasive software life cycle concern that is addressed in many of the SWEBOK V3 KAs. In addition, the Software Quality KA includes fundamentals of software quality (software engineering cultures, software quality characteristics, the value and cost of software quality, and software quality improvement); software quality management processes (software quality assurance, verification and validation, reviews and audits); and practical considerations (defect characterization, software quality measurement, and software quality tools).
Software Engineering Professional Practice
Software engineering professional practice is concerned with the knowledge, skills, and attitudes that software engineers must possess to practice software engineering in a professional, responsible, and ethical manner. The Software Engineering Professional Practice KA covers professionalism (professional conduct, professional societies, software engineering standards, employment contracts, and legal issues); codes of ethics; group dynamics (working in teams, cognitive problem complexity, interacting with stakeholders, dealing with uncertainty and ambiguity, dealing with multicultural environments); and communication skills.

Knowledge Areas Characterizing the Educational Requirements of Software Engineering

Software Engineering Economics
The Software Engineering Economics KA is concerned with making decisions within the business context to align technical decisions with the business goals of an organization. Topics covered include fundamentals of software engineering economics (proposals, cash flow, the time-value of money, planning horizons, inflation, depreciation, replacement and retirement decisions); not for-profit decision-making (cost-benefit analysis, optimization analysis); estimation, economic risk and uncertainty (estimation techniques, decisions under risk and uncertainty); and multiple attribute decision making (value and measurement scales, compensatory and non-compensatory techniques).

Computing Foundations
The Computing Foundations KA covers fundamental topics that provide the computing background necessary for the practice of software engineering. Topics covered include problem solving techniques, abstraction, algorithms and complexity, programming fundamentals, the basics of parallel and distributed computing, computer organization, operating systems, and network communication.

Mathematical Foundations
The Mathematical Foundations KA covers fundamental topics that provide the mathematical background necessary for the practice of software engineering. Topics covered include sets, relations, and functions; basic propositional and predicate logic; proof techniques; graphs and trees; discrete probability; grammars and finite state machines; and number theory.

Engineering Foundations
The Engineering Foundations KA covers fundamental topics that provide the engineering background necessary for the practice of software engineering. Topics covered include empirical methods and experimental techniques; statistical analysis; measurements and metrics; engineering design; simulation and modeling; and root cause analysis.

Related Disciplines
SWEBOK V3 also discusses related disciplines. The related disciplines are those that share a boundary, and often a common intersection, with software engineering. SWEBOK V3 does not characterize the knowledge of the related disciplines but, rather, indicates how those disciplines interact with the software engineering discipline. The related disciplines include

- Computer Engineering
- Computer Science
- General Management
An Overview of the SWEBOK Guide

- Mathematics
- Project Management
- Quality Management
- Systems Engineering

References

Works Cited

Primary References

Additional References
None.

SEBoK v. 1.8 released 27 March 2017

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox.¹

¹ ENCODED_CONTENT

MTUzNzMPGRpdBiZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgICogKiBDT05GSUdVUkFUSU9 ... 9tIiBjbGFzcz0iZHNxLWJybGluayI+YmxvZyBjb21tZW50cyBwb3dlcmVkIGJ5IDxzcGFuIGNsYXNzPSJsb2dvLWRpc3F1cyI+RGlzcXVzPC9zcGFuPjwvYT4=

END_ENCODED_CONTENT
Key Points a Systems Engineer Needs to Know about Software Engineering

The field of Software Engineering (glossary) is extensive and specialized. Its importance to modern systems makes it necessary for systems engineers to be knowledgeable about software engineering and its relationship to systems engineering.

Key Concepts a Systems Engineer Needs to Know about Software Engineering

The following items are significant aspects that systems engineers need to know about software and software engineering. Most are documented in (Fairley and Willshire 2011):

1. **For the time, effort, and expense devoted to developing it, software is more complex than most other system components** - Software complexity arises because few elements in a software program (even down to the statement level) are identical as well as because of the large number of possible decision paths found even in small programs, with the number of decision paths through a large program often being astronomical. There are several detailed references on software complexity. The SWEBOK (Bourque and Fairley 2014) discusses minimizing complexity as part of software construction fundamentals. Zuse (1991) has a highly cited article on software complexity measures and methods. Chapters 2 and 3 of the SWEBOK also have further references.

2. **Software testing and reviews are sampling processes** - In all but the simplest cases, exhaustive testing of software is impossible because of the large number of decision paths through most programs. Also, the combined values of the input variables selected from a wide combinatorial range may reveal defects that other combinations of the variables would not detect. Software test cases and test scenarios are chosen in an attempt to gain confidence that the testing samples are representative of the ways the software will be used in practice. Structured reviews of software are an effective mechanism for finding defects, but the significant effort required limits exhaustive reviewing. Criteria must be established to determine which components (or sub-components) should be reviewed. Although there are similar concerns about exhaustive testing and reviewing of physical products, the complexity of software makes software testing, reviews, and the resulting assurance provided, more challenging. Other points include:
   1. All software testing approaches and techniques are heuristic. Hence, there is no universal "best" approach, practice, or technique for testing, since these must be selected based on the software context.
   2. Exhaustive testing is not possible.
   3. Errors in software tend to cluster within the software structures; therefore, any one specific approach or a random approach to testing is not advised.
   4. Pesticide paradox exists. As a result, running the same test over and over on the same software-system provides no new information.
   5. Testing can reveal the presence of defects but cannot guarantee that there will be no errors, except under the specific conditions of a given test.
   6. Testing, including verification and validation (V&V), must be performed early and continually throughout the lifecycle (end to end.
   7. Even after extensive testing and V&V, errors are likely to remain after long term use of the software.
   8. Chapter 4 of the SWEBOK discusses software testing and provides a bibliography.

3. **Software often provides the interfaces that interconnect other system components** - Software is often referred to as the glue that holds a system together because the interfaces among components, as well as the interfaces to the environment and other systems, are often provided by digital sensors and controllers that operate via software. Because software interfaces are behavioral rather than physical, the interactions that occur among software components often exhibit emergent behaviors that cannot always be predicted in advance. In addition to...
component interfaces, software usually provides the computational and decision algorithms needed to generate command and control signals. The SWEBOK has multiple discussions of interfaces: Chapter 2 on Software Design is a good starting point and includes a bibliography.

4. **Every software product is unique** - The goal of manufacturing physical products is to produce replicated copies that are as nearly identical as much as possible, given the constraints of material sciences and manufacturing tools and techniques. Because replication of existing software is a trivial process (as compared to manufacturing of physical products), the goal of software development is to produce one perfect copy (or as nearly perfect as can be achieved given the constraints on schedule, budget, resources, and technology). Much of software development involves altering existing software. The resulting product, whether new or modified, is uniquely different from all other software products known to the software developers. Chapter 3 of the SWEBOK provides discussion of software reuse and several references.

5. **In many cases, requirements allocated to software must be renegotiated and reprioritized** - Software engineers often see more efficient and effective ways to restate and prioritize requirements allocated to software. Sometimes, the renegotiated requirements have system-wide impacts that must be taken into account. One or more senior software engineers should be, and often are, involved in analysis of system-level requirements. This topic is addressed in the SWEBOK in Chapter 1, with topics on the iterative nature of software and change management.

6. **Software requirements are prone to frequent change** - Software is the most frequently changed component in complex systems, especially late in the development process and during system sustainment. This is due to the fact that software is perceived to be the most easily changed component of a complex system. This is not to imply that changes to software requirements, and the resulting changes to the impacted software, can be easily done without undesired side effects. Careful software configuration management is necessary, as discussed in Chapter 6 of the SWEBOK, which includes extensive references.

7. **Small changes to software can have large negative effects** (A corollary to frequently changing software requirements: There are no small software changes) - In several well-known cases, modifying a few lines of code in very large systems that incorporated software negatively impacted the safety, security, and/or reliability of those systems. Applying techniques such as traceability, impact analysis, object-oriented software development, and regression testing reduces undesired side effects of changes to software code. These approaches limit but do not eliminate this problem.

8. **Some quality attributes for software are subjectively evaluated** - Software typically provides the interfaces to systems that have human users and operators. The intended users and operators of these systems often subjectively evaluate quality attributes, such as ease of use, adaptability, robustness, and integrity. These quality attributes determine the acceptance of a system by its intended users and operators. In some cases, systems have been rejected because they were not judged to be suitable for use by the intended users in the intended environment, even though those systems satisfied their technical requirements. Chapter 10 of the SWEBOK provides an overview of software quality, with references.

9. **The term prototyping has different connotations for systems engineers and software engineers** - For a systems engineer, a prototype is typically the first functioning version of a hardware. For software engineers, software prototyping is primarily used for two purposes: (1) as a mechanism to elicit user requirements by iteratively evolving mock-ups of user interfaces, and (2) as an experimental implementation of some limited element of a proposed system to explore and evaluate alternative algorithms. Chapter 1 of the SWEBOK discusses this and provides excellent references.

10. **Cyber security is a present and growing concern for systems that incorporate software** - In addition to the traditional specialty disciplines of safety, reliability, and maintainability, systems engineering teams increasingly include security specialists at both the software level and the systems level in an attempt to cope with the cyber attacks that may be encountered by systems that incorporate software. Additional information about security engineering can be found in the Systems Engineering and Specialty Engineering KA.
11. **Software growth requires spare capacity** - Moore’s Law no longer fully comes to the rescue (Moore, 1965). As systems adapt to changing circumstances, the modifications can most easily be performed and upgraded in the software, requiring additional computer execution cycles and memory capacity (Belady and Lehman 1979). For several decades, this growth was accommodated by Moore’s Law, but recent limits that have occurred as a result of heat dissipation have influenced manufacturers to promote potential computing power growth by slowing down the processors and putting more of them on a chip. This requires software developers to revise their programs to perform more in parallel, which is often an extremely difficult problem (Patterson 2010). This problem is exacerbated by the growth in mobile computing and limited battery power.

12. **Several Pareto 80-20 distributions apply to software** - These refers to the 80% of the avoidable rework that comes from 20% of the defects, that 80% of the defects come from 20% of the modules, and 90% of the downtime comes from at most 10% of the defects (Boehm and Basili 2001). These, along with recent data indicating that 80% of the testing business value comes from 20% of the test cases (Bullock 2000), indicate that much more cost-effective software development and testing can come from determining which 20% need the most attention.

13. **Software estimates are often inaccurate** - There are several reasons software estimates are frequently inaccurate. Some of these reasons are the same as the reasons systems engineering estimates are often inaccurate: unrealistic assumptions, vague and changing requirements, and failure to update estimates as conditions change. In addition, software estimates are often inaccurate because productivity and quality are highly variable among seemingly similar software engineers. Knowing the performance characteristics of the individuals who will be involved in a software project can greatly increase the accuracy of a software estimate. Another factor is the cohesion of the software development team. Working with a team that has worked together before and knowing their collective performance characteristics can also increase the accuracy of a software estimate. Conversely, preparing an estimate for unknown teams and their members can result in a very low degree of accuracy. Chapter 7 of the SWEBOK [1] briefly discusses this further. Kitchenam (1997) discusses the organizational context of uncertainty in estimates. Lederer and Prasad (1995) also identify organizational and management issues that increase uncertainty; additionally, a recent dissertation from Sweden by Magazinus (2012) shows that the issues persist.

14. **Most software projects are conducted iteratively** - "Iterative development" has a different connotation for systems engineers and software engineers. A fundamental aspect of iterative software development is that each iteration of a software development cycle adds features and capabilities to produce a next working version of partially completed software. In addition, each iteration cycle for software development may occur on a daily or weekly basis, while (depending on the scale and complexity of the system) the nature of physical system components typically involves iterative cycles of longer durations. Classic articles on this include (Royce 1970) and (Boehm 1988), among others. Larman and Basili (2003) provide a history of iterative development, and the SWEBOK discusses this in life cycle processes in Chapter 8.

15. **Teamwork within software projects is closely coordinate** - The nature of software and its development requires close coordination of work activities that are predominately intellectual in nature. Certainly other engineers engage in intellectual problem solving, but the collective and ongoing daily problem solving required of a software team requires a level of communication and coordination among software developers that is of a different more elevated type. Highsmith (2000) gives a good overview.

16. **Agile development processes are increasingly used to develop software** - Agile development of software is a widely used and growing approach to developing software. Agile teams are typically small and closely coordinated, for the reasons cited above. Multiple agile teams may be used on large software projects, although this is highly risky without an integrating architecture (Elssamadisy and Schalliol 2002). Agile development proceeds iteratively in cycles that produce incremental versions of software, with cycle durations that vary from one day to one month, although shorter durations are more common. Among the many factors that distinguish agile development is the tendency to evolve the detailed requirements iteratively. Most agile approaches do not

17. **Verification and validation (V&V) of software should preferably proceed incrementally and iteratively** - Iterative development of working product increments allows incremental verification, which ensures that the partial software product satisfies the technical requirements for that incremental version; additionally, it allows for the incremental validation (ISO/IEC/IEEE 24765) of the partial product to make certain that it satisfies its intended use, by its intended users, in its intended environment. Incremental verification and validation of working software allows early detection and correction of encountered problems. Waiting to perform integration, verification, and validation of complex system until later life cycle stages, when these activities are on the critical path to product release, can result in increased cost and schedule impacts. Typically, schedules have minimal slack time during later stages in projects. However, with iterative V&V, software configuration management processes and associated traceability aspects may become complex and require special care to avoid further problems. Chapter 4 of the SWEBOK discusses software testing, and provides numerous references, including standards. Much has been written on the subject; a representative article is (Wallace and Fujii 1989).

18. **Performance trade-offs are different for software than systems** - Systems engineers use “performance” to denote the entire operational envelope of a system; whereas, software engineers use “performance” to mean response time and the throughput of software. Consequentially, systems engineers have a larger design space in which to conduct trade studies. In software, performance is typically enhanced by reducing other attributes, such as security or ease of modification. Conversely, enhancing attributes such as security and ease of modification typically impacts performance of software (response time and throughput) in a negative manner.

19. **Risk management for software projects differs in kind from risk management for projects that develop physical artifacts** - Risk management for development of hardware components is often concerned with issues such as supply chain management, material science, and manufacturability. Software and hardware share some similar risk factors: uncertainty in requirements, schedule constraints, infrastructure support, and resource availability. In addition, risk management in software engineering often focuses on issues that result from communication problems and coordination difficulties within software development teams, across software development teams, and between software developers and other project members (e.g., hardware developers, technical writers, and those who perform independent verification and validation). See (Boehm 1991) for a foundational article on the matter.

20. **Software metrics include product measures and process measures** - The metrics used to measure and report progress of software projects include product measures and process (ISO/IEC/IEEE 24765) measures. Product measures include the amount of software developed (progress), defects discovered (quality), avoidable rework (defect correction), and budgeted resources used (technical budget, memory and execution cycles consumed, etc.). Process measures include: the amount of effort expended (because of the people-intensive nature of software development), productivity (software produced per unit of effort expended), production rate (software produced per unit time), milestones achieved and missed (schedule progress), and budgeted resources used (financial budget). Software metrics are often measured on each (or, periodically, some) of the iterations of a development project that produces a next working version of the software. Chapter 8 and Chapter 7 of the SWEBOK address this.

21. **Progress on software projects is sometimes inadequately tracked** - In some cases, progress on software projects is not adequately tracked because relevant metrics are not collected and analyzed. A fundamental problem is that accurate tracking of a software project depends on knowing how much software has been developed that is suitable for delivery into the larger system or into a user environment. Evidence of progress, the form of working software, is one of the primary advantages of the iterative development of working software increments.
Key Points a Systems Engineer Needs to Know about Software Engineering

References

Works Cited


Primary References


**Additional References**


**SEBoK Discussion**

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. **If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.**

**If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox** [1].

ENCODED_CONTENT

MjEwMDUPGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NvbnRhaW5lciB0aGlzIHRvIGludGVyLjwvYT4=

END_ENCODED_CONTENT
In recent decades software has become ubiquitous. Almost all modern engineered systems include significant software subsystems; this includes systems in the transportation, finance, education, healthcare, legal, military, and business sectors. Along with the increase in software utility, capability, cost, and size there has been a corresponding growth in methods, models, tools, metrics and standards, which support software engineering.

Chapter 10 of the SWEBOK discusses modeling principles and types, and the methods and tools that are used to develop, analyze, implement, and verify the models. The other SWEBOK chapters on the software development phases (e.g., Software Design) discuss methods and tools specific to the phase. Table 1 identifies software engineering features for different life-cycle phases. The table is not meant to be complete; it simply provide examples. In Part 2 of the SEBoK there is a discussion of models and the following is one of the definitions offered: "an abstraction of a system, aimed at understanding, communicating, explaining, or designing aspects of interest of that system" (Dori 2002).

For the purposes of Table 1 we extend the definition of a model to some aspect of the software system or it development. So as an example, we list "Project Plan" as a model in the Software Management area. The idea is that the Project Plan provides a model of how the project is going to be carried out: the project team organization, the process to be used, the work to be done, the project schedule, and the resources needed.

**Table 1: SWE Features (SEBoK Original)**

<table>
<thead>
<tr>
<th>Life-Cycle Activity</th>
<th>Models</th>
<th>Methods &amp; Tools</th>
<th>Standards</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software Management</td>
<td>Life-Cycle Process Model</td>
<td>Effort, Schedule and Cost Estimation</td>
<td>[IEEE 828]</td>
</tr>
<tr>
<td></td>
<td>Work Breakdown Structure</td>
<td>Risk Analysis</td>
<td>[IEEE 1058]</td>
</tr>
<tr>
<td></td>
<td>Constructive Cost Model (COCOMO)</td>
<td>Data Collection</td>
<td>[IEEE 1540]</td>
</tr>
<tr>
<td></td>
<td>Project Plan</td>
<td>Project Tracking</td>
<td>[IEEE 12207]</td>
</tr>
<tr>
<td></td>
<td>Configuration Management (CM) Plan</td>
<td>CM Management</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Risk Management Plan</td>
<td>Iterative/Incremental Development</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Agile Development</td>
<td></td>
</tr>
<tr>
<td>Software Requirements</td>
<td>Functional Model</td>
<td>Requirements Elicitation</td>
<td>[IEEE 830]</td>
</tr>
<tr>
<td></td>
<td>User Class Model</td>
<td>Prototyping</td>
<td>[IEEE 1012]</td>
</tr>
<tr>
<td></td>
<td>Data Flow Diagram</td>
<td>Structural Analysis</td>
<td>[IEEE 12207]</td>
</tr>
<tr>
<td></td>
<td>Object Model</td>
<td>Data-Oriented Analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Formal Model</td>
<td>Object-Oriented Analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td>User Stories</td>
<td>Object Modeling Language (OML)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Formal Methods</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requirements Specification</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Requirements Inspection</td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td>Architectural Model</td>
<td>Structured Design</td>
<td>[IEEE 1012]</td>
</tr>
<tr>
<td></td>
<td>Structure Diagram</td>
<td>Object-Oriented Design</td>
<td>[IEEE 1016]</td>
</tr>
<tr>
<td></td>
<td>Object Diagram</td>
<td>OML</td>
<td>[IEEE 12207]</td>
</tr>
<tr>
<td></td>
<td>Class Specification</td>
<td>Modular Design</td>
<td>[IEEE 42010]</td>
</tr>
<tr>
<td></td>
<td>Data Model</td>
<td>Integrated Development Environment (IDE)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Database Management System (DBMS)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Design Review</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refinement</td>
<td></td>
</tr>
</tbody>
</table>
Software Metric

A software metric is a quantitative measure of the degree a software system, component, or process possesses a given attribute. Because of the abstract nature of software and special problems with software schedule, cost, and quality, data collection and the derived metrics are an essential part of software engineering. This is evidenced by the repeated reference to measurement and metrics in the SWEBOK. Table 2 describes software metrics that are collected and used in different areas of software development. As in Table 1 the list is not meant to be complete, but to illustrate the type and range of measures used in practice.

**Table 2: Software Metrics** *(SEBoK Original)*

<table>
<thead>
<tr>
<th>Category</th>
<th>Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management Metrics</td>
<td>• Size: Lines of Code (LOC*), Thousand Lines of Code (KLOC)</td>
</tr>
<tr>
<td></td>
<td>• Size: Function points, Feature Points</td>
</tr>
<tr>
<td></td>
<td>• Individual Effort: hours</td>
</tr>
<tr>
<td></td>
<td>• Task Completion Time: hours, days, weeks</td>
</tr>
<tr>
<td></td>
<td>• Project Effort: person-hours</td>
</tr>
<tr>
<td></td>
<td>• Project Duration: months</td>
</tr>
<tr>
<td></td>
<td>• Schedule: earned value</td>
</tr>
<tr>
<td></td>
<td>• Risk Projection: risk description, risk likelihood, risk impact</td>
</tr>
<tr>
<td>Software Quality Metrics</td>
<td>• Defect Density - Defects/KLOC (e.g., for system test)</td>
</tr>
<tr>
<td></td>
<td>• Defect Removal Rate – defect removed/hour (for review and test)</td>
</tr>
<tr>
<td></td>
<td>• Test Coverage</td>
</tr>
<tr>
<td></td>
<td>• Failure Rate</td>
</tr>
</tbody>
</table>
Software Requirements Metrics
• Change requests received, open, closed
• Change request frequency
• Effort required to implement a requirement change
• Status of requirements traceability
• User stories in the backlog

Software Design Metrics
• Cyclomatic Complexity
• Weighted Methods per Class
• Cohesion - Lack of Cohesion of Methods
• Coupling - Coupling Between Object Classes
• Inheritance - Depth of Inheritance Tree, Number of Children

Software Maintenance and Operation
• Mean Time Between Changes (MTBC)
• Mean Time To Change (MTTC)
• System Reliability
• System Availability
• Total Hours of Downtime

*Note: Even though the LOC metric is widely used, using it comes with some problems and concerns: different languages, styles, and standards can lead to different LOC counts for the same functionality; there are a variety of ways to define and count LOC—source LOC, logical LOC, with or without comment lines, etc.; and automatic code generation has reduced the effort required to produce LOC.

References

Works Cited


Additional References


SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].
The goal of project management is to plan and coordinate the work activities needed to deliver a satisfactory product, service, or enterprise endeavor within the constraints of schedule, budget, resources, infrastructure, and available staffing and technology. The purpose of this knowledge area (KA) is to acquaint systems engineers with the elements of project management and to explain the relationships between systems engineering (SE) and project management (PM).

**Topics**

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- The Nature of Project Management
- An Overview of the PMBOK® Guide
- Relationships between Systems Engineering and Project Management
- The Influence of Project Structure and Governance on Systems Engineering and Project Management
- Procurement and Acquisition

**References**

**Works Cited**

None.

**Primary References**


**Additional References**

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.8 released 27 March 2017

**SEBoK Discussion**

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*
The Nature of Project Management

While *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)* provides an overview of project management for those seeking PMI certification, Fairley (2009) and Forsberg (2005) suggest another way to characterize the important aspects of project management:

- Planning and Estimating
- Measuring and Controlling
- Leading and Directing
- Managing Risk

Introduction

Project managers and systems engineers are both concerned with management issues such as planning, measuring and controlling, leading, directing, and managing risk. In the case of project managers, the project attributes to be managed include project plans; estimates; schedule; budget; project structure; staffing; resources; infrastructure; and risk factors. Product attributes managed by systems engineers include items such as requirements allocation and flow-down; system architecture; structure of and interactions among technical teams; specialty engineering; integration; verification; and validation.

The exact allocation of the SE and PM duties depend on many factors, such as customer and stakeholder interactions, organizational structure of the parent organization, and relationships with affiliate contractors and subcontractors. (See the article on The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships in this KA.)

Planning and Estimating

Planning

Planning a project involves providing answers to the who, what, where, when, and why of every project:

- **Who:** Addresses staffing issues (competencies, numbers of staff, communication and coordination)
- **What:** Addresses the scope of activities
- **Where:** Addresses issues of locale (local, geographically distributed)
- **When:** Addresses scheduling issues
- **Why:** Addresses rationale for conducting a project

Guidance for developing project plans can be found in INCOSE (2012), NASA (2007), and ISO/IEC/IEEE Standard 16326:2009. It is often observed that communication and coordination among stakeholders during project planning are equally as important as (and sometimes more important than) the documented plan that is produced.

In defense work, event-driven integrated master plans and time-driven integrated master schedules are planning products. Chapter 11 of the Defense Acquisition Guidebook provides details (DAU 2010).
Estimating

Estimation is an important element of planning. An estimate is a projection from past to future, adjusted to account for differences between past and future. Estimation techniques include analogy, rule of thumb, expert judgment, and use of parametric models such as the PRICE model for hardware, COCOMO for software projects and COSYSMO for systems projects (Stewart 1990; Boehm et al. 2000; Valerdi 2008).

Entities estimated include (but are not limited to) schedule; cost; performance; and risk.

Systems engineering contributes to project estimation efforts by ensuring that

- the overall system life cycle is understood;
- dependencies on other systems and organizations are identified;
- the logical dependencies during development are identified; and
- resources and key skills are identified and planned.

Additionally, high-level system architecture and risk assessment provide the basis for both the work breakdown structure and the organizational breakdown structure.

Measuring and Controlling

Measuring and controlling are the key elements of executing a project. Measurement includes collecting measures for work products and work processes. For example, determining the level of coverage of requirements in a design specification can be assessed through review, analysis, prototyping, and traceability. Effort and schedule expended on the work processes can be measured and compared to estimates; earned value tracking can be used for this purpose. Controlling is concerned with analyzing measurement data and implementing corrective actions when actual status does not align with planned status.

Systems engineers may be responsible for managing all technical aspects of project execution, or they may serve as staff support for the project manager or project management office. Organizational relationships between systems engineers and project managers are presented in Team Capability. Other organizational considerations for the relationships between systems engineering and project management are covered in the Enabling Systems Engineering knowledge area.

Additional information on measurement and control of technical factors can be found in the Measurement and Assessment and Control articles in Part 3: Systems Engineering and Management.

Leading and Directing

Leading and directing requires communication and coordination among all project stakeholders, both internal and external. Systems engineers may be responsible for managing all technical aspects of project execution, or they may serve as staff support for the project manager or project management office. Organizational relationships between systems engineers and project managers are presented in the article Team Capability in Part 5. Other organizational considerations for the relationships between systems engineering and project management are discussed in Part 5: Enabling Systems Engineering.
Managing Risk

Risk management is concerned with identifying and mitigating potential problems before they become real problems. Systems engineering projects are, by nature, high-risk endeavors because of the many unknowns and uncertainties that are inherent in projects. Because new risk factors typically emerge during a project, ongoing continuous risk management is an important activity for both systems engineers and project managers.

Potential and actual problems may exist within every aspect of a project. Systems engineers are typically concerned with technical risk and project managers with programmatic risk. Sometimes, technical risk factors are identified and confronted by systems engineers and programmatic risk factors are identified and confronted by project managers without adequate communication between them. In these cases, appropriate tradeoffs among requirements, schedule, budget, infrastructure, and technology may not be made, which creates additional risk for the successful outcome of a project.

In the last ten years, there has been an increasing interest in opportunity management as the converse of risk management. Hillson (2003), Olsson (2007), and Chapman and Ward (2003) provide highly cited introductions. Additional information on risk management for systems engineering projects can be found in the Risk Management article in Part 3: Systems Engineering and Management.

References

Works Cited


Primary References


Additional References


SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox[1].
An Overview of the PMBOK® Guide

The Guide to the Project Management Book of Knowledge (PMBOK® Guide) is published and maintained by the Project Management Institute (PMI). It is acknowledged as the authoritative documentation of good practices in project management. It is also the basis for certification exams to qualify Project Management Professionals (PMPs). Many organizations require PMP certification as a basic qualification for the role of project manager.

Overview

According to Section 1.3 of the PMBOK® Guide, project management is accomplished through the appropriate application and integration of the 47 logically grouped project management processes, which are categorized into five Process Groups (PMI 2013). The five Process Groups are

1. Initiating Process Group
2. Planning Process Group
3. Executing Process Group
4. Monitoring and Controlling Process Group
5. Closing Process Group

Each of the 47 processes is specified by Inputs, Tools & Techniques, and Outputs. Data flow diagrams are used in the PMBOK to illustrate the relationships between each process and the other processes in which each process interacts. The processes are also grouped into ten Knowledge Areas. These Knowledge Areas are

1. Project Integration Management
2. Project Scope Management
3. Project Time Management
4. Project Cost Management
5. Project Quality Management
6. Project Human Resources Management
7. Project Communications Management
8. Project Risk Management
9. Project Procurement Management
10. Project Stakeholder Management

The five process groups are discussed in more detail next.

Initiating Process Group

Activities performed in the Initiating process group include obtaining authorization to start a project; defining the high-level scope of the project; developing and obtaining approval for the project charter; performing key stakeholder analysis; and identifying and documenting high-level risks, assumptions, and constraints. The Initiating process group contains two processes: develop the project charter and identify stakeholders.

Planning Process Group

The Planning process group consists of 24 processes, including assessing detailed project requirements, constraints, and assumptions with stakeholders; developing the project management plan; creating the work breakdown structure; developing a project schedule; determining a project budget; and planning for quality management, human resource management, communication management, change and risk management, procurement management, and stakeholder management. The integrated project management plan is presented to key stakeholders.
Executing Process Group

The Executing process group includes eight processes that involve performing the work necessary to achieve the stated objectives of the project. Activities include obtaining and managing project resources; executing the tasks defined in the project plan; implementing approved changes according to the change management plan; performing quality assurance; acquiring, developing, and managing the project team; managing communications; conducting procurements; and managing stakeholder engagement.

Monitoring and Controlling Process Group

The Monitoring and Controlling process group is comprised of 11 processes that include validate and control scope; control schedule; control cost; control quality; control communications, control risks; control procurements; and control stakeholder engagement. Activities include measuring project performance and using appropriate tools and techniques; managing changes to the project scope, schedule, and costs; ensuring that project deliverables conform to quality standards; updating the risk register and risk response plan; assessing corrective actions on the issues register; and communicating project status to stakeholders.

Closing Process Group

The Closing process group involves two processes: closing project or phase and closing procurements. Closing the project or phase involves finalizing all project activities, archiving documents, obtaining acceptance for deliverables, and communicating project closure. Other activities include transferring ownership of deliverables; obtaining financial, legal, and administrative closure; distributing the final project report; collating lessons learned; archiving project documents and materials; and measuring customer satisfaction.

The scope of project management, as specified in the PMBOK Guide, encompasses the total set of management concerns that contribute to successful project outcomes.

References

Works Cited

Primary References

Additional References
Relationships between Systems Engineering and Project Management

This topic discusses the relationship between systems engineering (SE) and project management (PM). As with software engineering, there is a great deal of overlap. Depending on the environment and organization, the two disciplines can be disjoint, partially intersecting, or one can be seen as a subset of the other. While there is no standard relationship, the project manager and the systems engineer encompass the technical and managerial leadership of a project between them, which requires the enterprise of each project manager and system engineer to work out the particular details for their own context.

Overlap

There is a great deal of significant overlap between the scope of systems engineering, as described here (in the SEBoK), CMMI (2011), and other resources and the scope of project management, as described in the PMBOK® Guide (PMI 2013), CMMI (2011), and other resources as illustrated in Figure 1.
These sources describe the importance of understanding the scope of the work at hand, how to plan for critical activities, how to manage efforts while reducing risk, and how to successfully deliver value to a customer. The systems engineer working on a project will plan, monitor, confront risk, and deliver the technical aspects of the project, while the project manager is concerned with the same kinds of activities for the overall project. Because of these shared concerns, at times there may be confusion and tension between the roles of the project manager and the systems engineer on a given project. As shown in Figure 2, on some projects, there is no overlap in responsibility. On other projects, there may be shared responsibilities for planning and managing activities. In some cases, particularly for smaller projects, the project manager may also be the lead technical member of the team performing both roles of project manager and systems engineer.

![Figure 2. Overlap of Project Roles. (SEBoK Original)](image)

**Defining Roles and Responsibilities**

Regardless of how the roles are divided up on a given project, the best way to reduce confusion is to explicitly describe the roles and responsibilities of the project manager and the systems engineer, as well as other key team members. The Project Management Plan (PMP) and the Systems Engineering Management Plan (SEMP) are key documents used to define the processes and methodologies the project will employ to build and deliver a product or service.

The PMP is the master planning document for the project. It describes all activities, including technical activities, to be integrated and controlled during the life of the program. The SEMP is the master planning document for the systems engineering technical elements. It defines SE processes and methodologies used on the project and the relationship of SE activities to other project activities. The SEMP must be consistent with, and evolve in concert, with the PMP. In addition, some customers have technical management plans and expectations that the project’s SEMP integrate with customer plans and activities. In the U.S. Department of Defense, most government project teams have a systems engineering plan (SEP) with an expectation that the contractor’s SEMP will integrate and remain consistent with customer technical activities. In cases where the project is developing a component of a larger system, the component project’s SEMP will need to integrate with the overall project’s SEMP.

Given the importance of planning and managing the technical aspects of the project, an effective systems engineer will need to have a strong foundation in management skills and prior experience, as well as possess strong technical depth. From developing and defending basis of estimates, planning and monitoring technical activities, identifying and mitigating technical risk, and identifying and including relevant stakeholders during the life of the project, the systems engineer becomes a key member of the project’s management and leadership team. Additional information on Systems Engineering Management and Stakeholder Needs and Requirements can be found in Part 3: Systems Engineering and Management.
Practical Considerations

Effective communication between the project manager and the system engineer is essential for mission accomplishment. This communication needs to be established early, and occur frequently. Resource reallocation, schedule changes, product/system changes and impacts, risk changes: all these and more need to be quickly and clearly discussed between the PM and SE.

References

Works Cited


Primary References


Additional References


SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox[1].
The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships

This article reviews various project structures that impact or provide governance to the project and that require key involvement from the program manager and the systems engineer. These structures include: the structure of the organization itself (functional, project, matrix, and specialized teams, such as Integrated Product Teams (IPTs), Change Control Boards (CCBs), and Engineering Review Boards (ERBs). This article also addresses the influence of schedule-driven versus requirements-driven projects on these structures.

The Relationships between Systems Engineering and Project Management is covered in a related article.

An Overview of Project Structures

Project management and systems engineering governance are dependent on the organization's structure. For some projects, systems engineering is subordinated to project management and in other cases, project management provides support to systems engineering. These alternatives are illustrated in Figures 1 and 2 of the Organizing the Team section in Team Capability.

A project exists within the structural model of an organization. Projects are one-time, transient events that are initiated to accomplish a specific purpose and are terminated when the project objectives are achieved. Sometimes, on small projects, the same person accomplishes the work activities of both project management and systems engineering. Because the nature of the work activities are significantly different, it is sometimes more effective to have two persons performing project management and systems engineering, each on a part-time basis. On larger projects there are typically too many tasks to be accomplished for one person to accomplish all of the necessary work. Very large projects may have project management and systems engineering offices with a designated project manager and a designated lead systems engineer.

Projects are typically organized in one of three ways: (1) by functional structure, (2) by project structure, and (3) by a matrix structure (see Systems Engineering Organizational Strategy for a fourth structure and related discussion). In a function-structured organization, workers are grouped by the functions they perform. The systems engineering functions can be: (1) distributed among some of the functional organizations, (2) centralized within one organization or (3) a hybrid, with some of the functions being distributed to the projects, others centralized and others are distributed to functional organization. The following figure provides an organizational structure continuum and illustrates levels of governance among the functional organizations and the project.

- In a functional-structured organization, the project manager is a coordinator and typically has only limited control over the systems engineering functions. In this type of organization, the functional manager typically controls the project budget and has authority over the project resources. However, the organization may or may not have a functional unit for systems engineering. In the case where there is a functional unit for systems engineering, systems engineers are assigned across existing projects. Trades can be made among their projects to move the priority of a specific systems engineering project ahead of other projects; thus, reducing the nominal schedule for that selected project. However, in the case where there is not a functional unit for systems engineering, the project manager may have to find alternate sources of staffing for systems engineering – for example, hiring systems engineering talent or consultants, or may consider promoting or expanding the responsibilities of a current team member, etc.

- In a project-structured organization, the project manager has full authority and responsibility for managing the budget and resources to meet the schedule requirements. The systems engineer is subject to the direction of the project manager. The project manager may work with human resources or a personnel manager or may go outside
the organization to staff the project.

- Matrix-structured organization can have the advantages of both the functional and project structures. For a schedule driven project, function specialists are assigned to projects as needed to work for the project manager to apply their expertise on the project. Once they are no longer needed, they are returned to their functional groups (e.g. home office). In a weak matrix, the functional managers have authority to assign workers to projects and project managers must accept the workers assigned to them. In a strong matrix, the project manager controls the project budget and can reject workers from functional groups and hire outside workers if functional groups do not have sufficient available and trained workers.

![Organizational Continuum Diagram](image)

*Figure 1. The Organizational Continuum (2). (SEBoK Original and Adapted from Fairley 2009). Reprinted with permission of the IEEE Computer Society. All other rights are reserved by the copyright owner.*

In all cases, it is essential that the organizational and governance relationships be clarified and communicated to all project stakeholders and that the project manager and systems engineer work together in a collegial manner.

The Project Management Office (PMO) provides centralized control for a set of projects. The PMO is focused on meeting the business objectives leveraging a set of projects, while the project managers are focused on meeting the objectives of those projects that fall under their purview. PMOs typically manage shared resources and coordinate communication across the projects, provide oversight and manage interdependencies, and drive project-related policies, standards, and processes. The PMO may also provide training and monitor compliance (PMI 2013).
Schedule-Driven versus Requirements-Driven Influences on Structure and Governance

This article addresses the influences on governance relationships between the project manager and the systems engineer. One factor that establishes this relationship is whether a project is schedule-driven or requirements-driven. In general, a project manager is responsible for delivering an acceptable product/service on the specified delivery date and within the constraints of the specified schedule, budget, resources, and technology.

The systems engineer is responsible for collecting and defining the operational requirements, specifying the systems requirements, developing the system design, coordinating component development teams, integrating the system components as they become available, verifying that the system to be delivered is correct, complete and consistent to its technical specification, and validating the operation of the system in its intended environment.

From a governance perspective, the project manager is often thought of as being a movie producer who is responsible for balancing the schedule, budget, and resource constraints to meet customer satisfaction. The systems engineer is responsible for product content; ergo, the systems engineer is analogous to a movie director.

Organizational structures, discussed previously, provide the project manager and systems engineer with different levels of governance authority. In addition, schedule and requirements constraints can influence governance relationships. A schedule-driven project is one for which meeting the project schedule is more important than satisfying all of the project requirements; in these cases lower priority requirements may not be implemented in order to meet the schedule.

Classic examples of these types of projects are:

- a project that has an external customer with a contractual delivery date and an escalating late delivery penalty, and
- a project for which delivery of the system must meet a major milestone (e.g. a project for an announced product release of a cell phone that is driven by market considerations).

For schedule-driven projects, the project manager is responsible for planning and coordinating the work activities and resources for the project so that the team can accomplish the work in a coordinated manner to meet the schedule. The systems engineer works with the project manager to determine the technical approach that will meet the schedule. An Integrated Master Schedule (IMS) is often used to coordinate the project.

A requirements-driven project is one for which satisfaction of the requirements is more important than the schedule constraint. Classic examples of these types of projects are:

1. exploratory development of a new system that is needed to mitigate a potential threat (e.g. military research project) and
2. projects that must conform to government regulations in order for the delivered system to be safely operated (e.g., aviation and medical device regulations).

An Integrated Master Plan is often used to coordinate event-driven projects.

To satisfy the product requirements, the systems engineer is responsible for the making technical decisions and making the appropriate technical trades. When the trade space includes cost, schedule, or resources, the systems engineer interacts with the project manager who is responsible for providing the resources and facilities needed to implement a system that satisfies the technical requirements.

Schedule-driven projects are more likely to have a management structure in which the project manager plays the central role, as depicted in Figure 1 of the Organizing the Team section in Team Capability. Requirement-driven projects are more likely to have a management structure in which the systems engineer plays the central role, as depicted in Figure 2 of the Organizing the Team section in Team Capability.

Along with the Project Management Plan and the Systems Engineering Management Plan, IMP/IMS are critical to this process.
Related Structures

Integrated Product Teams (IPTs), Change Control Boards (CCBs), and Engineering Review Boards (ERBs) are primary examples of project structures that play a significant role in project governance and require coordination between the project manager, systems engineer and other members of the team.

Integrated Product Team

The Integrated Product Team (IPT) ensures open communication flow between the government and industry representatives as well as between the various product groups (see Good Practices in Planning). There is typically a top level IPT, sometimes referred to as the Systems Engineering and Integration Team (SEIT) (see Systems Engineering Organizational Strategy), that oversees the lower level IPTs. The SEIT can be led by either the project manager for a specific project or by the systems engineering functional manager or functional lead across many projects. Each IPT consists of representatives from the appropriate management and technical teams that need to collaborate on systems engineering, project management, and other activities to create a high quality product. These representatives meet regularly to ensure that the technical requirements are understood and properly implemented in the design. Also see Team Capability.

Change Control Board

An effective systems engineering approach includes a disciplined process for change control as part of the larger goal of configuration management. The primary objective of configuration management is to track changes to project artifacts that include software, hardware, plans, requirements, designs, tests, and documentation. Alternatively, a Change Control Board (CCB) with representatives from appropriate areas of the project is set up to effectively analyze, control and manage changes being proposed to the project. The CCB typically receives an Engineering Change Proposal (ECP) from design/development, production, or operations/support and initially reviews the change for feasibility. The ECP may also be an output of the Engineering Review Board (ERB) (see next section). If determined feasible, the CCB ensures there is an acceptable change implementation plan and proper modification and installation procedures to support production and operations.

There may be multiple CCBs in a large project. CCBs may be comprised of members from both the customer and the supplier. As with the IPTs, there can be multiple levels of CCB starting with a top level CCB with CCBs also existing at the subsystem levels. A technical lead typically chairs the CCB; however, the board includes representation from project management since the CCB decisions will have an impact on schedule, budget, and resources.

See Figure 2 under Configuration Management for a flow of the change control process adapted from Blanchard and Fabrycky (2011). See also Capability Updates, Upgrades, and Modernization, and topics included under Enabling Teams. See also the UK West Coast Route Modernisation Project Vignette which provides an example where change control was an important success factor.

Engineering Review Board

Another example of a board that requires collaboration between technical and management is the Engineering Review Board (ERB). Examples of ERBs include the Management Safety Review Board (MSRB) (see Safety Engineering. Responsibilities of the ERB may include technical impact analysis of pending change requests (like the CCB), adjudication of results of engineering trade studies, and review of changes to the project baseline. In some cases the ERB may be the management review board and the CCB may be the technical review board. Alternatively, in a requirement driven organization the ERB may have more influence while in a schedule driven organization the CCB may have more impact.
The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships

References

Works Cited


Primary References

Additional References
None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.8 released 27 March 2017

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

*If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox[^1].*

[^1]: ENCODED_CONTENT

NzgzNzQPGRpdibpZD0iZGlzdvzX3RoYcmvHlZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXEND_ENCODED_CONTENT
Procurement and Acquisition

Procurement is the act of buying goods and services. Acquisition covers the conceptualization, initiation, design, development, testing, contracting, production, deployment, logistics support, modification, and disposal of weapons and other systems, as well as supplies or services (including construction) to satisfy organizational needs intended for use in, or in support of, defined missions (DAU 2010; DoD 2001).

Acquisition covers a much broader range of topics than procurement. Acquisition spans the whole life cycle of acquired systems. The procurement of appropriate systems engineering (SE) acquisition activities and levels of SE support is critical for an organization to meet the challenge of developing and maintaining complex systems.

The Guide for Integrating Systems Engineering into DoD Acquisition Contracts addresses how systems engineering activities are integrated into the various elements of acquisition and procurement (DoD 2006a).

Acquisition Process Model

Multiple acquisition process models exist. An acquisition process for major systems in industry and defense is shown in Figure 1. The process of acquisition is defined by a series of phases during which technology is defined and matured into viable concepts. These concepts are subsequently developed and readied for production, after which the systems produced are supported in the field.

Acquisition planning is the process of identifying and describing needs, capabilities, and requirements, as well as determining the best method for meeting those requirements (e.g., program acquisition strategy). This process includes procurement; thus, procurement is directly linked to the acquisition process model. The process model present in Figure 1 allows a given acquisition to enter the process at any of the development phases.

For example, a system using unproven technology would enter at the beginning stages of the process and would proceed through a lengthy period of technology maturation. On the other hand, a system based on mature and proven technologies might enter directly into engineering development or sometimes even production.

![Figure 1. An Acquisition Process Model (DAU 2010). Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).](image-url)
Systems Engineering Role in the Acquisition Process

The procurement of complex systems usually requires a close relationship between the offeror and supplier SE teams due to the breadth and depth of SE activities. SE is an overarching process that the program team applies in order to transition from a stated capability need to an affordable, operationally effective, and suitable system. SE is important to every phase of the acquisition process. SE encompasses the application of SE processes across the acquisition life cycle and is intended to be an integrating mechanism for balanced solutions addressing capability needs, design considerations, and constraints. It is also intended to address limitations imposed by technology, budget, and schedule.

SE is an interdisciplinary approach; that is, it is a structured, disciplined, and documented technical effort to simultaneously design and develop system products and processes to satisfy the needs of the customer. Regardless of the scope and type of program, or at what point it enters the program acquisition life cycle, the technical approach to the program needs to be integrated with the acquisition strategy to obtain the best program solution.

Acquisition and procurement in the commercial sector have many characteristics in common with their counterparts in the realm of government contracting, although the processes in the commercial world are usually accomplished with fewer rigors than occur between government and contractor interactions. Offshore outsourcing is commonly practiced in the commercial software arena with the goal of reducing the cost of labor. Commercial organizations sometimes subcontract with other commercial organizations to provide missing expertise and to balance the ebb and flow of staffing needs.

In some cases, relations between the contracting organization and the subcontractor are strained because of the contracting organization’s desire to protect its intellectual property and development practices from potential exposure to the subcontractor. Commercial organizations often have lists of approved vendors that are used to expedite the procurement of needed equipment, products, and services. In these situations, commercial organizations have processes to evaluate and approve vendors in ways that are analogous to the qualification of government contractors. Many commercial organizations apply SE principles and procedures even though they may not identify the personnel and job functions as “systems engineers” or “systems engineering.”

Importance of the Acquisition Strategy in the Procurement Process

The acquisition strategy is usually developed during the front end of the acquisition life cycle. (For an example of this, see the Technology Development Phase in Figure 1.) The acquisition strategy provides the integrated strategy for all aspects of the acquisition program throughout the program life cycle.

In essence, the acquisition strategy is a high-level business and technical management approach designed to achieve program objectives within specified resource constraints. It acts as the framework for planning, organizing, staffing, controlling, and leading a program, as well as for establishing the appropriate contract mechanisms. It provides a master schedule for research, development, testing, production, fielding, and other SE related activities essential for program success, as well as for formulating functional strategies and plans.

The offeror’s program team, including systems engineering, is responsible for developing and documenting the acquisition strategy, which conveys the program objectives, direction, and means of control based on the integration of strategic, technical, and resource concerns. A primary goal of the acquisition strategy is the development of a plan that will minimize the time and cost of satisfying an identified, validated need while remaining consistent with common sense and sound business practices. While the contract officer (CO) is responsible for all contracting aspects, including determining which type of contract is most appropriate, and following the requirements of existing regulations, directives, instructions, and policy memos of an organization, the program manager (PM) works with the CO to develop the best contract/procurement strategy and contract types.
**Relating Acquisition to Request for Proposal and Technical Attributes**

There are several formats for requesting proposals from offerors for building complex systems. Figure 2 relates acquisition program elements to a representative request for proposal (RFP) topical outline and key program technical attributes that have been used by the Department of Defense. In general, programs have a better chance of success when both the offeror and supplier understand the technical nature of the program and the need for the associated SE activities.

The offeror and supplier need to clearly communicate the technical aspect of the program throughout the procurement process. The offeror's RFP and the associated supplier proposal represent one of the formal communications paths. A partial list of key program technical attributes is presented in Figure 2.

![Figure 2. Relating Acquisition to Request for Proposal and Technical Attributes. (DoD 2006a). Released by the U.S. Office of the Secretary of Defense.](image)

**Contract-Related Activities and the Offeror’s Systems Engineering and Project Management Roles**

A clear understanding of the technical requirements is enhanced via the development of a Systems Engineering Plan (SEP). The SEP documents the system engineering strategy for a project or program and acts as the blueprint for the conduct, management, and control of the technical aspects of the acquisition program (DoD 2011). The SEP documents the SE structure and addresses government and contractor boundaries. It also summarizes the program’s selected acquisition strategy. It identifies and links to program risks. It also describes how the contractor's, and sometimes the subcontractor's and suppliers’, technical efforts are to be managed.

Once the technical requirements are understood, a contract may be developed and followed by the solicitation of suppliers. The offeror's PM, chief or lead systems engineer, and CO must work together to translate the program’s acquisition strategy and associated technical approach (usually defined in a SEP) into a cohesive, executable
Table 1 shows some key contracting-related tasks with indicators of the roles of the PM and LSE.

**Table 1. Offeror’s Systems Engineering and Program Management Roles (DoD 2006).**
Released by the U.S. Office of the Secretary of Defense.

<table>
<thead>
<tr>
<th>Typical Contract-Related Activities</th>
<th>System Engineer and Project Manager Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Identify overall procurement requirements and associated budget. Describe the offer’s needs and any constraints on the procurement.</td>
<td>Lead system engineer (LSE) provides program technical requirements. PM provides any programmatic related requirements.</td>
</tr>
<tr>
<td>2. Identify technical actions required to successfully complete technical and procurement milestones. The program’s SEP is the key source for capturing this technical planning.</td>
<td>LSE defines the technical strategy/approach and required technical efforts. This should be consistent with the program’s Acquisition Strategy.</td>
</tr>
<tr>
<td>3. Document market research results and identify potential industry sources.</td>
<td>PM and LSE identify programmatic and technical information needed and assist in evaluating the results.</td>
</tr>
<tr>
<td>4. Prepare a Purchase Request, including product descriptions; priorities, allocations and allotments; architecture; government-furnished property or equipment (or Government-Off-The-Shelf (GOTS); government-furnished information; information assurance and security considerations; and required delivery schedules.</td>
<td>PM and LSE ensure the specific programmatic and technical needs are defined clearly (e.g., commercial-off-the-shelf (COTS) products).</td>
</tr>
<tr>
<td>5. Identify acquisition streamlining approach and requirements, budgeting and funding, management information requirements, environmental considerations, offeror’s expected skill sets, and milestones. These should be addressed in the Acquisition Strategy.</td>
<td>The procurement team work together, but the CO has the prime responsibility. The PM is the owner of the program Acquisition Strategy. The LSE develops and reviews (and the PM approves) the technical strategy.</td>
</tr>
<tr>
<td>6. Plan the requirements for the contract Statement of Objectives (SOO) / Statement of Work (SOW) / specification, project technical reviews, acceptance requirements, and schedule.</td>
<td>LSE is responsible for the development of the technical aspects of the SOO/SOW.</td>
</tr>
<tr>
<td>7. Plan and conduct Industry Days as appropriate.</td>
<td>PM and LSE supports the CO in planning the meeting agenda to ensure technical needs are discussed.</td>
</tr>
<tr>
<td>8. Establish contract cost, schedule, and performance reporting requirements. Determine an incentive strategy and appropriate mechanism (e.g., Award Fee Plan and criteria).</td>
<td>LSE provides technical resource estimates. LSE supports development of the Work Breakdown Structure (WBS) based on preliminary system specifications, determines event-driven criteria for key technical reviews, and determines what technical artifacts are baselined. The PM and LSE advise the CO in developing the metrics/criteria for an incentive mechanism.</td>
</tr>
<tr>
<td>9. Identify data requirements.</td>
<td>LSE identifies all technical Contractor Data Requirements List (CDRL) and technical performance expectations.</td>
</tr>
<tr>
<td>10. Establish warranty requirements, if applicable.</td>
<td>LSE works with the CO to determine cost-effective warranty requirements.</td>
</tr>
<tr>
<td>11. Prepare a Source Selection Plan (SSP) and RFP (for competitive contracts).</td>
<td>PM and LSE provide input to the SSP per the SOO/SOW.</td>
</tr>
<tr>
<td>12. Conduct source selection and award the contract to the successful offeror.</td>
<td>PM and LSE participate on evaluation teams.</td>
</tr>
</tbody>
</table>
Offeror and Supplier Interactions

There should be an environment of open communication prior to the formal source selection process. This ensures that the supplier understands the offeror's requirements and that the offeror understands the supplier's capabilities and limitations, as well as enhancing the supplier's involvement in the development of a program acquisition strategy. During the pre-solicitation phase, the offeror develops the solicitation and may ask suppliers to provide important insights into the technical challenges, program technical approach, and key business motivations.

For example, potential bidders could be asked for their assessment of a proposed system's performance based on the maturity level of new and existing technologies.

Contracts and Subcontracts

Typical types of contracts include the following:

- **Fixed Price**: In a fixed price contract the offeror proposes a single price for all products and services to implement the project. This single price is sometimes referred to as low bid or lump sum. A fixed price contract transfers the project risks to the supplier. When there is a cost overrun, the supplier absorbs it. If the supplier performs better than planned, their profit is higher. Since all risks are absorbed by the supplier, a fixed price bid may be higher to reflect this.

- **Cost-reimbursement [Cost plus]**: In a cost-reimbursement contract the offeror provides a fixed fee, but also reimburses the contractor for labor, material, overhead, and administration costs. Cost-reimbursement type contracts are used when there is a high level of project risk and uncertainty. With this type of contract the risks reside primarily with the offeror. The supplier gets reimbursed for all of its costs. Additional costs that arise due to changes or rework are covered by the offeror. This type of contract is often recommended for the system definition of hardware and software development when there is a risk of stakeholder changes to the system.

- **Subcontracts**: A subcontractor performs work for another company as part of a larger project. A subcontractor is hired by a general contractor (also known as a prime or main contractor) to perform a specific set of tasks as part of the overall project. The incentive to hire subcontractors is either to reduce costs or to mitigate project risks. The systems engineering team is involved in establishing the technical contract requirements, technical selection criteria, acceptance requirements, and the technical monitoring and control processes.

- **Outsource contracts**: Outsourced contracts are used to obtain goods or services by contracting with an outside supplier. Outsourcing usually involves contracting a business function, such as software design and code development, to an external provider.

- **Exclusively Commercial Off-the-Shelf (COTS)**: Exclusively COTS contracts are completely satisfied with commercial solutions that require no modification for use. COTS solutions are used in the environment without modifying the COTS system. They are integrated into an existing user's platform or integrated into an existing operational environment. The systems engineering team is involved in establishing the technical contract requirements, technical acceptance, and technical selection criteria.

- **Integrated COTS**: Integrated COTS contracts use commercially available products and integrate them into existing user platforms or operational environments. In some cases, integrated COTS solutions modify the system's solution. The cost of integrating the commercial COTS product into the operational environment can exceed the cost of the COTS product itself. As a result, the systems engineering team is usually involved in establishing the technical outsourcing contract requirements, technical selection criteria, technical monitoring and control processes, and technical acceptance and integration processes.

- **COTS Modification**: COTS modification requires the most time and cost because of the additional work needed to modify the COTS product and integrate it into the system. Depending on how complex and critical the need is, the systems engineering team is usually involved in establishing the technical outsourcing contract requirements, technical selection criteria, technical monitoring and control processes, and technical acceptance requirements.
• **IT services**: IT services provide capabilities that can enable an enterprise, application, or Web service solution. IT services can be provided by an outsourced service provider. In many cases, the user interface for these Web services is as simple as a Web browser. Depending on how complex and critical the needs are, the systems engineering team can be involved in establishing the technical outsourcing contract requirements, technical selection criteria, and technical acceptance process.

**References**

**Works Cited**


**Primary References**


**Additional References**


---

SEBoK v. 1.8 released 27 March 2017

**SEBoK Discussion**

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

*If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox*[^1].
Systems Engineering and Industrial Engineering

*Industrial Engineering is concerned with the design, improvement and installation of integrated systems of people, materials, information, equipment and energy. It draws upon specialized knowledge and skill in the mathematical, physical, and social sciences together with the principles and methods of engineering analysis and design, to specify, predict, and evaluate the results to be obtained from such systems.* (IIE 1992)

Industrial engineering (IE) encompasses several aspects of systems engineering (SE) (i.e., production planning and analysis, continuous process improvement, etc.) and also many elements of the engineered systems domain (production control, supply chain management, operations planning and preparation, operations management, etc.), as depicted in Figure 3 of the article Scope and Context of the SEBoK.

This knowledge area covers the overarching aspects of industrial engineering and describes the synergies between IE and SE.

Overview of Industrial Engineering

Industrial engineers are trained to design and analyze the components of which man-machine systems are composed. They bring together individual elements that are designed via other engineering disciplines and properly synergize these subsystems together with the people components for a completely integrated man-machine system. Industrial engineers are focused on the improvement of any system that is being designed or evaluated. They make individual human tasks more productive and efficient by optimizing flow, eliminating unnecessary motions, utilizing alternate materials to improve manufacturing, improving the flow of product through processes, and optimizing the configuration of work spaces. Fundamentally, the industrial engineer is charged with reducing costs and increasing profitability through ensuring the efficient use of human, material, physical, and/or financial resources (Salvendy 2001).

A systems engineer leverages industrial engineering knowledge to provide:

- production planning and analysis
- systems integration
- lifecycle planning and estimating
- change analysis and management
- continuous process improvement
- quality assurance
- business case analysis / return on investment
- engineering management
- systems integration

Industrial engineers complement systems engineers with knowledge in:

- supply chain management
- budgeting and economic analysis
- production line preparation
- production
- production control
- testing
Industrial Engineering Body of Knowledge

The current overview of the industrial engineering body of knowledge is provided in the *Handbook of Industrial Engineering* (Salvendy 2001) and *Maynard's Industrial Engineering Handbook* (Zandin 2001). The Institute of Industrial Engineers (IIE 1992) is currently in the process of developing a specific industrial engineering body of knowledge. Additionally, industrial engineering terminology defines specific terms related to the industrial engineering profession. Definitions used in this section are from this reference. Turner et al. (1992) provide an overview of industrial and systems engineering.

The elements of IE include the following:

**Operations Engineering**

Operations engineering involves the management and control aspects of IE and works to ensure that all the necessary requirements are in place to effectively execute a business. Key areas of knowledge in this field include: product and process life cycles, forecasting, project scheduling, production scheduling, inventory management, capacity management, supply chain, distribution, and logistics. Concepts such as materials requirements planning and enterprise resource planning find their roots in this domain.

**Operations Research**

Operations research is the organized and systematic analysis of complex situations, such as if there is a spike in the activities of organizations of people and resources. The analysis makes use of certain specific disciplinary methods, such as probability, statistics, mathematical programming, and queuing theory. The purpose of operations research is to provide a more complete and explicit understanding of complex situations, to promote optimal performance utilizing the all the resources available. Models are developed that describe deterministic and probabilistic systems and these models are employed to aid the decision maker. Knowledge areas in operations research include linear programming, network optimization, dynamic programming, integer programming, nonlinear programming, metaheuristics, decision analysis and game theory, queuing systems, and simulation. Classic applications include the transportation problem and the assignment problem.

**Production Engineering / Work Design**

Production engineering is the design of a production or manufacturing process for the efficient and effective creation of a product. Included in this knowledge area is classic tool and fixture design, selection of machines to produce product, and machine design. Closely related to production engineering, work design involves such activities as process, procedural and work area design, which are geared toward supporting the efficient creation of goods and services. Knowledge in work simplification and work measurement are crucial to work design. These elements form a key foundation, along with other knowledge areas in IE, for lean principles.
Facilities Engineering and Energy Management

Facilities engineering involves attempting to achieve the optimal organization in factories, buildings, and offices. In addition to addressing the aspects of the layout inside a facility, individuals in this field also possess knowledge of material and equipment handling as well as storage and warehousing. This area also involves the optimal placement and sizing of facilities according to the activities they are required to contain. An understanding of code compliance and use of standards is incorporated. The energy management aspect of this area encompasses atmospheric systems and lighting and electrical systems. Through the development of responsible management of resources in the energy management domain, industrial engineers have established a basis in sustainability.

Ergonomics

Ergonomics is the application of knowledge in the life sciences, physical sciences, social sciences, and engineering that studies the interactions between the human and the total working environment, such as atmosphere, heat, light and sound, as well as the interactions of all tools and equipment in the workplace. Ergonomics is sometimes referred to as human factors. Individuals in this field have a specialized knowledge in areas such as: anthropometric principles, standing/sitting, repetitive task analysis, work capacity and fatigue, vision and lighting, hearing, sound, noise, vibration, human information processing, displays and controls, and human-machine interaction. Members in this field also consider the organizational and social aspects of a project.

Engineering Economic Analysis

Engineering economic analysis concerns techniques and methods that estimate output and evaluate the worth of commodities and services relative to their costs. Engineering economic analysis is used to evaluate system affordability. Fundamental to this knowledge area are value and utility, classification of cost, time value of money and depreciation. These are used to perform cash flow analysis, financial decision making, replacement analysis, break-even and minimum cost analysis, accounting and cost accounting. Additionally, this area involves decision making involving risk and uncertainty and estimating economic elements. Economic analysis also addresses any tax implications.

Quality and Reliability

Quality is the totality of features and characteristics of a product or service that bear on its ability to satisfy stated or implied needs. Reliability is the ability of an item to perform a required function under stated conditions for a stated period of time. The understanding of probability and statistics form a key foundation to these concepts. Knowledge areas in quality and reliability include: quality concepts, control charts, lot acceptance sampling, rectifying inspection and auditing, design of experiments, and maintainability. Six sigma has its roots in the quality domain; however, its applicability has grown to encompass a total business management strategy.

Engineering Management

Engineering management refers to the systematic organization, allocation, and application of economic and human resources in conjunction with engineering and business practices. Knowledge areas include: organization, people, teamwork, customer focus, shared knowledge systems, business processes, resource responsibility, and external influences.

Supply Chain Management

Supply chain management deals with the management of the input of goods and services from outside sources that are required for a business to produce its own goods and services. Information is also included as a form of input. Knowledge areas include: building competitive operations, planning and logistics, managing customer and supplier relationships, and leveraging information technology to enable the supply chain.
References

Works Cited


Primary References


Additional References

Operations Engineering


Operations Research


Production Engineering / Work Design

Facilities Engineering and Energy Management

Ergonomics

Engineering Economic Analysis

Quality & Reliability


**Engineering Management**


**Supply Chain Management**


---

< Previous Article | Parent Article | Next Article >

**SEBoK v. 1.8 released 27 March 2017**

**SEBoK Discussion**

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

*If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox* [1].

ENCODED_CONTENT

ODg4OTEGRpciBpZDiZGkzXVz3RocmVhZCI+PC9kaXY+CjxzY3JpcdHwZT0idGV4dC9qYXZhc2Nya:

END_ENCODED_CONTENT
Specialty engineering disciplines support product, service and enterprise development by applying crosscutting knowledge to system design decisions, balancing total system performance and affordability. This knowledge area presents several of the supporting engineering disciplines with a focus on the systems engineer.

To help develop a consistent picture of how SE is related to these specialty disciplines the SEBoK uses a standard set of headings: Description, Discipline Management, Discipline Relationships, Personnel Considerations, Metrics, Models, Tools, and References. Please note that not all of these sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Topics
Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. The KAs in turn are divided into topics. This KA contains the following topics:

- Reliability, Availability, and Maintainability
- Human Systems Integration
- Security Engineering
- Electromagnetic Interference/Electromagnetic Compatibility
- Resilience Engineering
- Manufacturability and Producibility
- Affordability
- Environmental Engineering
- Logistics Engineering

Specialty Requirements
The systems engineering team must ensure that specialty requirements are properly reviewed with regard to their impact on life cycle costs, development schedule, technical performance, and operational utility. For example, security requirements can impact operator workstations, electromagnetic interference requirements can impact the signal in the interfaces between subsystems, and mass-volume requirements may preclude the use of certain materials to reduce subsystem weight.

Engineering specialists audit the evolving design and resulting configuration items to ensure that the overall system performance also satisfies the specialty requirements. Including appropriate specialty engineers within each systems engineering team assures that all system requirements are identified and balanced throughout the development cycle.

Integration of Specialty Engineering
Integration of engineering specialties into a project or program is, or should be, a major objective of systems engineering management. With properly implemented procedures, the rigor of the systems engineering process ensures participation of the specialty disciplines at key points in the technical decision making process. Special emphasis on integration is mandatory because a given design could in fact be accomplished without consideration of these "specialty" disciplines, leading to the possibility of system ineffectiveness or failure when an unexamined situation occurs in the operational environment.

For example, human factors considerations can contribute to reduced workloads and therefore lower error rates by operators in aircraft cockpits, at air-traffic consoles, or nuclear reactor stations. Similarly, mean-time-to-repair features can significantly increase overall system availability in challenging physical environments, such as mid-ocean or outer space. Specialty engineering requirements are often manifest as constraints on the overall system.
design space. The role of system engineering is to balance these constraints with other functionality in order to harmonize total system performance. The end goal is to produce a system that provides utility and effectiveness to the customer at an affordable price.

As depicted in Figure 1, systems engineering plays a leadership role in integrating traditional disciplines, specialty disciplines, and unique system product demands to define the system design. Relationships for this integration process are represented as interactions among three filters.

The first filter is a conceptual analysis that leverages traditional design consideration (structural, electronics, aerodynamics, mechanical, thermodynamics, and other). The second filter evaluates the conceptual approach using specialty disciplines, such as safety, affordability, quality assurance, human factors, reliability and maintainability, producibility, packaging, test, logistics, and others, to further requirements development. Design alternatives that pass through these two processes go through a third filter that incorporates facility design, equipment design, procedural data, computer programs, and personnel to develop the final requirements for design selection and further detailed development.

![Integration Process for Specialty Engineering (USAF 2000).](image)

**Figure 1. Integration Process for Specialty Engineering (USAF 2000).** Released by the U.S. Air Force.
References

Works Cited


Primary References


Additional References

None.

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK  as a whole, please see the SEBoK Sandbox [1].
Reliability, Availability, and Maintainability

Reliability, availability, and maintainability (RAM) are three system attributes that are of tremendous interest to systems engineers, logisticians, and users. Collectively, they affect economic life-cycle costs of a system and its utility.

Overview

Reliability, maintainability, and availability (RAM) are three system attributes that are of great interest to systems engineers, logisticians, and users. Collectively, they affect both the utility and the life-cycle costs of a product or system. The origins of contemporary reliability engineering can be traced to World War II. The discipline's first concerns were electronic and mechanical components (Ebeling, 2010). However, current trends point to a dramatic rise in the number of industrial, military, and consumer products with integrated computing functions. Because of the rapidly increasing integration of computers into products and systems used by consumers, industry, governments, and the military, reliability must consider both hardware, and software.

Maintainability models present some interesting challenges. The time to repair an item is the sum of the time required for evacuation, diagnosis, assembly of resources (parts, bays, tool, and mechanics), repair, inspection, and return. Administrative delay (such as holidays) can also affect repair times. Often these sub-processes have a minimum time to complete that is not zero, resulting in the distribution used to model maintainability having a threshold parameter.

A threshold parameter is defined as the minimum probable time to repair. Estimation of maintainability can be further complicated by queuing effects, resulting in times to repair that are not independent. This dependency frequently makes analytical solution of problems involving maintainability intractable and promotes the use of simulation to support analysis.

System Description

This section sets forth basic definitions, briefly describes probability distributions, and then discusses the role of RAM engineering during system development and operation. The final subsection lists the more common reliability test methods that span development and operation.

Basic Definitions

Reliability

defined as the probability of a system or system element performing its intended function under stated conditions without failure for a given period of time (ASQ 2011). A precise definition must include a detailed description of the function, the environment, the time scale, and what constitutes a failure. Each can be surprisingly difficult to define as precisely as one might wish.
Maintainability
defined as the probability that a system or system element can be repaired in a defined environment within a specified period of time. Increased maintainability implies shorter repair times (ASQ 2011).

Availability
probability that a repairable system or system element is operational at a given point in time under a given set of environmental conditions. Availability depends on reliability and maintainability and is discussed in detail later in this topic (ASQ 2011).

A failure is the event(s), or inoperable state, in which any item or part of an item does not, or would not, perform as specified (GEIA 2008). The failure mechanism is the physical, chemical, electrical, thermal, or other process that results in failure (GEIA 2008). In computerized system, a software defect or fault can be the cause of a failure (Laprie 1992) and the failure may have been preceded by an error which was internal to the item. The failure mode is the way or the consequence of the mechanism through which an item fails (GEIA 2008, Laprie 1992.). The severity of the failure mode is the magnitude of its impact (Laprie, 1992).

Probability Distributions used in Reliability Analysis
Reliability can be thought of as the probability of the survival of a component until time t. Its complement is the probability of failure before or at time t. If we define a random variable T as the time te to failure, then

\[
R(t) = 1 - F(t)
\]

where R(t) is the reliability and F(t) is the failure probability. The failure probability is the cumulative distribution function (CDF) of a mathematical probability distribution. Continuous distributions used for this purpose include exponential, Weibull, log-normal, and generalized gamma. Discrete distributions such as the Bernoulli, Binomial, and Poisson are used for calculating the expected number of failures or for single probabilities of success.

The same continuous distributions used for reliability can also be used for maintainability although the interpretation is different (i.e., probability that a failed component is restored to service prior to time t). However, predictions of maintainability may have to account for processes such as administrative delays, travel time, sparing, and staffing and can therefore most complex.

The probability distributions used in reliability and maintainability estimation are referred to as models because they only provide estimates of the true failure and restoration of the items under evaluation. Ideally, the values of the parameters used in these models would be estimated from life testing or operating experience. However, performing such tests or collecting credible operating data once items are fielded can be costly. Therefore, approximations use of data from "similar systems", "engineering judgment", other methods are sometimes used. As a result, that estimates based on limited data may be very imprecise. Testing methods to gather such data are discussed below.

RAM Considerations during Systems Development
RAM are inherent product or system attributes that should be considered throughout the development lifecycle. Reliability standards, textbook authors, and others have proposed multiple development process models (O'Connor 2014, Kapur 2014, Ebeling 2010, DoD 2005). The discussion in this section relies on a standard developed by a joint effort by the Electronic Industry Association and the U.S. Government and adopted by the U.S. Department of Defense (GEIA 2008) that defines 4 processes: understanding user requirements and constraints, design for reliability, production for reliability, and monitoring during operation and use (discussed in the next section).
Understanding User Requirements and Constraints

Understanding user requirements involves eliciting information about functional requirements, constraints (e.g., mass, power consumption, spatial footprint, life cycle cost), and needs that correspond to RAM requirements. From these emerge system requirements that should include specifications for reliability, maintainability, and availability, and each should be conditioned on the projected operating environments. RAM requirements definition is as challenging but as essential to development success as is the definition of general functional requirements.

Design for Reliability

System designs based on user requirements and system design alternatives can then be formulated and evaluated. Reliability engineering during this phase seeks to increase system robustness through measures such as redundancy, diversity, built-in test, advanced diagnostics, and modularity to enable rapid physical replacement. In addition, it may be possible to reduce failure rates through measures such as use of higher strength materials, increasing the quality components, moderating extreme environmental conditions, or shortened maintenance, inspection, or overhaul intervals. Design analyses may include mechanical stress, corrosion, and radiation analyses for mechanical components, thermal analyses for mechanical and electrical components, and Electromagnetic Interference (EMI) analyses or measurements for electrical components and subsystems.

In most computer based systems, hardware mean time between failures are hundreds of thousands of hours so that most system design measures will be to increase system reliability are focused on software. The most obvious way to improve software reliability is by improving its quality through more disciplined development efforts and test. Methods for doing so are in the scope of software engineering but not in the scope of this section. However, reliability and availability can also be increased through architectural redundancy, independence, and diversity. Redundancy must be accompanied by measures to ensure data consistency, and managed failure detection and switchover. Within the software architecture, measures such as watchdog timers, flow control, data integrity checks (e.g., hashing or cyclic redundancy checks), input and output validity checking, retries, and restarts can increase reliability and failure detection coverage (Shooman, 2002).

System RAM characteristics should be continuously evaluated as the design progresses. Where failure rates are not known (as is often the case for unique or custom developed components, assemblies, or software), developmental testing may be undertaken assess the reliability of custom-developed components. Evaluations based on quantitative analyses assess the numerical reliability and availability of the system and are usually based on reliability block diagrams, fault trees, Markov models, and Petri nets (O’Connor, 2011). Markov models and Petri nets are of particular value for computer-based systems that use redundancy. Evaluations based on qualitative analyses assess vulnerability to single points of failure, failure containment, recovery, and maintainability. The primary qualitative methods is the failure mode effects and criticality analyses (FMECA) (Kececioglu 1991). The development program Discrepancy Reporting (DR) or Failure Reporting and Corrective Action System (FRACAS) should also be used to identify failure modes which may not have been anticipated by the FMECA and to identify common problems that can be corrected through an improved design or development process.

Analyses from related disciplines during design time also affect RAM. Human factor analyses are necessary to ensure that operators and maintainers can interact with the system in a manner that minimizes failures and the restoration times when they do occur. There is also a strong link between RAM and cybersecurity in computer based systems. On the one hand defensive measures reduce the frequency of failures due to malicious events. On the other, devices such as firewalls, policy enforcement devices, and access/authentication serves (also known as "directory servers") can also become single points of failure or performance bottlenecks that reduce system reliability and availability.
Production for Reliability

Many production issues associated with RAM are related to quality. The most important of these are ensuring repeatability and uniformity of production processes and complete unambiguous specifications for items from the supply chain. Other are related to design for manufacturability, storage, and transportation (Kapur, 2014; Eberlin 2010). Large software intensive systems information systems are affected by issues related to configuration management, integration testing, and installation testing. Testing and recording of failures in the problem reporting and corrective action systems (PRACAS) or the FRACAS capture data on failures and improvements to correct failures. Depending on organizational considerations, this may be the same or a separate system as used during the design.

Monitoring During Operation and Use

After systems are fielded, their reliability and availability to assess whether system or product has met its RAM objectives, to identify unexpected failure modes, to record fixes, to assess the utilization of maintenance resources, and to assess the operating environment. The FRACAS or a maintenance management database may be used for this purpose. In order to assess RAM, it is necessary to maintain an accurate record not only of failures but also of operating time and the duration of outages. Systems that report only on repair actions and outage incidents may not be sufficient for this purpose.

An organization should have an integrated data system that allows reliability data to be considered with logistical data, such as parts, personnel, tools, bays, transportation and evacuation, queues, and costs, allowing a total awareness of the interplay of logistical and RAM issues. These issues in turn must be integrated with management and operational systems to allow the organization to reap the benefits that can occur from complete situational awareness with respect to RAM.

Reliability and Maintainability Testing

Reliability Testing can be performed at the component, subsystem, and system level throughout the product or system lifecycle. Examples of hardware related categories of reliability testing include (Ebeling, 2010, O’Connor 2014)

- Reliability Life Tests: Reliability Life Tests are used to empirically assess the time to failure for non-repairable products and systems and the times between failure for repairable or restorable systems. Termination criteria for such tests can be based on a planned duration or planned number of failures. Methods to account for “censoring” of the failures or the surviving units enable a more accurate estimate of reliability.

- Accelerated Life Tests: Accelerated life testing is performed by subjecting the items under test (usually electronic parts) by increasing the temperature to well above the expecting operating temperature and extrapolating results using an Arhenius relation.

- Highly Accelerated Life Testing/Highly Accelerated Stress Testing (HALT/HASS) subjects units under test (components or subassemblies) to extreme temperature and vibration tests with the objective of identifying failure modes, margins, and design weaknesses.

- Parts Screening: Parts screening is not really a test but a procedure to operate components for a duration beyond the “infant mortality” period during which less durable items fail and the more durable parts that remain are then assembled into the final product or system.

Examples of system level testing (including both hardware and software) are (O’Connor 2014, Ebeling 2010)

Stability tests: Stability tests are life tests for integrated hardware and software systems. The goal of such testing is to determine the integrated system failure rate and assess operational suitability. Test conditions must include accurate simulation of the operating environment (including workload) and a means of identifying and recording failures.
• Reliability Growth Tests: Reliability Growth Testing is part of a reliability growth program in which items are
tested throughout the development and early production cycle with the intent of assessing reliability increases due
to improvements in the manufacturing process (for hardware) or software quality (for software).

• Failure/recovery tests: Such testing assesses the fault tolerance of a system by measuring probability of
switchover for redundant systems. Failures are simulated and the ability of the hardware and software to detect
the condition and reconfigure the system to remain operational are tested.

• Maintainability Tests: Such testing assess the system diagnostics capabilities, physical accessibility, and
maintainer training by simulating hardware or software failures that require maintainer action for restoration.

Because of its potential impact on cost and schedule, reliability testing should be coordinated with the overall system
engineering effort. Test planning considerations include the number of test units, duration of the tests, environmental
conditions, and the means of detecting failures.

Data Issues

True RAM models for a system are generally never known. Data on a given system is assumed or collected, used to
select a distribution for a model, and then used to fit the parameters of the distribution. This process differs
significantly from the one usually taught in an introductory statistics course.

First, the normal distribution is seldom used as a life distribution, since it is defined for all negative times. Second,
and more importantly, reliability data is different from classic experimental data. Reliability data is often censored,
based, observational, and missing information about covariates such as environmental conditions. Data from testing
is often expensive, resulting in small sample sizes. These problems with reliability data require sophisticated
strategies and processes to mitigate them.

One consequence of these issues is that estimates based on limited data can be very imprecise.

Discipline Management

In most large programs, RAM experts report to the system engineering organization. At project or product
conception, top level goals are defined for RAM based on operational needs, lifecycle cost projections, and warranty
cost estimates. These lead to RAM derived requirements and allocations that are approved and managed by the
system engineering requirements management function. RAM testing is coordinated with other product or system
testing through the testing organization, and test failures are evaluated by the RAM function through joint meetings
such as a Failure Review Board. In some cases, the RAM function may recommend design or development process
changes as a result of evaluation of test results or software discrepancy reports, and these proposals must be
adjudicated by the system engineering organization, or in some cases, the acquiring customer if cost increases are
involved.

Post-Production Management Systems

Once a system is fielded, its reliability and availability should be tracked. Doing so allows the producer / owner to
verify that the design has met its RAM objectives, to identify unexpected failure modes, to record fixes, to assess the
utilization of maintenance resources, and to assess the operating environment.

One such tracking system is generically known as a FRACAS system (Failure Reporting and Corrective Action
System). Such a system captures data on failures and improvements to correct failures. This database is separate
from a warranty data base, which is typically run by the financial function of an organization and tracks costs only.

A FRACAS for an organization is a system, and itself should be designed following systems engineering principles.
In particular, a FRACAS system supports later analyses, and those analyses impose data requirements. Unfortunately, the lack of careful consideration of the backward flow from decision to analysis to model to required
data too often leads to inadequate data collection systems and missing essential information. Proper prior planning
prevents this poor performance.

Of particular importance is a plan to track data on units that have not failed. Units whose precise times of failure are unknown are referred to as censored units. Inexperienced analysts frequently do not know how to analyze censored data, and they omit the censored units as a result. This can bias an analysis.

An organization should have an integrated data system that allows reliability data to be considered with logistical data, such as parts, personnel, tools, bays, transportation and evacuation, queues, and costs, allowing a total awareness of the interplay of logistical and RAM issues. These issues in turn must be integrated with management and operational systems to allow the organization to reap the benefits that can occur from complete situational awareness with respect to RAM.

**Discipline Relationships**

**Interactions**

- RAM interacts with nearly all aspects of the system development effort. Specific dependencies and interactions include:
  
  - Systems Engineering: RAM interacts with the system engineering as described in the previous section.
  
  - Product Management (Life Cycle Cost and Warranty): RAM interacts with the product or system lifecycle cost and warranty management organizations by assisting in the calculation of expected repair rates, downtimes, and warranty costs. RAM may work with those organizations to perform tradeoff analyses to determine the most cost efficient solution and to price service contracts.
  
  - Quality Assurance: RAM may also interact with the procurement and quality assurance organizations with respect to selection and evaluation of materials, components, and subsystems.

**Dependencies**

- Systems Safety: RAM and system safety engineers have many common concerns with respect to managing the failure behavior of a system (i.e., single points of failure and failure propagation). RAM and safety engineers use similar analysis techniques, with safety being concerned about failures affecting life or unique property and RAM being concerned with those failures as well as lower severity events that disrupt operations. RAM and safety are both concerned with failures occurring during development and test – FRACAS is the primary methodology used for RAM; hazard tracking is the methodology used for system safety.

- Cybersecurity: In systems or products integrating computers and software, cybersecurity and RAM engineers have common concerns relating to the availability of cyberdefenses and system event monitoring. However, there are also tradeoffs with respect to access control, boundary devices, and authentication where security device failures could impact the availability of the product or system to users.

- Software and Hardware Engineering: Design and RAM engineers have a common goal of creating dependable product and systems. RAM interacts with the software and hardware reliability functions through design analyses such as failure modes and effects analyses, reliability predictions, thermal analyses, reliability measurement, and component specific analyses. RAM may recommend design changes as a result of these analyses that may have to be adjudicated by program management, the customer, or system engineering if there are cost or schedule impacts.

- Testing: RAM interacts with the testing program during planning to assess the most efficient (or feasible) test events to perform life testing, failure/recovery testing, and stability testing as well as to coordinate requirements for reliability or stress tests. RAM also interacts with the testing organization to assess test results and analyze failures for the implications on product or system RAM.
• Logistics: RAM works with logistics in providing expected failure rates and downtime constraints in order for logistics engineers to determine staffing, sparing, and special maintenance equipment requirements.

**Discipline Standards**

Because of the importance of reliability, availability, and maintainability, as well as related attributes, there are hundreds of standards associated. Some are general but more are specific to domains such as automotive, aviation, electric power distribution, nuclear energy, rail transportation, software, and many others. Standards are produced by both governmental agencies and professional associations, and international standards bodies such as

- The International Electrotechnical Commission (IEC), Geneva, Switzerland and the closely associated International Standards Organization (ISO)
- The Institute of Electrical and Electronic Engineers (IEEE), New York, NY, USA
- The Society of Automotive Engineers (SAE), Warrendale, PA, USA
- Governmental Agencies – primarily in military and space systems

The following table lists selected standards from each of these agencies. Because of differences in domains and because many standards handle the same topic in slightly different ways, selection of the appropriate requires consideration of previous practices (often documented as contractual requirements), domain specific considerations, certification agency requirements, end user requirements (if different from the acquisition or producing organization), and product or system characteristics.

<table>
<thead>
<tr>
<th>Organization</th>
<th>Number, Title, and Year</th>
<th>Domain</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEC</td>
<td>IEC 60812 Analysis techniques for system reliability - Procedure for failure mode and effects analysis (FMEA), 2006</td>
<td>General</td>
<td></td>
</tr>
<tr>
<td>IEC</td>
<td>IEC 61703 Mathematical expressions for reliability, availability, maintainability and maintenance, 2001</td>
<td>General</td>
<td></td>
</tr>
<tr>
<td>IEC</td>
<td>IEC 62308, Equipment reliability - Reliability assessment methods, 2006</td>
<td>General</td>
<td></td>
</tr>
<tr>
<td>IEC</td>
<td>IEC 62347, Guidance on system dependability specifications, 2006</td>
<td>General</td>
<td></td>
</tr>
<tr>
<td>IEC</td>
<td>IEC 62278 Railway applications – Specification and demonstration of reliability, availability, maintainability and safety (RAMS), 2002</td>
<td>Railways</td>
<td></td>
</tr>
<tr>
<td>SAE</td>
<td>ARP 4754A, Guidelines for the Development of Civil Aircraft and Systems, 2010</td>
<td>Aviation</td>
<td></td>
</tr>
<tr>
<td>SAE</td>
<td>ARP 5890, Guidelines for Preparing Reliability Assessment Plans for Electronic Engine Controls, 2011</td>
<td>Aviation</td>
<td></td>
</tr>
<tr>
<td>SAE</td>
<td>J1213/2- Use of Model Verification and Validation in Product Reliability and Confidence Assessments, 2011</td>
<td>General</td>
<td></td>
</tr>
</tbody>
</table>
Reliability, Availability, and Maintainability


SAE JA 1002, Software Reliability Program Standard, 2012 Software


U.S. Government MIL HDBK 470A Designing And Developing Maintainable Products And Systems, 1997 Defense Systems

U.S. Government MIL HDBK 217F (Notice 2), Reliability Prediction of Electronic Equipment, 1995 Defense Systems Although formally titled a "Handbook" and more than 2 decades old, the values and methods constitute a de facto standard for some U.S. military acquisitions


Personnel Considerations

Becoming a reliability engineer requires education in probability and statistics as well as the specific engineering domain of the product or system under development or in operation. A number of universities throughout the world have departments of reliability engineering (which also address maintainability and availability) and more have research groups and courses in reliability and safety – often within the context of another discipline such as computer science, system engineering, civil engineering, mechanical engineering, or bioengineering. Because most academic engineering programs do not have a full reliability department, most engineers working in reliability have been educated in other disciplines and acquire the additional skills through additional coursework or by working with other qualified engineers. A certification in reliability engineering is available from the American Society for Quality (ASQ 2016). However, only a minority of engineers working in the discipline have this certification.

Metrics

The three basic metrics of RAM are (not surprisingly) Reliability, Maintainability, and Availability. Reliability can be characterized in terms of the parameters, mean, or any percentile of a reliability distribution. However, in most cases, the exponential distribution is used, and a single value, the mean time to failure (MTTF) for non-restorable systems, or mean time between failures (MTBF for restorable systems are used). The metric is defined as

\[ \{MTTF|MTBF\} = \frac{T_{op, total}}{n_{fails}} \]

where \(T_{op}\), \(T_{fails}\) the total operating time and \(n_{fails}\) is the number of failures

Maintainability is often characterized in terms of the exponential distribution and the mean time to repair and be similarly calculated, i.e.,

\[ MTTR = \frac{T_{down, total}}{n_{outages}} \]

Where \(T_{down, total}\) the total down time and \(n_{outages}\) is the number of outages.

As was noted above, accounting for downtime requires definitions and specificity. Down time might be counted only for corrective maintenance actions, or it may include both corrective and preventive maintenance actions. Where the lognormal rather than the exponential distribution is used, a mean down time can still be calculated, but both the log of the downtimes and the variance must be known in order to fully characterize maintainability. Availability can be calculated from the total operating time and the downtime, or in the alternative, as a function of MTBF and MTTR (Mean Time To Repair) as
As was the case with maintainability, availability may be qualified as to whether it includes only unplanned failures and repairs (inherent availability) or downtime due to all causes including administrative delays, staffing outages, or spares inventory deficiencies (operational availability).

Probabilistic metrics describe system performance for RAM. Quantiles, means, and modes of the distributions used to model RAM are also useful.

Availability has some additional definitions, characterizing what downtime is counted against a system. For **inherent availability**, only downtime associated with corrective maintenance counts against the system. For **achieved availability**, downtime associated with both corrective and preventive maintenance counts against a system. Finally, **operational availability** counts all sources of downtime, including logistical and administrative, against a system.

Availability can also be calculated instantaneously, averaged over an interval, or reported as an asymptotic value. **Asymptotic availability** can be calculated easily, but care must be taken to analyze whether or not a systems settles down or settles up to the asymptotic value, as well as how long it takes until the system approaches that asymptotic value.

**Reliability importance** measures the effect on the system reliability of a small improvement in a component’s reliability. It is defined as the partial derivative of the system reliability with respect to the reliability of a component.

**Criticality** is the product of a component’s reliability, the consequences of a component failure, and the frequency with which a component failure results in a system failure. Criticality is a guide to prioritizing reliability improvement efforts.

Many of these metrics cannot be calculated directly because the integrals involved are intractable. They are usually estimated using simulation.

### Models

There are a wide range of models that estimate and predict reliability (Meeker and Escobar 1998). Simple models, such as exponential distribution, can be useful for 'back of the envelope' calculations.

System models are used to (1) combine probabilities or their surrogates, failure rates and restoration times, at the component level to find a system level probability or (2) to evaluate a system for maintainability, single points of failure, and failure propagation. The three most common are reliability block diagrams, fault trees, and failure modes and effects analyses.

There are more sophisticated probability models used for life data analysis. These are best characterized by their failure rate behavior, which is defined as the probability that a unit fails in the next small interval of time, given it has lived until the beginning of the interval, and divided by the length of the interval.

Models can be considered for a fixed environmental condition. They can also be extended to include the effect of environmental conditions on system life. Such extended models can in turn be used for accelerated life testing (ALT), where a system is deliberately and carefully overstressed to induce failures more quickly. The data is then extrapolated to usual use conditions. This is often the only way to obtain estimates of the life of highly reliable products in a reasonable amount of time (Nelson 1990).

Also useful are **degradation models**, where some characteristic of the system is associated with the propensity of the unit to fail (Nelson 1990). As that characteristic degrades, we can estimate times of failure before they occur.

The initial developmental units of a system often do not meet their RAM specifications. **Reliability growth models** allow estimation of resources (particularly testing time) necessary before a system will mature to meet those goals (Meeker and Escobar 1998).
Maintainability models describe the time necessary to return a failed repairable system to service. They are usually the sum of a set of models describing different aspects of the maintenance process (e.g., diagnosis, repair, inspection, reporting, and evacuation). These models often have threshold parameters, which are minimum times until an event can occur.

Logistical support models attempt to describe flows through a logistics system and quantify the interaction between maintenance activities and the resources available to support those activities. Queue delays, in particular, are a major source of down time for a repairable system. A logistical support model allows one to explore the trade space between resources and availability.

All these models are abstractions of reality, and so at best approximations to reality. To the extent they provide useful insights, they are still very valuable. The more complicated the model, the more data necessary to estimate it precisely. The greater the extrapolation required for a prediction, the greater the imprecision.

Extrapolation is often unavoidable, because high reliability equipment typically can have long life and the amount of time required to observe failures may exceed test times. This requires strong assumptions be made about future life (such as the absence of masked failure modes) and that these assumptions increase uncertainty about predictions. The uncertainty introduced by strong model assumptions is often not quantified and presents an unavoidable risk to the system engineer.

There are many ways to characterize the reliability of a system, including fault trees, reliability block diagrams, and failure mode effects analysis.

A Fault Tree (Kececioglu 1991) is a graphical representation of the failure modes of a system. It is constructed using logical gates, with AND, OR, NOT, and K of N gates predominating. Fault trees can be complete or partial; a partial fault tree focuses on a failure mode or modes of interest. They allow 'drill down' to see the dependencies of systems on nested systems and system elements. Fault trees were pioneered by Bell Labs in the 1960s.

A Failure Mode Effects Analysis is a table that lists the possible failure modes for a system, their likelihood, and the effects of the failure. A Failure Modes Effects Criticality Analysis scores the effects by the magnitude of the product of the consequence and likelihood, allowing ranking of the severity of failure modes (Kececioglu 1991).
A **Reliability Block Diagram (RBD)** is a graphical representation of the reliability dependence of a system on its components. It is a directed, acyclic graph. Each path through the graph represents a subset of system components. As long as the components in that path are operational, the system is operational. Component lives are usually assumed to be independent in a RBD. Simple topologies include a series system, a parallel system, a $k$ of $n$ system, and combinations of these.

RBDs are often nested, with one RBD serving as a component in a higher level model. These hierarchical models allow the analyst to have the appropriate resolution of detail while still permitting abstraction.

RBDs depict paths that lead to success, while fault trees depict paths that lead to failure.
A **Failure Mode Effects Analysis** is a table that lists the possible failure modes for a system, their likelihood, and the effects of the failure. A **Failure Modes Effects Criticality Analysis** scores the effects by the magnitude of the product of the consequence and likelihood, allowing ranking of the severity of failure modes (Kececioglu 1991).

System models require even more data to fit them well. "Garbage in, garbage out" (GIGO) particularly applies in the case of system models.

**Tools**

The specialized analyses required for RAM drive the need for specialized software. While general purpose statistical languages or spreadsheets can, with sufficient effort, be used for reliability analysis, almost every serious practitioner uses specialized software.

Minitab (versions 13 and later) includes functions for life data analysis. Win Smith is a specialized package that fits reliability models to life data and can be extended for reliability growth analysis and other analyses. Relex has an extensive historical database of component reliability data and is useful for estimating system reliability in the design phase.

There is also a suite of products from ReliaSoft (2007) that is useful in specialized analyses. Weibull++ fits life models to life data. ALTA fits accelerated life models to accelerated life test data. BlockSim models system reliability, given component data.

**Discipline Specific Tool Families**

Reliasoft (http://www.reliasoft.com/products.htm) and PTC Windchill Product Risk and Reliability (http://www.ptc.com/product-lifecycle-management/windchill/product-risk-and-reliability) produce a comprehensive family of tools for component reliability prediction, system reliability predictions (both reliability block diagrams and fault trees), reliability growth analysis, failure modes and effects analyses, FRACAS databases, and other specialized analyses. In addition to these comprehensive tool families, there are more narrowly scoped tools. Minitab (versions 13 and later) includes functions for life data analysis.
General Purpose Statistical Analysis Software with Reliability Support

Some general purpose statistical analysis software include functions for reliability data analysis. Minitab (https://www.minitab.com/en-us/products/minitab/look-inside/) has a module for reliability and survival analysis. SuperSmith (http://www.barringer1.com/wins.htm) is a more specialized package that fits reliability models to life data and can be extended for reliability growth analysis and other analyses. R (https://www.r-project.org/) is a widely used open source and well supported general purpose statistical language with specialized packages that can be used for fitting reliability models, Bayesian analysis, and Markov modeling.

Special Purpose Analysis Tools

Fault tree generation and analysis tools include CAFTA (http://teams.epri.com/RR/News%20Archives/CAFTAFactSheet.pdf) from the Electric Power Research Institute and OpenFTA (http://www.openfta.com/), an open source software tool originally developed by Auvation Software. PRISM (http://www.prismmodelchecker.org/) is an open source probabilistic model checker that can be used for Markov modeling (both continuous and discrete time) as well as for more elaborate analyses of system (more specifically, “timed automata”) behaviors such as communication protocols with uncertainty.

References

Works Cited


American Society for Quality (ASQ) 2016, Reliability Engineering Certification – CRE, information available online at http://asq.org/cert/reliability-engineer


Shooman, Martin, 2002, Reliability of Computer Systems and Networks, New York, John Wiley & Sons
Primary References


Additional References


SAE. 2000b. *Surface Vehicle Recommended Practice J1739: (R) Potential Failure Mode and Effects Analysis in Design (Design FMEA), Potential Failure Mode and Effects Analysis in Manufacturing and Assembly Processes (Process FMEA), and Potential Failure Mode and Effects Analysis for Machinery (Machinery FMEA)*. Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.
Human Systems Integration

Human systems integration (HSI) is an interdisciplinary technical and management process for integrating human considerations with and across all system elements, an essential enabler to systems engineering practice. Human activity considered by HSI includes operating, maintaining, and supporting the system. HSI also considers training and training devices, as well as the infrastructure used for operations and support (DAU 2010). HSI incorporates the following domains as integration considerations: manpower, personnel, training, human factors engineering, occupational health, environment, safety, habitability, and human survivability.

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

Human factors engineering is primarily concerned with designing human-machine interfaces consistent with the physical, cognitive, and sensory abilities of the user population. Human-machine interfaces include: • functional interfaces (functions and tasks, and allocation of functions to human performance or automation); • informational interfaces (information and characteristics of information that provide the human with the knowledge, understanding, and awareness of what is happening in the tactical environment and in the system); • environmental interfaces (the natural and artificial environments, environmental controls, and facility design); • co-operational interfaces (provisions for team performance, cooperation, collaboration, and communication among team members and with other personnel); • organizational interfaces (job design, management structure, command authority, and policies and regulations that impact behavior); • operational interfaces (aspects of a system that support successful operation of the system such as procedures, documentation, workloads, and job aids); • cognitive interfaces (decision rules, decision support systems, provisions for maintaining situational awareness, mental models of the tactical environment,
provisions for knowledge generation, cognitive skills and attitudes, and memory aids); and physical interfaces (hardware and software elements designed to enable and facilitate effective and safe human performance such as controls, displays, workstations, worksites, accesses, labels and markings, structures, steps and ladders, handholds, maintenance provisions, etc.) (DAU 2010).

System Description

HSI is more than human factors, human-computer interaction, or systems engineering. It is an technical and managerial set of processes that involves the consideration and integration of multiple domains. Various organizations represent the HSI domains differently as the number and names of the domains are aligned with existing organizational structures. Booher (2003) presents the seven US Army domains. The Canadian Forces have a different number of domains while the UK Ministry of Defense has another. All the technical work of the domains is present while the number and names and the domains is the same. According to the Defense Acquisition University, the HSI domains are Manpower: Manpower describes the number and mix of personnel required to carry out a task, multiple tasks, or mission in order to operate, maintain, support, and provide training for a system. Manpower factors are those variables that define manpower requirements. These variables include job tasks, operation/maintenance rates, associated workload, and operational conditions (e.g., risk of operator injury) (DAU 2010).

Environment: Environment includes the physical conditions in and around the system, as well as the operational context within which the system will be operated and supported. Environmental attributes include temperature, humidity, noise, vibration, radiation, shock, air quality, among many others. This "environment" affects the human's ability to function as a part of the system (DAU 2010).

Habitability: Habitability factors are those living and working conditions that are necessary to sustain the morale, safety, health, and comfort of the user population. They directly contribute to personnel effectiveness and mission accomplishment and often preclude recruitment and retention problems. Examples include: lighting, space, ventilation, and sanitation; noise and temperature control (i.e., heating and air conditioning); religious, medical, and food services availability; and berthing, bathing, and personal hygiene. Habitability consists of those characteristics of systems, facilities (temporary and permanent), and services necessary to satisfy personnel needs. Habitability factors are those living and working conditions that result in levels of personnel morale, safety, health, and comfort adequate to sustain maximum personnel effectiveness, support mission performance, and avoid personnel retention problems (DAU 2010). Safety: The design features and operating characteristics of a system that serve to minimize the potential for human or machine errors or failure that cause injurious accidents (DAU, 2010). Safety also encompasses the administrative procedures and controls associated with the operations, maintenance, and storage of a system.

Human factors engineering: Human factors engineering is primarily concerned with designing human-machine interfaces consistent with the physical, cognitive, and sensory abilities of the user population. Human-machine interfaces include: functional interfaces (functions and tasks, and allocation of functions to human performance or automation); informational interfaces (information and characteristics of information that provide the human with the knowledge, understanding, and awareness of what is happening in the tactical environment and in the system); environmental interfaces (the natural and artificial environments, environmental controls, and facility design); co-operational interfaces (provisions for team performance, cooperation, collaboration, and communication among team members and with other personnel); organizational interfaces (job design, management structure, command authority, and policies and regulations that impact behavior); operational interfaces (aspects of a system that support successful operation of the system such as procedures, documentation, workloads, and job aids); cognitive interfaces (decision rules, decision support systems, provisions for maintaining situational awareness, mental models of the tactical environment, provisions for knowledge generation, cognitive skills and attitudes, and memory aids); and physical interfaces (hardware and software elements designed to enable and facilitate effective and safe human performance such as controls, displays, workstations, worksites, accesses, labels and markings, structures, steps and
Human survivability: Survivability factors consist of those system design features that reduce the risk of fratricide, detection, and the probability of being attacked, and that enable personnel to withstand man-made hostile environments without aborting the mission, objective, or suffering acute chronic illness, disability, or death. Survivability attributes are those that contribute to the survivability of manned systems (DAU 2010).

Occupational health: Occupational health factors are those system design features that serve to minimize the risk of injury, acute or chronic illness, or disability, and/or reduce job performance of personnel who operate, maintain, or support the system. Prevalent issues include noise, chemical safety, atmospheric hazards (including those associated with confined space entry and oxygen deficiency), vibration, ionizing and non-ionizing radiation, and human factors issues that can create chronic disease and discomfort such as repetitive motion diseases. Many occupational health problems, particularly noise and chemical management, overlap with environmental impacts. Human factors stresses that creating a risk of chronic disease and discomfort overlaps with occupational health considerations (DAU 2010).

Personnel: Personnel factors are those human aptitudes (i.e., cognitive, physical, and sensory capabilities), knowledge, skills, abilities, and experience levels that are needed to properly perform job tasks. Personnel factors are used to develop occupational specialties for system operators, maintainers, trainers, and support personnel (DAU 2010). The selection and assignment of personnel is critical to the success of a system, as determined by the needs set up by various work-related requirements.

Safety: The design features and operating characteristics of a system that serve to minimize the potential for human or machine errors or failure that cause injurious accidents (DAU, 2010). Safety also encompasses the administrative procedures and controls associated with the operations, maintenance, and storage of a system.

Training: Training is the learning process by which personnel individually or collectively acquire or enhance pre-determined job-relevant knowledge, skills, and abilities by developing their cognitive, physical, sensory, and team dynamic abilities. The "training/instructional system" integrates training concepts and strategies, as well as elements of logistic support to satisfy personnel performance levels required to operate, maintain, and support the systems. It includes the "tools" used to provide learning experiences, such as computer-based interactive courseware, simulators, actual equipment (including embedded training capabilities on actual equipment), job performance aids, and Interactive Electronic Technical Manuals (DAU 2010).

Discipline Management
Information to be supplied at a later date.

Discipline Relationships

Interactions
Information to be supplied at a later date.

Dependencies
Information to be supplied at a later date.

Personnel Considerations
Personnel factors are those human aptitudes (i.e., cognitive, physical, and sensory capabilities), knowledge, skills, abilities, and experience levels that are needed to properly perform job tasks. Personnel factors are used to develop occupational specialties for system operators, maintainers, trainers, and support personnel (DAU 2010). The selection and assignment of personnel is critical to the success of a system, as determined by the needs set up by various work-related requirements.
Manpower describes the number and mix of personnel required to carry out a task, multiple tasks, or mission in order to operate, maintain, support, and provide training for a system. Manpower factors are those variables that define manpower requirements. These variables include job tasks, operation/maintenance rates, associated workload, and operational conditions (e.g., risk of operator injury) (DAU 2010).

1. **Training**: Training is the learning process by which personnel individually or collectively acquire or enhance pre-determined job-relevant knowledge, skills, and abilities by developing their cognitive, physical, sensory, and team dynamic abilities. The "training/instructional system" integrates training concepts and strategies, as well as elements of logistic support to satisfy personnel performance levels required to operate, maintain, and support the systems. It includes the "tools" used to provide learning experiences, such as computer-based interactive courseware, simulators, actual equipment (including embedded training capabilities on actual equipment), job performance aids, and Interactive Electronic Technical Manuals (DAU 2010).

**Metrics**
Information to be supplied at a later date.

**Models**
Information to be supplied at a later date.

**Tools**
Information to be supplied at a later date.

**References**

**Works Cited**


**Primary References**

**Additional References**


SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].

References

Safety Engineering

In the most general sense, safety is freedom from harm. As an engineering discipline, system safety is concerned with minimizing hazards that can result in a mishap with an expected severity and with a predicted probability. These events can occur in elements of life-critical systems as well as other system elements. MIL-STD-882E defines system safety as "the application of engineering and management principles, criteria, and techniques to achieve acceptable risk, within the constraints of operational effectiveness and suitability, time, and cost, throughout all phases of the system life cycle" (DoD 2012). MIL-STD-882E defines standard practices and methods to apply as engineering tools in the practice of system safety. These tools are applied to both hardware and software elements of the system in question."

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

System safety engineering focuses on identifying hazards, their causal factors, and predicting the resultant severity and probability. The ultimate goal of the process is to reduce or eliminate the severity and probability of the identified hazards, and to minimize risk and severity where the hazards cannot be eliminated. MIL STD 882E defines a hazard as "A real or potential condition that could lead to an unplanned event or series of events (i.e., mishap) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment." (DoD 2012).

While Systems safety engineering attempt to minimize safety issues throughout the planning and design of systems, mishaps do occur from combinations of unlikely hazards with minimal probabilities. As a result, safety engineering is often performed in reaction to adverse events after deployment. For example, many improvements in aircraft safety come about as a result of recommendations by the National Air Traffic Safety Board based on accident investigations. Risk is defined as "A combination of the severity of the mishap and the probability that the mishap will occur" (DoD 2012, 7). Failure to identify risks to safety, and the according inability to address or "control" these risks, can result in massive costs, both human and economic (Roland and Moriarty 1990)."

System Description

Information to be supplied at a later date.

Discipline Management

Information to be supplied at a later date.

Discipline Relationships

Interactions

Information to be supplied at a later date.

Dependencies

Information to be supplied at a later date.

Discipline Standards
Personnel Considerations

System Safety specialists are typically responsible for ensuring system safety. Air Force Instruction (AFI) provides the following guidance:

9.1 System safety disciplines apply engineering and management principles, criteria, and techniques throughout the life cycle of a system within the constraints of operational effectiveness, schedule, and costs.

9.1.1. System safety is an inherent element of system design and is essential to supporting system requirements. Successful system safety efforts depend on clearly defined safety objectives and system requirements.

9.1.2. System safety must be a planned, integrated, comprehensive effort employing both engineering and management resources.

(SAF 1998, 91-202)

Safety personnel are responsible for the integration of system safety requirements, principles, procedures, and processes into the program and into lower system design levels to ensure a safe and effective interface. Two common mechanisms are the Safety Working Group (SWG) and the Management Safety Review Board (MSRB). The SWG enables safety personnel from all integrated product teams (IPTs) to evaluate, coordinate, and implement a safety approach that is integrated at the system level in accordance with MIL-STD-882E (DoD 2012). Increasingly, safety reviews are being recognized as an important risk management tool. The MSRB provides program level oversight and resolves safety related program issues across all IPTs. Table 1 provides additional information on safety.

<table>
<thead>
<tr>
<th>Ontology Element Name</th>
<th>Ontology Element Attributes</th>
<th>Relationships to Safety</th>
</tr>
</thead>
<tbody>
<tr>
<td>Failure modes</td>
<td>Manner of failure</td>
<td>Required attribute</td>
</tr>
<tr>
<td>Severity</td>
<td>Consequences of failure</td>
<td>Required attribute</td>
</tr>
<tr>
<td>Criticality</td>
<td>Impact of failure</td>
<td>Required attribute</td>
</tr>
<tr>
<td>Hazard Identification</td>
<td>Identification of potential failure modes</td>
<td>Required to determine failure modes</td>
</tr>
<tr>
<td>Risk</td>
<td>Probability of a failure occurring</td>
<td>Required attribute</td>
</tr>
<tr>
<td>Mitigation</td>
<td>Measure to take corrective action</td>
<td>Necessary to determine criticality and severity</td>
</tr>
</tbody>
</table>

Table 1 indicates that achieving System safety involves a close tie between Safety Engineering and other specialty Systems Engineering disciplines such as Reliability and Maintainability Engineering.

System safety engineering focuses on identifying hazards, their causal factors, and predicting the resultant severity and probability. The ultimate goal of the process is to reduce or eliminate the severity and probability of the identified hazards, and to minimize risk and severity where the hazards cannot be eliminated. MIL STD 882E defines a hazard as "A real or potential condition that could lead to an unplanned event or series of events (i.e., mishap) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment." (DoD 2012).

While Systems safety engineering attempt to minimize safety issues throughout the planning and design of systems, mishaps do occur from combinations of unlikely hazards with minimal probabilities. As a result, safety engineering is often performed in reaction to adverse events after deployment. For example, many improvements in aircraft safety come about as a result of recommendations by the National Air Traffic Safety Board based on accident investigations. Risk is defined as "A combination of the severity of the mishap and the probability that the mishap
will occur” (DoD 2012, 7). Failure to identify risks to safety, and the according inability to address or "control" these risks, can result in massive costs, both human and economic (Roland and Moriarty 1990).

Metrics

Information to be supplied at a later date.

Models

Information to be supplied at a later date.

Tools

Information to be supplied at a later date.

References

Works Cited


Primary References
None.

Additional References


Security Engineering

Security engineering is concerned with building systems that remain secure despite malice or error. It focuses on the tools, processes, and methods needed to design and implement complete systems that proactively and reactively mitigate vulnerabilities. Security engineering is a primary discipline used to achieve system assurance.

The term System Security Engineering (SSE) is used to denote this specialty engineering field and the US Department of Defense define it as: "an element of system engineering that applies scientific and engineering principles to identify security vulnerabilities and minimize or contain risks associated with these vulnerabilities" (DODI5200.44, 12).

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

Security engineering incorporates a number of cross-disciplinary skills, including cryptography, computer security, tamper-resistant hardware, applied psychology, supply chain management, and law. Security requirements differ greatly from one system to the next. System security often has many layers built on user authentication, transaction accountability, message secrecy, and fault tolerance. The challenges are protecting the right items rather than the wrong items and protecting the right items but not in the wrong way.

Security engineering is an area of increasing emphasis in the defense domain. Baldwin et al. (2012) provide a survey of the issues and a detailed reference list.

The primary objective of System Security Engineering (SSE) is to minimize or contain defense system vulnerabilities to known or postulated security threats and to ensure that developed systems protect against these threats. Engineering principles and practices are applied during all system development phases to identify and reduce these system vulnerabilities to the identified system threats.

The basic premise of SSE is recognition that an initial investment in “engineering out” security vulnerabilities and “designing-in” countermeasures is a long-term benefit and cost saving measure. Further, SSE provides a means to ensure adequate consideration of security requirements, and, when appropriate, that specific security-related designs are incorporated into the overall system design during the engineering development program. Security requirements include: physical; personnel; procedural; emission; transmission; cryptographic; communications; operations; and, computer security.

There may be some variation in the SSE process from program to program, due mainly to the level of design assurance—that is, ensuring that appropriate security controls have been implemented correctly as planned—required of the contractor. These assurance requirements are elicited early in the program (where they can be adequately planned), implemented, and verified in due course of the system development.

The System Security Engineering Management Plan (SSEMP) is a key document to develop for SSE. The SSEMP identifies the planned security tasks for the program and the organizations and individuals responsible for security aspects of the system. The goals of the SSEMP are to ensure that pertinent security issues are raised at the appropriate points in the program, to ensure adequate precautions are taken during design, implementation, test, and fielding, and to ensure that only an acceptable level of risk is incurred when the system is released for fielding. The SSEMP forms the basis for an agreement with SSE representing the developer, the government program office, the certifier, the accreditor, and any additional organizations that have a stake in the security of the system. The SSEMP identifies the major tasks for certification & accreditation (C&A), document preparation, system evaluation, and engineering; identifies the responsible organizations for each task; and presents a schedule for the completion of those tasks.
SSE security planning and risk management planning includes task and event planning associated with establishing statements of work and detailed work plans as well as preparation and negotiation of SSE plans with project stakeholders. For each program, SSE provides the System Security Plan (SSP) or equivalent. An initial system security Concept of Operations (CONOPS) may also be developed. The SSP provides: the initial planning of the proposed SSE work scope; detailed descriptions of SSE activities performed throughout the system development life cycle; the operating conditions of the system; the security requirements; the initial SSE risk assessment (includes risks due to known system vulnerabilities and their potential impacts due to compromise and/or data loss); and, the expected verification approach and validation results.

These plans are submitted with the proposal and updated as required during engineering development. In the case where a formal C&A is contracted and implemented, these plans comply with the government’s C&A process, certification responsibilities, and other agreement details, as appropriate. The C&A process is the documented agreement between the customer and contractor on the certification boundary. Upon agreement of the stakeholders, these plans guide SSE activities throughout the system development life cycle.

**System Assurance**

NATO AEP-67 (Edition 1), Engineering for System Assurance in NATO Programs, defines system assurance as:

…the justified confidence that the system functions as intended and is free of exploitable vulnerabilities, either intentionally or unintentionally designed or inserted as part of the system at any time during the life cycle... This confidence is achieved by system assurance activities, which include a planned, systematic set of multi-disciplinary activities to achieve the acceptable measures of system assurance and manage the risk of exploitable vulnerabilities. (NATO 2010, 1)

The NATO document is organized based on the life cycle processes in ISO/IEC 15288:2008 and provides process and technology guidance to improve system assurance.

**Software Assurance**

Since most modern systems derive a good portion of their functionality from software, software assurance becomes a primary consideration in systems assurance. The Committee on National Security Systems (CNSS) (2010, 69) defines software assurance as a “level of confidence that software is free from vulnerabilities, either intentionally designed into the software or accidentally inserted at anytime during its lifecycle and that the software functions in the intended manner.”

Goertzel, et. al (2008, 8) point out that "the reason software assurance matters is that so many business activities and critical functions—from national defense to banking to healthcare to telecommunications to aviation to control of hazardous materials—depend on the on the correct, predictable operation of software."

**System Description**

Robust security design explicitly rather than implicitly defines the protection goals. The Certified Information Systems Security Professional (CISSP) Common Body of Knowledge (CBK) partitions robust security into ten domains (Tipton 2006):

1. Information security governance and risk management addresses the framework, principles, policies, and standards that establish the criteria and then assess the effectiveness of information protection. Security risk management contains governance issues, organizational behavior, ethics, and security awareness training.

2. Access control is the procedures and mechanisms that enable system administrators to allow or restrict operation and content of a system. Access control policies determine what processes, resources, and operations users can invoke.
3. Cryptography can be defined as the principles and methods of disguising information to ensure its integrity, confidentiality, and authenticity during communications and while in storage. Type I devices are certified by the US National Security Agency (NSA) for classified information processing. Type 2 devices are certified by NSA for proprietary information processing. Type 3 devices are certified by NSA for general information processing. Type 4 devices are produced by industry or other nations without any formal certification.

4. Physical (environmental) security addresses the actual environment configuration, security procedures, countermeasures, and recovery strategies to protect the equipment and its location. These measures include separate processing facilities, restricted access into those facilities, and sweeps to detect eavesdropping devices.

5. Security architecture and design contains the concepts, processes, principles, and standards used to define, design, and implement secure applications, operating systems, networks, and equipment. The security architecture must integrate various levels of confidentiality, integrity, and availability to ensure effective operations and adherence to governance.

6. Business continuity and disaster recovery planning are the preparations and practices which ensure business survival given events, natural or man-made, which cause a major disruption in normal business operations. Processes and specific action plans must be selected to prudently protect business processes and to ensure timely restoration.

7. Telecommunications and network security are the transmission methods and security measures used to provide integrity, availability, and confidentiality of data during transfer over private and public communication networks.

8. Application development security involves the controls applied to application software in a centralized or distributed environment. Application software includes tools, operating systems, data warehouses, and knowledge systems.

9. Operations security is focused on providing system availability for end users while protecting data processing resources both in centralized data processing centers and in distributed client/server environments.

10. Legal, regulations, investigations, and compliance issues include the investigative measures to determine if an incident has occurred and the processes for responding to such incidents.

One response to the complexity and diversity of security needs and domains that contribute to system security is "defense in depth," a commonly applied architecture and design approach. Defense in depth implements multiple layers of defense and countermeasures, making maximum use of certified equipment in each layer to facilitate system accreditation.

Discipline Management

Information to be supplied at a later date.

Discipline Relationships

Interactions

Information to be supplied at a later date.

Dependencies

Web-based Resource

A good online resource for system and software assurance is the US Department of Homeland Security's Build Security In [1] web site (DHS 2010), which provides resources for best practices, knowledge, and tools for engineering secure systems.
**Discipline Standards**
Information to be supplied at a later date.

**Personnel Considerations**
Information to be supplied at a later date.

**Metrics**
Information to be supplied at a later date.

**Models**
Information to be supplied at a later date.

**Tools**
Information to be supplied at a later date.

**References**

**Works Cited**


Primary References


Additional References


Electromagnetic Interference/Electromagnetic Compatibility

Electromagnetic Interference (EMI) is the disruption of operation of an electronic device when it is in the vicinity of an electromagnetic field in the radio frequency (RF) spectrum. Many electronic devices fail to work properly in the presence of strong RF fields. The disturbance may interrupt, obstruct, or otherwise degrade or limit the effective performance of the circuit. The source may be any object, artificial or natural, that carries rapidly changing electrical currents.

Electromagnetic Compatibility (EMC) is the ability of systems, equipment, and devices that utilize the electromagnetic spectrum to operate in their intended operational environments without suffering unacceptable degradation or causing unintentional degradation because of electromagnetic radiation or response. It involves the application of sound electromagnetic spectrum management; system, equipment, and device design configuration that ensures interference-free operation; and clear concepts and doctrines that maximize operational effectiveness (DAU 2010, Chapter 7).

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

Spectrum

Each nation has the right of sovereignty over the use of its spectrum and must recognize that other nations reserve the same right. It is essential that regional and global forums exist for the discussion and resolution of spectrum development and infringement issues between bordering and proximal countries that might otherwise be difficult to resolve.

The oldest, largest, and unquestionably the most important such forum, with 193 member countries, is the International Telecommunications Union (ITU) agency of the United Nations, which manages spectrum at a global level. As stated in Chapter 3 of the NTIA Manual, “The International Telecommunication Union (ITU)...is responsible for international frequency allocations, worldwide telecommunications standards and telecommunication development activities” (NTIA 2011, 3-2). The broad functions of the ITU are the regulation, coordination and
development of international telecommunications.

The spectrum allocation process is conducted by many different international telecommunication geographical committees. Figure 1 shows the various international forums represented worldwide.

Assigning frequencies is very complicated, as shown in the radio spectrum allocation chart in Figure 2. Sometimes, commercial entities try to use frequencies that are actually assigned to US government agencies, such as the Department of Defense (DoD). One such incident occurred when an automatic garage door vendor installed doors on homes situated near a government installation. Random opening and closing of the doors created a problem for the vendor that could have been avoided.

Four ITU organizations affect spectrum management (Stine and Portigal 2004):

1. World Radio-communication Conference (WRC)
2. Radio Regulations Board (RRB)
3. Radio-communications Bureau (RB)
4. Radio-communication Study Groups (RSG)

The WRC meets every four years to review and modify current frequency allocations. The RB registers frequency assignments and maintains the master international register. The RRB approves the Rules of Procedures used by the BR to register frequency assignments and adjudicates interference conflicts among member nations. The SG analyzes spectrum usage in terrestrial and space applications and makes allocation recommendations to the WRC. Most member nations generally develop national frequency allocation policies that are consistent with the Radio Regulations (RR). These regulations have treaty status.
Dual Management of Spectrum in the US

Whereas most countries have a single government agency to perform the spectrum management function, the US has a dual management scheme intended to insure that

- decisions concerning commercial interests are made only after considering their impact on government systems; and
- government usage supports commercial interests.

The details of this scheme, established by the Communications Act of 1934, are as follows:

- the Federal Communications Commission (FCC) is responsible for all non-government usage
- the FCC is directly responsible to Congress;
- the president is responsible for federal government usage, and by executive order, delegates the federal government spectrum management to the National Telecommunications and Information Administration (NTIA); and
- the NTIA is under the authority of the Secretary of Commerce.

The FCC regulates all non-federal government telecommunications under Title 47 of the Code of Federal Regulations. For example, see FCC (2009, 11299-11318). The FCC is directed by five Commissioners appointed by the president and confirmed by the Senate for five-year terms. The Commission staff is organized by function. The responsibilities of the six operating Bureaus include processing applications for licenses, analyzing complaints, conducting investigations, implementing regulatory programs, and conducting hearings (http://www.fcc.gov).

The NTIA performs spectrum management function through the Office of Spectrum Management (OSM), governed by the Manual of Regulations and Procedures for Federal Radio Frequency Management. The IRAC develops and executes policies, procedures, and technical criteria pertinent to the allocation, management, and usage of spectrum. The Spectrum Planning and Policy Advisory Committee (SPAC) reviews the IRAC plans, balancing considerations of manufacturing, commerce, research, and academic interests.

Within the DoD, spectrum planning and routine operation activities are cooperatively managed. Spectrum certification is a mandated process designed to ensure that

1. frequency band usage and type of service in a given band are in conformance with the appropriate national and international tables of frequency allocations;
2. equipment conforms to all applicable standards, specifications, and regulations; and
3. approval is provided for expenditures to develop equipment dependent upon wireless communications.

Host Nation Coordination and Host Nation Approval

In peacetime, international spectrum governance requires military forces to obtain host nation permission — Host Nation Coordination (HNC)/Host Nation Approval (HNA) — to operate spectrum-dependent systems and equipment within a sovereign nation. For example, international governance is honored and enforced within the United States by the US departments of State, Defense, and the user service.

In wartime, international spectrum governance is not honored between warring countries; however, the sovereign spectrum rights of bordering countries must be respected by military forces executing their assigned missions. For example, HNA is solicited by US naval forces to use spectrum-dependent systems and equipment in bordering countries' airspace and/or on bordering countries' soil. HNA must be obtained before the operation of spectrum-dependent systems and equipment within a sovereign nation. The combatant commander is responsible for coordinating requests with sovereign nations within his or her area of responsibility. Because the combatant commander has no authority over a sovereign nation, the HNC/HNA process can be lengthy and needs to be started early in the development of a system. Figure 2 illustrates a spectrum example.
Practical Considerations

EMI/EMC is difficult to achieve for systems that operate world-wide because of the different frequencies in which products are designed to operate in each of the telecommunication areas. Billions of US dollars have been spent in retrofitting US DoD equipment to operate successfully in other countries.

It is important to note that the nuclear radiation environment is drastically more stressing than, and very different from, the space radiation environment.

System Description

Narrowband and Broadband Emissions

To help in analyzing conducted and radiated interference effects, EMI is categorized into two types—narrowband and broadband—which are defined as follows:

- **Narrowband Emissions**:
  
  A narrowband signal occupies a very small portion of the radio spectrum... Such signals are usually continuous sine waves (CW) and may be continuous or intermittent in occurrence... Spurious emissions, such as harmonic outputs of narrowband communication transmitters, power-line hum, local oscillators, signal generators, test equipment, and many other man made sources are narrowband emitters. (Bagad 2009, G-1)

- **Broadband Emissions**:
  
  A broadband signal may spread its energy across hundreds of megahertz or more... This type of signal is composed of narrow pulses having relatively short rise and fall times. Broadband signals are further
Electromagnetic Interference/Electromagnetic Compatibility

*divided into random and impulse sources. These may be transient, continuous or intermittent in occurrence. Examples include unintentional emissions from communication and radar transmitters, electric switch contacts, computers, thermostats, ignition systems, voltage regulators, pulse generators, and intermittent ground connections.* (Bagad 2009, G-1)

**TEMPEST**

TEMPEST is a codename used to refer to the field of emission security. The National Security Agency (NSA) investigations conducted to study compromising emission (CE) were codenamed TEMPEST. National Security Telecommunications Information Systems Security Issuance (NSTISSI)-7000 states:

*Electronic and electromechanical information-processing equipment can produce unintentional intelligence-bearing emanations, commonly known as TEMPEST. If intercepted and analyzed, these emanations may disclose information transmitted, received, handled, or otherwise processed by the equipment.* (NSTISS 1993, 3)

These compromising emanations consist of electrical, mechanical, or acoustical energy intentionally or unintentionally emitted by sources within equipment or systems which process national security information. Electronic communications equipment needs to be secured from potential eavesdroppers while allowing security agencies to intercept and interpret similar signals from other sources. The ranges at which these signals can be intercepted depends upon the functional design of the information processing equipment, its installation, and prevailing environmental conditions.

Electronic devices and systems can be designed, by means of Radiation Hardening techniques, to resist damage or malfunction caused by ionizing and other forms of radiation (Van Lint and Holmes Siedle 2000). Electronics in systems can be exposed to ionizing radiation in the Van Allen radiation belts around the Earth’s atmosphere, cosmic radiation in outer space, gamma or neutron radiation near nuclear reactors, and electromagnetic pulses (EMP) during nuclear events.

A single charged particle can affect thousands of electrons, causing electronic noise that subsequently produces inaccurate signals. These errors could affect safe and effective operation of satellites, spacecraft, and nuclear devices. Lattice displacement is permanent damage to the arrangement of atoms in element crystals within electronic devices. Lattice displacement is caused by neutrons, protons, alpha particles, and heavy ions. Ionization effects are temporary damages that create latch-up glitches in high power transistors and soft errors like bit flips in digital devices. Ionization effects are caused by charged particles.

Most radiation-hardened components are based on the functionality of their commercial equivalents. Design features and manufacturing variations are incorporated to reduce the components’ susceptibility to interference from radiation. Physical design techniques include insulating substrates, package shielding, chip shielding with depleted boron, and magneto-resistive RAM. Logical design techniques include error-correcting memory, error detection in processing paths, and redundant elements at both circuit and subsystem levels (Dawes 1991). Nuclear hardness is expressed as susceptibility or vulnerability for given environmental conditions. These environmental conditions include peak radiation levels, overpressure, dose rates, and total dosage.
**Discipline Management**
Information to be supplied at a later date.

**Discipline Relationships**

**Interactions**
Information to be supplied at a later date.

**Dependencies**
Information to be supplied at a later date.

**Discipline Standards**
Information to be supplied at a later date.

**Personnel Considerations**
Information to be supplied at a later date.

**Metrics**
Information to be supplied at a later date.

**Models**
Information to be supplied at a later date.

**Tools**
Information to be supplied at a later date.

**References**

**Works Cited**


Primary References

Additional References
None.

**SEBoK Discussion**
Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

*If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox[^1].*

[^1]: ENCODED_CONTENT
MTgwNjEPGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2Nya7END_ENCODED_CONTENT
Resilience Engineering

According to the Oxford English Dictionary on Historical Principles (1973), resilience (glossary) is "the act of rebounding or springing back." This definition most directly fits the situation of materials which return to their original shape after deformation. For human-made systems this definition can be extended to say "the ability of a system to recover from a disruption (glossary)." The US government definition for infrastructure (glossary) systems is the "ability of systems, infrastructures, government, business, communities, and individuals to resist, tolerate, absorb, recover from, prepare for, or adapt to an adverse occurrence that causes harm, destruction, or loss of national significance" (DHS 2010). The concept of creating a resilient human-made system or resilience engineering is discussed by Hollnagel, Woods, and Leveson (2006). The principles are elaborated by Jackson (2010).

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

Resilience is a relatively new term in the SE realm, appearing only in the 2006 timeframe and becoming popularized in the 2010 timeframe. The recent application of "resilience" to engineered systems has led to confusion over its meaning and a proliferation of alternative definitions. (One expert claims that well over 100 unique definitions of resilience have appeared.) The details will continue to be discussed and debated, but the information here should provide a working understanding of the meaning and implementation of resilience, sufficient for a system engineer to effectively address it.

Definition

It is difficult to identify a single definition that — word for word — satisfies all. However, it is possible to gain general agreement of what is meant by resilience of engineered systems; viz., Resilience is the ability to provide required capability in the face of adversity.

Scope of the means

In applying this definition, one needs to consider the range of means by which resilience is achieved: The means of achieving resilience include avoiding, withstanding, recovering from and evolving and adapting to adversity. These may also be considered the fundamental objectives of resilience. Classically, resilience includes "withstanding" and "recovering" from adversity. For the purpose of engineered systems, "avoiding" adversity is considered a legitimate means of achieving resilience. Jackson and Ferris (2016). Also, it is believed that resilience should consider the system’s ability to "evolve and adapt" to future threats and unknown-unknowns.

Scope of the adversity

Another consideration will be to identify the range of adversities that are to be considered by resilience. We propose that the SE must consider all; i.e., environmental, normal failure and opponent, friendly & neutral parties.

The purpose of resilience engineering and architecting is to achieve full or partial recovery of a system following an encounter with a threat (glossary) that disrupts the functionality of that system. Threats can be natural, such as earthquakes, hurricanes, tornadoes, or tsunamis. Threats can be internal and human-made such as reliability flaws and human error. Threats can be external and human-made, such as terrorist attacks. Often, a single incident is the result of multiple threats, such as a human error committed in the attempt to recover from another threat.

Figure 1 depicts the loss and recovery of the functionality of a system. System types include product systems of a technological nature and enterprise systems such as civil infrastructures. They can be either individual systems or systems of systems. A resilient system possesses four attributes — capacity (glossary), flexibility (glossary),
tolerance (glossary), and cohesion (glossary) — and thirteen top level design principles through which to achieve these attributes. The four attributes are adapted from Hollnagel, Woods, and Leveson (2006), and the design principles are extracted from Hollnagel et al. and are elaborated based on Jackson (2010).

The Capacity Attribute

Capacity is the attribute of a system that allows it to withstand a threat. Resilience allows that the capacity of a system may be exceeded, forcing the system to rely on the remaining attributes to achieve recovery. The following design principles apply to the capacity attribute:

- The absorption (glossary) design principle calls for the system to be designed including adequate margin to withstand a design-level threat.
- The physical redundancy (glossary) design principle states that the resilience of a system is enhanced when critical components are physically redundant.
- The functional redundancy design principle calls for critical functions to be duplicated using different means.
- The layered defense design principle states that single points of failure should be avoided.

The absorption design principle requires the implementation of traditional specialties, such as Reliability and Safety.
The Flexibility Attribute

Flexibility is the attribute of a system that allows it to restructure itself in the face of a threat. The following design principles apply to the flexibility attribute:

- The reorganization design principle says that the system should be able to change its own architecture (glossary) before, during, or after the encounter with a threat. This design principle is applicable particularly to human systems.
- The human backup design principle requires that humans be involved to back up automated systems especially when unprecedented threats are involved.
- The complexity (glossary) avoidance design principle calls for the minimization of complex elements, such as software and humans, except where they are essential (see human backup design principle).
- The drift correction (glossary) design principle states that detected threats or conditions should be corrected before the encounter with the threat. The condition can either be immediate as for example the approach of a threat, or they can be latent (glossary) within the design or the organization.

The Tolerance Attribute

Tolerance is the attribute of a system that allows it to degrade gracefully following an encounter with a threat. The following design principles apply to the tolerance attribute.

- The localized capacity (glossary) design principle states that, when possible, the functionality of a system should be concentrated in individual nodes of the system and stay independent of the other nodes.
- The loose coupling (glossary) design principle states that cascading failures in systems should be checked by inserting pauses between the nodes. According to Perrow (1999) humans at these nodes have been found to be the most effective.
- The neutral state (glossary) design principle states that systems should be brought into a neutral state before actions are taken.
- The reparability design principle states that systems should be reparable to bring the system back to full or partial functionality.

Most resilience design principles affect system design processes such as architecting. The reparability design principle affects the design of the sustainment system.

The Cohesion Attribute

Cohesion is the attribute of a system that allows it to operate before, during, and after an encounter with a threat. According to (Hitchins 2009), cohesion is a basic characteristic of a system. The following global design principle applies to the cohesion attribute.

- The inter-node interaction (glossary) design principle requires that nodes (glossary) (elements) of a system be capable of communicating, cooperating, and collaborating with each other. This design principle also calls for all nodes to understand the intent of all the other nodes as described by (Billings 1991).
The Resilience Process

Implementation of resilience in a system requires the execution of both analytic and holistic processes. In particular, the use of architecting with the associated heuristics is required. Inputs are the desired level of resilience and the characteristics of a threat or disruption. Outputs are the characteristics of the system, particularly the architectural characteristics and the nature of the elements (e.g., hardware, software, or humans).

Artifacts depend on the domain of the system. For technological systems, specification and architectural descriptions will result. For enterprise systems, enterprise plans will result.

Both analytic and holistic methods, including the principles of architecting, are required. Analytic methods determine required capacity. Holistic methods determine required flexibility, tolerance, and cohesion. The only aspect of resilience that is easily measurable is that of capacity. For the attributes of flexibility, tolerance, and cohesion, the measures are either Boolean (yes/no) or qualitative. Finally, as an overall measure of resilience, the four attributes (capacity, flexibility, tolerance, and cohesion) can be weighted to produce an overall resilience score.

The greatest pitfall is to ignore resilience and fall back on the assumption of protection. The Critical Thinking project (CIPP 2007) lays out the path from protection to resilience. Since resilience depends in large part on holistic analysis, it is a pitfall to resort to reductionist thinking and analysis. Another pitfall is failure to consider the systems of systems philosophy, especially in the analysis of infrastructure systems. Many examples show that systems are more resilient when they employ the cohesion attribute — the New York Power Restoration case study by Mendoca and Wallace (2006, 209-219) is one. The lesson is that every component system in a system of systems must recognize itself as such, and not as an independent system.

Practical Considerations

Resilience is difficult to achieve for infrastructure systems because the nodes (cities, counties, states, and private entities) are reluctant to cooperate with each other. Another barrier to resilience is cost. For example, achieving redundancy in dams and levees can be prohibitively expensive. Other aspects, such as communicating on common frequencies, can be low or moderate cost; even there, cultural barriers have to be overcome for implementation.

System Description

A system is "[a]n integrated set of elements, subsystems or assemblies that accomplish a defined objective." INCOSE (2015) A capability is "...an expression of a system ... to achieve a specific objective under stated conditions." INCOSE (2015)

Resilience is the ability of a system to provide required capability in the face of adversity. Resilience in the realm of systems engineering involves identifying: 1) the capabilities that are required of the system, 2) the adverse conditions under which the system is required to deliver those capabilities, and 3) the systems engineering to ensure that the system can provide the required capabilities.

Put simply, resilience is achieved by a systems engineering focusing on adverse conditions.

Principles for Achieving Resilience

34 principles and support principles described by Jackson and Ferris (2013) include both design and process principles that will be used to define a system of interest in an effort to make it resilient. These principles were extracted from many sources. Prominent among these sources is Hollnagel et al (2006). Other sources include Leveson (1995), Reason (1997), Perrow (1999), and Billings(1997). Some principles were implied in case study reports, such as the 9/11 Commission report (2004) and the US-Canada Task Force report (2004) following the 2003 blackout.

These principles include very simple and well-known principles as physical redundancy and more sophisticated principles as loose coupling. Some of these principles are domain dependent, such as loose coupling, which is
important in the power distribution domain as discussed by Perrow (1999). These principles will be the input to the state-transition analysis of Section 8 to determine the characteristics of a given system for a given threat.

In the resilience literature the term principle is used to describe both scientifically accepted principles and also heuristics, design rules determined from experience as described by Rechtin (1991). Jackson and Ferris (2013) showed that it is necessary to invoke these principles in combinations to enhance resilience. This concept is called defense in depth. Paré (2011) illustrates how defense in depth was used to achieve resilience in the 2009 ditching of US Airways Flight 1549.

Uday and Marais (2015) apply the above principles to the design of a system-of-systems. Henry and Ramirez-Marquez (2016) describe the state of the U.S. East Coast infrastructure in resilience terms following the impact of Hurricane Sandy in 2012. Bodeau & Graubert (2011) propose a framework for understanding and addressing cyber-resilience. They propose a taxonomy comprised of four goals, eight objectives, and fourteen cyber-resilience practices. Many of these goals, objectives and practices can be applied to non-cyber resilience.

**Discipline Management**

Most enterprises, both military and commercial, include organizations generally known as Advanced Design. These organizations are responsible for defining the architecture of a system at the very highest level of the system architecture. This architecture reflects the resilience principles described in Section 2 and the processes associated with that system. In many domains, such as fire protection, no such organization will exist. However, the system architecture will still need to be defined by the highest level of management in that organization. In addition, some aspects of resilience will be defined by government imposed requirements as described in Section 5.

**Discipline Relationships**

**Interactions**

**Resilience Discipline Outputs**

The primary outputs of the resilience discipline are a subset of the principles described by Jackson and Ferris (2013) which have been determined to be appropriate for a given system, threat, and desired state of resilience as determined by the state-transition analysis described below. The processes requiring these outputs are the system design and system architecture processes.

**Resilience Discipline Inputs**

Inputs to the state-transition analysis described in Section 8 include (1) type of system of interest, (2) nature of threats to the system (earthquakes, terrorist threats, human error, etc.).
Dependencies

Information to be supplied at a later date.

Discipline Standards

ASIS International

ASIS (2009) has published a standard pertaining to the resilience of organizational systems. Some of the principles described in this standard can also be found in the larger set of principles described by Jackson and Ferris (2013) for engineered systems in general containing hardware, software, and humans.

Personnel Considerations

None have been identified.

Metrics

Uday & Marais (2015) performed a survey of resilience metrics. Those identified include:

- Time duration of failure
- Time duration of recovery
- Ratio of performance recovery to performance loss
- A function of speed of recovery
- Performance before and after the disruption and recovery actions
- System importance measures

Jackson (2016) developed a metric to evaluate various systems in four domains: aviation, fire protection, rail, and power distribution, for the principles that were lacking in ten different case studies. The principles are from the set identified by Jackson and Ferris (2013) and are represented in the form of a histogram plotting principles against frequency of omission. The data in these gaps were taken from case studies in which the lack of principles was inferred from recommendations by domain experts in the various cases cited.

Britis (2016) surveyed and evaluated a number of potential resilience metrics and identified the following: [Note: This reference is going through approval for public release and should be referenceable by the end of July 2016.]

- Maximum outage period
- Maximum brownout period
- Maximum outage depth
- Expected value of capability: the probability-weighted average of capability delivered
- Threat resiliency (the time integrated ratio of the capability provided divided by the minimum needed capability)
- Expected availability of required capability (the likelihood that for a given adverse environment the required capability level will be available)
- Resilience levels (the ability to provide required capability in a hierarchy of increasingly difficult adversity)
- Cost to the opponent
- Cost-benefit to the opponent
- Resource resiliency (the degradation of capability that occurs as successive contributing assets are lost)

Britis found that multiple metrics may be required, depending on the situation. However, if one had to select a single most effective metric for reflecting the meaning of resilience, it would be the expected availability of the required capability. Expected availability of the required capability is the probability-weighted sum of the availability summed across the scenarios under consideration. In its most basic form, this metric can be represented mathematically as:
where,

\[ R = \text{Resilience of the required capability (Cr)}; \]
\[ n = \text{the number of exhaustive and mutually exclusive adversity scenarios within a context (n can equal 1)}; \]
\[ P_i = \text{the probability of adversity scenario I}; \]
\[ Cr(t)_i = \text{timewise availability of the required capability during scenario I; } \quad 0 \text{ if below the required level } \quad 1 \text{ if at } \]
\[ \text{or above the required value (Where circumstances dictate this may take on a more complex, non-binary function of } \]
\[ \text{time.}); \]
\[ T = \text{length of the time of interest}. \]

**Models**

The state-transition model described by Jackson et al (2015) describes a system in its various states before, during, and after an encounter with a threat. The model identifies seven different states as the system passes from a nominal operational state to minimally acceptable functional state. In addition, the model identifies 28 transition paths from state to state. To accomplish each transition the designer must invoke one or more of the 34 principles or support principles described by Jackson and Ferris (2013). The designs implied by these principles can then be entered into a simulation to determine the total effectiveness of each design.

**Tools**

No tools dedicated to resilience have been identified.

**References**

**Works Cited**


Primary References

Additional References

< Previous Article | Parent Article | Next Article >
SEBoK v. 1.8 released 27 March 2017

SEBoK Discussion
Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.
Manufacturability and Producibility

Manufacturability and producibility is an engineering specialty. The machines and processes used to build a system must be architected and designed. A systems engineering approach to manufacturing and production is necessary because manufacturing equipment and processes can sometimes cost more than the system being built (Maier and Rechtin 2002). Manufacturability and producibility can be a discriminator between competing system solution concepts and therefore must be considered early in the study period, as well as during the maturing of the final design solution.

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

The system being built might be intended to be one-of-a-kind, or to be reproduced multiple times. The manufacturing system differs for each of these situations and is tied to the type of system being built. For example, the manufacture of a single-board computer would be vastly different from the manufacture of an automobile. Production involves the repeated building of the designed system. Multiple production cycles require the consideration of production machine maintenance and downtime.

Manufacturing and production engineering involve similar systems engineering processes specifically tailored to the building of the system. Manufacturability and producibility are the key attributes of a system that determine the ease of manufacturing and production. While manufacturability is simply the ease of manufacture, producibility also encompasses other dimensions of the production task, including packaging and shipping. Both these attributes can be improved by incorporating proper design decisions that take into account the entire system life cycle (Blanchard and Fabrycky 2005).
System Description
Information to be supplied at a later date.

Discipline Management
Information to be supplied at a later date.

Discipline Relationships
Interactions
Information to be supplied at a later date.

Dependencies
Information to be supplied at a later date.

Discipline Standards
Information to be supplied at a later date.

Personnel Considerations
Information to be supplied at a later date.

Metrics
Information to be supplied at a later date.

Models
Information to be supplied at a later date.

Tools
Information to be supplied at a later date.

References

Works Cited
Primary References

None.

Additional References


Environmental engineering addresses four issues that arise in system design and operation. They include: (1) design for a given operating environment, (2) environmental impact, (3) green design, and (4) compliance with environment regulations.

Please note that not all of the generic below sections have mature content at this time. Anyone wishing to offer content suggestions should contact the SEBoK Editors in the usual ways.

Overview

A system is designed for a particular operating environment. Product systems, in particular, routinely consider conditions of temperature and humidity. Depending on the product, other environmental conditions may need to be considered, including UV exposure, radiation, magnetic forces, vibration, and others. The allowable range of these conditions must be specified in the requirements for the system.

Requirements

The general principles for writing requirements also apply to specifying the operating environment for a system and its elements. Requirements are often written to require compliance with a set of standards.

System Description

Information to be supplied at a later date.

Discipline Management

Many countries require assessment of environmental impact of large projects before regulatory approval is given. The assessment is documented in an environmental impact statement (EIS). In the United States, a complex project can require an EIS that greatly adds to the cost, schedule, and risk of the project.

Scope

In the U.S., the process in Figure 1 is followed. A proposal is prepared prior to a project being funded. The regulator examines the proposal. If it falls into an excluded category, no further action is taken. If not, an environmental assessment is made. If that assessment determines a finding of no significant impact (FONSI), no further action is taken. In all other cases, an environmental impact statement is required.

**Legal References**

Basic references in the U.S. include the National Environmental Policy Act of 1969 and its implementing regulations (NEPA 1969) and the European commission directive (EC 1985). State and local regulations can be extensive; Burby and Paterson (1993) discuss improving compliance.

**Cost and Schedule Implications**

Depending on the scale of the project, the preparation of an EIS can take years and cost millions. For example, the EIS for the Honolulu light rail project took four years and cost $156M (Hill 2011). While a project may proceed even if the EIS finds a negative impact, opponents to a project may use the EIS process to delay a project. A common tactic is to claim the EIS was not complete in that it omitted some environmental impacts. Eccleston (2000) provides a guide to planning for EIS.
**Energy Efficiency**

There is a large amount of literature that has been published about design for energy efficiency. Lovins (2010) offers ten design principles. He also provides case studies (Lovins et al. 2011). Intel (2011) provides guidance for improving the energy efficiency of its computer chips. A great deal of information is also available in regard to the efficient design of structures; DOE (2011) provides a good overview.

Increased energy efficiency can significantly reduce total life cycle cost for a system. For example, the Toyota Prius was found to have the lowest life cycle cost for 60,000 miles, three years despite having a higher initial purchase price (Brown 2011).

**Carbon Footprint**

Increased attention is being paid to the emission of carbon dioxide. BSI British Standards offers a specification for assessing life cycle greenhouse emissions for goods and services (BSI 2011).

**Sustainability**

Graedel and Allenby (2009), Maydl (2004), Stasinopoulos (2009), Meryman (2004), and Lockton and Harrison (2008) discuss design for sustainability. Sustainability is often discussed in the context of the UN report on Our Common Future (WCED 1987) and the Rio Declaration (UN 1992).

**Discipline Relationships**

An enterprise must attend to compliance with the various environmental regulations. Dechant et al. (1994) provide the example of a company in which 17% of every sales dollar goes toward compliance activities. They discuss gaining a competitive advantage through better compliance. Gupta (1995) studies how compliance can improve the operations function. Berry (1998) and Nash (2001) discuss methods for environmental management by the enterprise.

**Interactions**

Information to be supplied at a later date.

**Dependencies**


**Discipline Standards**

Depending on the product being developed, standards may exist for operating conditions. For example, ISO 9241-6 specifies the office environment for a video display terminal. Military equipment may be required to meet MILSTD 810G standard (DoD 2014) in the US, or DEF STAN 00-35 in the UK (MoD 2006).

The U.S. Federal Aviation Administration publishes a list of EIS best practices (FAA 2002).

The U.S. Environmental Protection Agency (EPA) defines Green Engineering (glossary) as: the design, commercialization, and use of processes and products, which are feasible and economical, while minimizing (1) generation of pollution at the source and (2) risk to human health and the environment (EPA 2011). Green engineering embraces the concept that decisions to protect human health and the environment can have the greatest impact and cost effectiveness when applied early to the design and development phase of a process or product.

The EPA (2011) offers the following principles of green engineering:
• Engineer processes and products holistically, use systems analysis, and integrate environmental impact assessment tools.
• Conserve and improve natural ecosystems while protecting human health and well-being.
• Use life-cycle thinking in all engineering activities.
• Ensure that all material and energy inputs and outputs are as inherently safe and benign as possible.
• Minimize depletion of natural resources.
• Strive to prevent waste.
• Develop and apply engineering solutions, while being cognizant of local geography, aspirations, and cultures.
• Create engineering solutions beyond current or dominant technologies; additionally, improve, innovate, and invent (technologies) to achieve sustainability.
• Actively engage communities and stakeholders in development of engineering solutions.

Personnel Considerations
Information to be supplied at a later date.

Metrics
Information to be supplied at a later date.

Models
Information to be supplied at a later date.

Tools
Information to be supplied at a later date.

References

Works Cited


**Primary References**


**Additional References**

None.

< Previous Article | Parent Article | Next Article (Part 7) >

SEBoK v. 1.8 released 27 March 2017

**SEBoK Discussion**

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

*If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].*
Part 7 is a collection of systems engineering (SE) implementation examples to illustrate the principles described in the Systems Engineering Body of Knowledge (SEBoK) Parts 1-6. These examples describe the application of SE practices, principles, and concepts in real settings.

The intent is to provide typical instances of the application of SE so readers can learn from these experiences. This can improve the practice of SE by illustrating to students, educators, and practitioners the benefits of effective practice, as well as the risks and liabilities of poor practice.

A matrix of implementation examples is used to map the implementation examples to topics in the SEBoK, primarily Part 3. To provide a broader set of domains, both formal case studies and shorter vignettes are used. For the case studies, an introduction and analysis of the case is given with references to the full external case study. For the vignettes, the implementation example is described directly. In the SE literature, a wide variety of examples and formats are considered "case studies." Here, the distinction between a case study and a vignette is that a vignette is a short wiki article written for the SEBoK and a case study exists externally in the SE literature.

An initial set of examples is included with the anticipation that more will be added over time to highlight the different aspects and applications of SE. In addition, new examples can be added to demonstrate the evolving state of practice, such as the application of model-based SE and the engineering of complex, adaptive systems.
Knowledge Areas in Part 7

Each part of the SEBoK is divided into knowledge areas (KAs), which are groupings of information with a related theme. Part 7 is organized into the following KAs:

- Matrix of Implementation Examples
- Case Studies
- Vignettes

References

Works Cited


Primary References

None.

Additional References

None.

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].
**Matrix of Implementation Examples**

The following matrix maps the SEBoK Systems Engineering Implementation Examples to topics in the Systems Engineering Body of Knowledge (SEBoK). It provides both a list of potential systems engineering implementation examples for topics of interest, and a list of relevant topics for each implementation example. Since the number of topics in the SEBoK is extensive, only a subset are included here for clarity. For additional information, see the implementation example of interest and the corresponding SEBoK topic.

**Organization and Mapping of Case Studies to the SEBoK**

The following short titles shown in Table 1 (Developed for BKCASE) are used for the Case Study implementation examples:

<table>
<thead>
<tr>
<th>Case Studies</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>BT</strong></td>
<td>Business Transformation</td>
</tr>
<tr>
<td><strong>ATC</strong></td>
<td>NextGen Air Traffic Control</td>
</tr>
<tr>
<td><strong>NASA</strong></td>
<td>NASA's Mission to Saturn</td>
</tr>
<tr>
<td><strong>HST</strong></td>
<td>Hubble Space Telescope</td>
</tr>
<tr>
<td><strong>GPS</strong></td>
<td>Global Positioning System</td>
</tr>
<tr>
<td><strong>GPS II</strong></td>
<td>Global Positioning System II</td>
</tr>
<tr>
<td><strong>Radiation</strong></td>
<td>Medical Radiation</td>
</tr>
<tr>
<td><strong>FBI VCF</strong></td>
<td>FBI Virtual Case File System</td>
</tr>
<tr>
<td><strong>MSTI</strong></td>
<td>Miniature Seeker Technology Integration</td>
</tr>
<tr>
<td><strong>Infusion Pump</strong></td>
<td>Next Generation Medical Infusion Pump</td>
</tr>
<tr>
<td><strong>DFM</strong></td>
<td>Design for Maintainability</td>
</tr>
<tr>
<td><strong>CAS</strong></td>
<td>Complex Adaptive Operating System</td>
</tr>
<tr>
<td><strong>PM</strong></td>
<td>Project Management</td>
</tr>
<tr>
<td><strong>TS</strong></td>
<td>Taxi Service</td>
</tr>
<tr>
<td><strong>TS</strong></td>
<td>Taxi Service</td>
</tr>
<tr>
<td><strong>SWFTS</strong></td>
<td>SWFTS MBSE</td>
</tr>
<tr>
<td><strong>NHS</strong></td>
<td>Northwest Hydro System</td>
</tr>
</tbody>
</table>

Table 2 shows how the topics (each row) align with the Case Study implementation examples (each column):
### Table 2. Implementation Examples. Coverage of SEBoK Topics for Each Case Study (SEBoK Original)

<table>
<thead>
<tr>
<th>SEBoK Topic</th>
<th>BT</th>
<th>ATC</th>
<th>NASA</th>
<th>HST</th>
<th>GPS</th>
<th>Radiation</th>
<th>FBI</th>
<th>VCF</th>
<th>MSTI</th>
<th>Infusion Pump</th>
<th>DM</th>
<th>CAS</th>
<th>PM</th>
<th>TS</th>
<th>SWFTS</th>
<th>NHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Systems Thinking</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Models and Simulation</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product Systems Engineering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Service Systems Engineering</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Enterprise Systems Engineering</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Systems of Systems (SoS)</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Life Cycle Models</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Business or Mission Analysis</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stakeholder Needs and Requirements</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Requirements</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Architecture</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Analysis</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>System Implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Integration</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Verification</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Validation</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>System Deployment</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operation of the System</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>System Maintenance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistics</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Planning</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assessment and Control</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Risk Management</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Measurement</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Decision Management</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Configuration Management</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information Management</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>
Table 3. Short Titles for the Vignettes. (SEBoK Original)

<table>
<thead>
<tr>
<th>Case Studies</th>
<th>Bag Handling</th>
<th>VA Sub</th>
<th>Route Mod</th>
<th>Water Mgmt</th>
<th>FAA AAS</th>
<th>Light Rail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business or Mission Analysis</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Stakeholder Needs and Requirements</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Architecture</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Analysis</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Implementation</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Integration</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Verification</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Validation</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Deployment</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Operation of the System</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Maintenance</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Logistics</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Planning</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Assessment and Control</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Risk Management</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Measurement</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Decision Management</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Configuration Management</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Matrix of Implementation Examples

<table>
<thead>
<tr>
<th>Information Management</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quality Management</td>
<td>X</td>
</tr>
</tbody>
</table>

References

Works Cited
None.

Primary References
None.

Additional References
None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.8 released 27 March 2017

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].
Case Studies

Systems engineering principles described in the Systems Engineering Body of Knowledge (SEBoK) Parts 1-6 are illustrated in Part 7, Systems Engineering Implementation Examples. These examples describe the application of systems engineering practices, principles, and concepts in real settings. These systems engineering examples can be used to improve the practice of systems engineering by illustrating to students and practitioners the benefits of effective practice and the risks of poor practice. The SEBoK systems engineering implementation examples are grouped in two categories, Case Studies (glossary) and Vignettes (glossary). Case studies reference cases that have already been published by external sources in the existing literature. Vignettes are short wiki articles written specifically for the SEBoK.

List of Case Studies

The following case studies are included:

- Successful Business Transformation within a Russian Information Technology Company
- Federal Aviation Administration Next Generation Air Transportation System
- How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn
- Hubble Space Telescope Case Study
- Global Positioning System Case Study
- Global Positioning System Case Study II
- Medical Radiation Case Study
- FBI Virtual Case File System Case Study
- MSTI Case Study
- Next Generation Medical Infusion Pump Case Study
- Design for Maintainability
- Complex Adaptive Operating System Case Study
- Complex Adaptive Project Management System Case Study
- Complex Adaptive Taxi Service Scheduler Case Study
- Submarine Warfare Federated Tactical Systems Case Study
- Northwest Hydro System

Systems engineering (SE) case studies can be characterized in terms of at least two relevant parameters, viz., their degrees of complexity and engineering difficulty, for example. Although a so-called quad chart is likely an oversimplification, a 2 x 2 array can be used to make a first-order characterization, as shown in Figure 1.
The x-axis depicts complicated, the simplest form of complexity, at the low-end on the left, and complex, representing the range of all higher forms of complexity on the right. The y-axis suggests how difficult it might be to engineer (or re-engineer) the system to be improved, using Conventional (classical or traditional) SE, at the low-end on the bottom, and Complex SE, representing all more sophisticated forms SE, on the top. This upper range is intended to cover system of systems (SoS) engineering (SoSE), enterprise systems engineering (ESE), as well as Complex SE (CSE). The distinctions among these various forms of SE may be explored by visiting other sections of the SEBoK. In summary, the SEBoK case study editors have placed each case study in one of these four quadrants to provide readers with a suggested characterization of their case study’s complexity and difficulty. For sake of compactness the following abbreviations have been used:

- Business Transformation (Successful Business Transformation within a Russian Information Technology Company)
- NextGen ATC (Federal Aviation Administration Next Generation Air Transportation System)
- Saturn Mission (How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn)
- Hubble (Hubble Space Telescope Case Study)
- GPS and GPS II (Global Positioning System Case Study)
- Medical Radiator (Medical Radiation Case Study)
- FBI Case Files (FBI Virtual Case File System Case Study)
- Small Satellite MSTI (MSTI Case Study)
- Medical Infusion Pump (Next Generation Medical Infusion Pump Case Study)
- Incubator Maintainability Design (Design for Maintainability)
- Complex Adaptive Operations (Complex Adaptive Operating System)
- Taxi Scheduler (The Development of the First Real-Time Complex Adaptive Scheduler for a London Taxi Service)
- Project Management (The Development of a Real-Time Complex Adaptive Project Management Systems)
- SWFTS MBSE (Submarine Warfare Federated Tactical Systems Case Study)

**Value of Case Studies**

Case studies have been used for decades in medicine, law, and business to help students learn fundamentals and to help practitioners improve their practice. A Matrix of Implementation Examples is used to show the alignment of systems engineering case studies to specific areas of the SEBoK. This matrix is intended to provide linkages between each implementation example to the discussion of the systems engineering principles illustrated. The selection of case studies cover a variety of sources, domains, and geographic locations. Both effective and ineffective use of systems engineering principles are illustrated.

The number of publicly available systems engineering case studies is growing. Case studies that highlight the aerospace domain are more prevalent, but there is a growing number of examples beyond this domain.

The United States Air Force Center for Systems Engineering (AF CSE) has developed a set of case studies "to facilitate learning by emphasizing the long-term consequences of the systems engineering/programmatic decisions on cost, schedule, and operational effectiveness." (USAF Center for Systems Engineering 2011) The AF CSE is using these cases to enhance SE curriculum. The cases are structured using the Friedman-Sage framework (Friedman and Sage 2003; Friedman and Sage 2004, 84-96), which decomposes a case into contractor, government, and shared responsibilities in the following nine concept areas:

1. Requirements Definition and Management
2. Systems Architecture Development
3. System/Subsystem Design
4. Verification/Validation
This framework forms the basis of the case study analysis carried out by the AF CSE. Two of these case studies are highlighted in this SEBoK section, the Hubble Space Telescope Case Study and the Global Positioning System Case Study.

The United States National Aeronautics and Space Administration (NASA) has a catalog of more than fifty NASA-related case studies (NASA 2011). These case studies include insights about both program management and systems engineering. Varying in the level of detail, topics addressed, and source organization, these case studies are used to enhance learning at workshops, training, retreats, and conferences. The use of case studies is viewed as important by NASA since "organizational learning takes place when knowledge is shared in usable ways among organizational members. Knowledge is most usable when it is contextual" (NASA 2011). Case study teaching is a method for sharing contextual knowledge to enable reapplication of lessons learned. The MSTI Case Study is from this catalog.

References

Works Cited


Primary References


Successful Business Transformation within a Russian Information Technology Company

This article describes a successful business transformation of an information technology enterprise. The topic may be of particular interest, especially because this transformation was accomplished by a Russian company during the republic's fast growing economic recovery.

For additional information, refer to the closely related topics of Enabling Businesses and Enterprises and Enterprise Systems Engineering.

Background

In 2001, the top management of the IBS company [1] in Moscow initiated a fundamental transformation to change the company’s strategy and business model. The company was one of the biggest Russian information technology (IT) systems integrators at that time, with about 900 employees. Annual revenues of about $80M were mainly generated by information technology (IT) infrastructure projects (complex computing systems, multi-service networks, etc.) and hardware and software distribution. The transformation of the company to form new capabilities in IT services and the associated consulting area is the main topic in the case study.

During the transformation period (from 2001 to the present) IBS was represented as a set of autonomous business units (BUs), called constituent systems, which are virtual, independent businesses with the following characteristics.

- Profit and loss reporting was required for each BU according to management accounting procedures
- BU management established and independently conducted human resources, technology, and product policy
- A centralized back-office was organized to provide supporting functions for each BU. Thus, BUs do not have back-offices; they rely on and “pay” a corporate governing center (CGC) for these services.
A thorough Enterprise System (glossary) (ES) transformation was executed as a set of activities: mission analysis and capabilities decomposition, business architecting, planning of the project program, and implementation of the new business model.

Before and after transformation IBS was an exemplar directed System of Systems (SoS) (glossary): the constituent BUs are autonomous but their operations are supervised by CGC. At the same time IBS also has significant features of an acknowledged SoS: the constituent BUs retain their independent development and sustainment approaches, and changes in the company are based on collaboration between the CGC and each constituent; even operations of BUs are not controlled but only supervised/governed by the CGC through "soft" recommendations and coordination.

IBS was a quite mature ES before the transformation, and it was thoroughly upgraded to form new capabilities of the whole system as well as of the constituents.

**Purpose**

In 2000-2001 IBS management forecasted considerable growth of the Russian IT services and consulting market based on the fast growing Russian economy, which was rapidly recovering from the national financial crisis of 1998. The largest corporations started overseas expansion and borrowed from international markets to finance this growth. IBS predicted corresponding growth in the complexity of business processes and their associated software and hardware systems all of which should require more consulting and IT services.

Based on this forecast, management established a strategy goal to double the share of IT services and consulting from 25% to 50% over one year; further growth in this business was planned as a long term trend.

The consulting and IT services business is very complex technologically and organizationally and dramatically differs from IBS’s former infrastructure focus. Thus, a fundamental transformation was required, and it was executed during 2002.

Initially detected problems appeared as expenditures exceeding resources, slow delivery of the projects and reworking. Later, as it was expected, new problems appeared, for example, disinterest of BUs’ managers in developing new technologies or raising qualified employees’ motivation. All those problems were solved during transformation and during further development.

The first step of the transformation included strategic analysis and mission-to-capabilities decomposition. Five major capability groups to be focused on were defined. The groups and exemplar capabilities for each group are represented at Figure 1.
Challenges

All main challenges were caused by knowledge/information deficit described by three factors listed as a, b, and c below.

a. The lack of experience in enterprise transformation (and capability based approaches, even the lack of any textbooks or guides in those areas) was the major challenge which IBS management faced. The task to be solved did not devolve to organizational changes (which was a well-developed and described area), but was appropriately allocated to Enterprise System (glossary) or system of systems (SoS) engineering. In spite of the lack of experience it was decided to prepare and execute the transformation based on the company's employees without involving external consultants. The following arguments supported the decision.

• The task to be solved was not typical, so there weren't widely used and well tested algorithms or methods, and there weren't a lot of consultants experienced in exactly what was needed. So only consultants with general experience (strategy consulting, organizational management) might be hired.
• The Russian consulting industry in 2001-2002 was not well developed, so only foreign professionals were available. But foreign consultants would have needed to study Russian specifics; such study would have unduly lengthened the duration and increased the cost of the transformation.
• A joint transformation team would have to be formed, and IBS employees would have to be involved: management would have to be interviewed and be involved in decision making. In any case all employees would have to participate in change implementations.
• External consultants are not stakeholders; so their level of interest in helping to achieve success might not be very high, and their output also might not be outstanding.
• Unwillingness to open professional secrets and other intellectual property issues to direct competitors were other factors that prevented hiring of external consultants.

Thus, the final decision was to execute the transformation without involvement of external consulting resources. A special BoU responsible for business processes development was established and an agile (glossary) program management approach was applied to handle challenges and to pursue opportunities as well as to mitigate risks.
b. A very high complexity IBS as an enterprise system or SoS. Management recognized that the company and its environment was very complex, with a lot different agents, many constituents, and countless relationships; and that an enterprise system or SoS might become even more complex after transformation. This complexification happened as the company became an “extended enterprise”, the governing hierarchies weakened, and the demand for more sophisticated relationships increased.

c. The risk of mistaken forecast of IT market development. The expected growth of the consulting and services market might have not happened. In this case the transformation would have been senseless. This challenge generated additional emotional stress for management.

**Systems Engineering Practices**

The SE task of the transformation was established in the form: to develop required capabilities for an enterprise system or SoS – IBS company. The SE process might be represented by the following specific IBS interpretation of the Vee (V) Model (glossary) (“V model”) with Stages 1 through 7 (Figure 2).

Initially (Stage 1) the mission was translated to capabilities (Figure 1); "understanding the constituent systems (BUs) and their relationships" was executed. The transformation team found that capabilities might not be directly translated to any business-agent. Neither BUs (they serve as resource pools), nor projects (being temporal elements), nor employees (each of them have a finite set of skills, experience, responsibilities, etc.) might realize necessary capabilities.

Realizing this (Stage 2) transformation team defined several key areas (Figure 2) of company’s operations or activities which were supposed to be changed to form new capabilities. Appropriate artifacts (procedures, guides, documents, software systems) to support new capabilities were developed and implemented for each of the areas; these new assets formed exactly the corporate infrastructure of new business model.
For each new and legacy system (Stage 3) a set of conceptual design documents was developed, describing approaches, polices, processes, and procedures. The entire set of documents formed the business architecture description of the company. The description connected all key areas and defined a target operation model of the company after transformation. This architecture represented multiple views of the IBS company, and thus aptly reflected its enterprise system or SoS nature.

Somewhat in contrast with the conventional linear systems engineering approach advocated by the V model, Stages 4-6 were conducted in parallel to save time and resources. The company’s performance (Stage 7) should be monitored based on indicators’ measurements, and improvements should be developed and implemented (arrows from Stage 3 to Stage 7). Such iterations have been executed in practice not only during transformation but also later, when procedures, guides and the whole systems were updated.

Integration and interoperability of the new systems required a thorough integration of parallel development jobs. So joint workgroups were formed of the employees at the level of low officers; and CGC played the role of integrated workgroup at the management level. Actually, multi-level integrated workgroups were formed.

The major complexity and risks derived from the challenges described above.

The transformation team developed and used an approach which is very similar to the agile development approach to address those risks. The following principles were used to manage the portfolio of projects in case of uncertainty and deficit of knowledge.

- Form solutions as fast as possible (but not necessarily with pure quality) to test them in practice faster.
- Recognizing failures are unavoidable, perceive them readily and react rationally.
- In case of failure analyze the situation and find a new solution, generate changes, and update the plan.
- Work in parallel, verifying and coordinating intermediate results.
• The schedule might be corrected and updated but should not be jeopardized by improper execution.
• Formulate and test the most critical and most questionable solutions at first.
• Start from pilot area and then expand to embrace the entire scope.
• Use high quality monitoring and a “manual control mode” for piloting and testing developing solutions but not additional aspects to limit waste of the resources.

Following those principles including a very strong discipline of execution, a high level of the sponsorship and all-employee involvement enabled the transformation to be completed on time without hiring consultants while keeping and developing on-going business.

Lessons learned

IBS’s accomplishment of the mission was the major result of ES transformation. Shareholders and management recognized that new capabilities had been formed, that the company could deliver consulting and services, sell and execute complex projects, manage consulting resources effectively, measure its performance, and plan and forecast financial results. Created capabilities are emergent in some sense because they are not directly related to concrete constituents (BUs, or employee, or projects) but are realized by means of integrated end-to-end processes and functions, which are executed in the projects by employees.

The systems organization did not dramatically change during transformation; “visible structure” was not practically changed: no new types of business-agents appeared, existing types did not change much. Those factors did not create new capabilities. Target capabilities were formed as the result of development and implementation of, it would seem, auxiliary and supporting tools – new capabilities support systems. New capabilities were formed mainly by the changes in the intangible areas of governing media, corporate culture, relations, and personnel competences; as well as by the creation of new capabilities support systems; without considerable changes in main company’s business-agents. (Refer to Figure 3.)

The main challenges which management faced (the lack of experience and the ambiguity of market growth forecast) made the uncertainty factor the critical one in the transformation.

What Worked and Why?

An agile program management in general demonstrated its efficiency and applicability to "soft and uncertain" tasks, especially in triggering a pre-established process for dealing with unexpected events; the main aspects of the approach are:
• Senior and credible sponsors
• Multi-level integrated project team(s)
• Open information exchange
• Partnership and collaboration
• Proactive and motivated parties and constituents
• Creative and innovative way of development
What Did Not Work and Why?

Perhaps corporate knowledge base development was the only more or less serious task which was not solved in transformation. The company's management understood the usefulness of knowledge accumulation and further alienation from the carriers in utilizing their business knowledge, so the goal of developing their own knowledge base was established. Special database and software system were developed with appropriate guides, reports and data collection forms; but formal regulation to fill in engineering knowledge accumulation templates did not work. However later this issue progressed quite naturally and simply: common folders were established to store project data in free formats. Such folders served to accumulate knowledge but in flat, unstructured form.
Best Practices and Replication Prospects

The following methods and approaches were proven as efficient and convenient in transformation.

1. Capability based development approach and capability based architecting might be recommended to be utilized in creation and transformation of an enterprise system or SoS. These focused all efforts on the required capabilities and involved very important relations from mission to capabilities and to functions in systems engineering process.

2. An agile program management might be used to solve a wide range of fuzzy and ambiguous problems of different scale in the areas of SE, ES engineering, SoS engineering where there is much uncertainty and lacks of expertise and proven methods or algorithms to solve them. The combination of “soft” and very creative designs with strong planning and progress control is the crucial foundation of this approach.

3. Key area definition and development appropriate to generating new capabilities for support systems (core consulting and services technologies, project implementation systems, systems for business unit growth, management accounting systems, motivation systems). Precisely defining these areas and developing integrated systems in these areas might be considered as quite common for application to a broader group of ESs.

References

Works Cited


Primary References


Additional References

None
References


Federal Aviation Administration Next Generation
Air Transportation System

This article describes a massive undertaking to modernize the air traffic management enterprise (glossary). The topic may be of particular interest to those involved in air transportation whether in connection with their careers or as pilots or passengers on airplanes. For additional information, refer to the closely related topics of Enabling Businesses and Enterprises and Enterprise Systems Engineering.

Background

This case study presents the Systems Engineering (glossary) and Enterprise Systems Engineering (ESE) (glossary) efforts in the Next Generation (NextGen) Air Transportation Systems by the Federal Aviation Administration (FAA 2008). NextGen is an unprecedented effort by multiple U.S. federal organizations to transform the U.S. air transportation infrastructure (glossary) from a fragmented ground-based navigation system to a net-centric satellite-based navigation system. This project is unique to the FAA because of its large scale, the huge number of stakeholder(s) involved, the properties of the system of interest, and the revolutionary changes required in the U.S. Air Transportation Network (U.S. ATN) enterprise.

A Sociotechnical System (glossary) like the U.S. ATN is a "large-scale [system] in which humans and technical constituents are interacting, adapting, learning, and coevolving. In [such] systems technical constraints and social and behavioral complexity are of essential essence". (Darabi and Mansouri 2014). Therefore, in order to understand changes in the U.S. ATN it was seen as necessary to view it through a lens of evolutionary adaptation rather than rigid systems design. The U.S. ATN serves both military and commercial aircraft with its 19,782 airports, including 547 are commercial airports. Nineteen major airlines, with more than a billion dollars in annual total revenue, along with other 57 national and regional airlines, transport 793 million passengers and realize 53 billion revenue ton-miles.

The Air Traffic Organization (ATO) is responsible for ensuring aircraft navigation in the U.S. National Air Space (NAS) system using a five-layer architecture (glossary). Each aircraft goes through different layers and possibly various zones of this architecture as it takes off from an airport until its lands at another airport (Donohue and Zellweger 2001). However, this architecture is fragmented and many issues are raised: an airplane’s path through its route is not optimized, and the path may change its direction from one zone to another, the destination airport’s capacity is limited by the current regulations of minimum aircraft separation distance due to navigation limitations, the real-time weather information is not integrated into the system, and communications are mainly voice-based, etc.

In NextGen major changes to the U.S. ATN design are planned. As already stated, the navigation system will be changed from ground-based communication to satellite-based navigation. The current fragmented architecture will be integrated into a seamless net-centric information system in which the digital communication will replace the current voice communications. Moreover, weather information will be assimilated into decision making and planning across the system.
Purpose

The FAA’s purpose is “to provide the safest, most efficient aerospace system in the world”. Toward this end the NextGen project is aimed at enhancing the U.S.’s leadership position in air transportation.

During the last three decades the demand for air transportation shows exponential growth. In just one decade from 1995 to 2005 this demand showed a 44% percent increase. Therefore, the change in infrastructure was inevitable. Moreover, 9/11 attacks on the U.S. ATN emphasized this need for change. The combination of a requirement for a safer and more secure network and increasing demand was the motivation for President Bush to enact the Vision 100-Century of Aviation Reauthorization Act on 2003. A major part of this Act was to revolutionize the U.S. ATN by means of the NextGen project. The first integration plan of the project was released in 2004, and the project is estimated to continue until 2025.

The demand behavior of the U.S. ATN shows diverse degrees of congestion among airports. Although there are multitudes of airports in the system, the top 35 most congested airports carried more than 60% of the total traffic consistently during the period of 2000 to 2008. Because the growth of the network demand is not proportional, the demand in congested airports will be even higher.

A study by the Joint Planning and Development Office (JPDO) shows that flight delays in the current network will cause $6.5 billion of economic loss until 2015, and $19.6 billion until 2025. By implementing NextGen the delays are estimated to be reduced by 38% until 2020. Moreover, aircraft CO2 emissions are a major part of environmental pollution in crowded cities; these will be reduced by 14 million metric tons by 2020. The current level of jet fuel usage is also a known problem because of increasing fuel prices. The NextGen project will improve fuel usage by 1.4 billion gallons cumulative through 2020.

NextGen is pursuing multiple goals to retain the U.S. leadership in aviation, to expand the U.S. ATN capacity, to continue to ensure safety, to increase environment protection, to help ensure national air defense, all generally helping to increase the nation’s security (JPDO 2007a).

Eight general capabilities are defined in conducting this mission: (1) network-enabled information access, (2) performance-based operations and services, (3) weather assimilated into decision making, (4) layered adaptive security, (5) positioning, navigation, and timing (PNT) services, (6) aircraft trajectory-based operations (TBO), (7) equivalent visual operations (EVO), and (8) super-density arrival/departure operations.

To create the desired capabilities, general areas of transformations are defined as air traffic management operations, airport operations and infrastructure services, net-centric infrastructure services, shared situational awareness services, layered and adaptive security services, environmental management services, safety management services, and performance management services. The detailed changes in each area are discussed in Concept of Operations for NextGen (JPDO 2007a).

Challenges

An instructive part of this case study is observing evolution in understanding challenges from initial steps of the project through current efforts for delivering it. As an overall conclusion, the perspective on challenges shifted from technical problems and intra-organizational issues to more enterprise-wide issues.

The NextGen Implementation Plan 2008 discussed the following challenges (FAA 2008):

- performance analysis, to understand and assess operational capabilities
- policy, to balance responsibility between humans and automation, for environmental management processes, and for global harmonization strategies
- acquisition workforce staffing
- environmental planning, to resolve conflicts with local environmental constraints
- security
• transition from current ground-based navigation to automatic dependent surveillance – broadcast (ADS-B) technology.

A more recent report on Targeted NextGen Capabilities for 2025 (JPDO 2011) highlights the effect of the multi-stakeholder nature of the project on raising additional challenges. Achieving Interagency Collaboration is the first issue, which is important in implementing security, safety, policy making, and technological advancement.

Increasing capacity, reducing delay and protecting the environment are the main three promises of the NextGen project. However, reaching the defined high standards is not an easy task. A major part of this challenge is integrating new technologies into legacy systems, aircraft, airports, facilities, and organizations. Airlines and general aviation pilots resist the expense of additional avionics and communications equipment, even though it bolsters the common good of air travel.

Maintaining airports and airspace security requires coherent and harmonious work of multiple U.S. agencies. The core of this challenge is not just changing the technology but also the processes, organizational structures, and enterprises to meet the new requirements of security.

Moreover, the need for greater information sharing in this net-centric environment is a challenge. The current culture of limited information sharing in which inter-organizational and intra-organizational information is strictly divided creates tension in a seamless information sharing infrastructure. In addition to that, the responsibility of generating, sharing, and utilizing useful information should be addressed in advance to avoid costly mistakes.

Verification and validation of NextGen deliverables is a major issue. The traditional systems engineering methods of verification and validation are tailored for testing an isolated system, while by definition a project like NextGen requires new methodologies of verification and validation beyond the scope of one system. The knowledge and experience of advancement in systems engineering in this area can be of priceless value for future projects.

Balance between human decision-making and automation is required to ensure a correct policy for increasing traffic and safety concerns. Changes in both human resource and technological facilities are required to effectively address this challenge.

The support of local communities is essential to facilitate development of the U.S. ATN and its physical infrastructure.

Communication, navigation, and surveillance systems in NextGen are going through major changes in terms of capacity and technology. However, planning required backups for them in case of any emergency is an area of challenge in developing NextGen.

The rise of Unmanned Aircraft Systems (UASs) provides significant opportunities for both military and commercial applications. However, integrating them into the NAS and developing policing and strategies for safe and secure use is a concern for the revolutionized U.S. ATN.

And finally realizing the benefits of NextGen is dependent on the critical mass of early adopters, similar to any technological advancement. Therefore, the NextGen project authority requires well-defined policies for motivating stakeholders' participation.

**Systems Engineering Practices**

The FAA NextGen is not just a revolution of the U.S. air transportation infrastructure, but also a shift in its enterprise. The enterprise architecture document, which is developed by JPDO, provides an overview of the desired capabilities (JPDO 2007b).

The Enterprise Architecture (glossary) is described using Department of Defense Architecture Framework (DoDAF) and the Federal Enterprise Architecture (FEA). DoDAF is used to describe the operational aspects of the project. The three views of DoDAF, the Operational View (OV), the Systems View (SV), and the Technical Standards View (TV), are presented in the enterprise architecture document. The Overview and Summary Information (AV-1) is the formal statement about how to use the architecture, the Integrated Dictionary (AV-2) defines the terms in the
document, the Community Model (OV-1) presents a high level depiction of the NextGen community, the Operational Node Connectivity Description (OV-2) presents the information flow among operational nodes in the system, Operational Information Exchange Matrix (OV-3) details the description of information flow in OV-2. Other architectural views of the system based on DoDAF are the Activity Model (OV-5) which documents activities (functions and processes), the Operational Event/Trace Description (OV-6c) is a part of sequence and timing description of activities, the System Functionality Description (SV-4) explains system functional hierarchies, and the Operational Activity to System Functionality Traceability Matrix (SV-5) is specification of relationships between operational activities in architecture and functional activities. However, a challenging part of applying this Enterprise Architecture is transformation from legacy systems to the new NextGen. This transformation is the ultimate test for relevance and comprehensiveness of the developed Enterprise Architecture.

Acquisition is the heart of systems engineering activities in the FAA NextGen project. As mentioned in Challenges above, the current practice of verification of validation in systems engineering (SE) is geared toward single isolated systems, rather than a myriad of interconnected System of Systems (SoS) (glossary). Moreover, the capabilities of NextGen are interdependent, and different programs rely on each other to deliver the promises. 250 unique and highly interconnected acquisition programs are identified in the FAA’s Capital Investment Plan, and these are to be delivered by 1820 FAA acquisition professionals. In addition, program complexity, budget uncertainty, and the challenge of finding acquisition professionals present other problems. The experience of systems acquisition in NextGen can provide a useful knowledge for future similar projects.

Lessons Learned

Although major portions of the FAA NextGen project are technical transformations and physical infrastructure developments, the transformation in the aviation enterprise is important but to some degree neglected. Part of the issue might be the fact that this transformation is beyond the responsibility and capability of FAA. However, to accomplish NextGen’s perceived benefits it is important to realize the effects of legacy systems, and most importantly the legacy enterprise architecture of the U.S. ATN. Many of the actual challenges in the system arose because of this inattention.

The sequestration in the U.S. government, the Budget Control Act of 2011, has cut the project funding substantially in recent years. As a result the project schedule and portfolio are subject to constant and wide-spread changes. The FAA was focused on delivering Optimization of Airspace and Procedures in the Metroplex (OPAM) program which is designed to reduce the delay, fuel consumption, and exhaust emission in busiest airports. The three areas of Houston, North Texas, and Washington D.C. were planned to complete the design phase on 2013 and start implementation.

Out of 700 planned ADS-B ground stations, 445 were operational on February 2013. ADS-B capability is a NextGen descendant of current radar systems and provides situational awareness for the players in the NAS using the Global Positioning System (GPS) and Wide Area Augmentation System (WAAS).

On the enterprise part of the project, the FAA Modernization and Reform Act of 2012 provided financial incentives for airlines and commercial aviation manufacturers to implement the required equipment in their aircraft. These incentives are designed to engage the air transportation community in the project and to create the critical mass of equipped airplanes.

There are considerable practices in applying NextGen. Establishment of the JPDO made the efforts of the project more coherent and integrated. JPDO’s main responsibility is to coordinate development of NextGen. The role of this organization is to represent multiple stakeholders of the project, which enables it to resolve possible conflicts of interests inside one entity. Moreover, such an organization provides a venue for technical knowledge-sharing, creating a consensus, and making an integrated system.

Emphasizing delivery of the mid-term objectives of NextGen is another lesson of the project. It was a well-known practice documented by Forman and Maier to establish mid-points for complex projects (Forman 2000). Developing
a mid-level system provides the system designers an opportunity to examine their underlying assumptions, to identify best practices and heuristics in the context of the project, and to reapply the acquired knowledge thorough evolutionary developments. A major shift in the policy of FAA in recent years was to focus on delivering project mid-term objectives.

There are unique characteristics of NextGen which makes it a valuable case for learning and replicating to other complex transformation projects of sociotechnical systems. The scale of the project for infrastructure transformation is unprecedented. The system includes legacy systems and cutting edge technology, and its performance is based on their coherent work. The project implementation is dependent on involved participation of multiple governmental and commercial organizations. Moreover, this case-study provides a great investigation in enterprise governance and enterprise transformation beyond a single organization.

References

Works Cited

Primary References
How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn

This article describes a deep space mission where more forthright information exchanges between teamed but rival agencies could have preserved the original plan as well as saved much time and money. The topic may be of particular interest to those involved in institutional collaborations where there are vested interests in protecting rather than sharing information.

For addition information, refer to the closely related topics of Information Management, Organizing Business and Enterprises to Perform Systems Engineering and Fundamentals of Services.

Background

Before the “Faster, Better, Cheaper” philosophy introduced in the 1990s, the United States National Aeronautics and Space Administration (NASA) focused on three classes of unmanned space missions. In order of increasing cost, these were the Discovery, New Frontiers, and Flagship programs. Flagship programs typically cost more than $1B, and included the Voyager (outer planets), Galileo (Jupiter), Cassini-Huygens (Saturn), Mars Science Laboratory (Mars), and the James Webb Space Telescope. (Wall 2012)

The concept of the Cassini-Huygens mission was initiated in 1982 as the result of a working group formed by the National Academy of Sciences and the European Science Foundation. This group sought opportunities for joint space missions; several subsequent reports endorsed the working group’s concept of a Saturn orbiter coupled with a Titan (Saturn’s largest moon) lander. (Russell 2003, p. 61)

By 1988, NASA was politically motivated to reverse earlier tensions with the European Space Agency (ESA) by engaging in a joint mission. Cassini-Huygens was seen as a mechanism to achieve this goal, and the cooperation...
How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn

between NASA and ESA helped the program survive potential budget cuts (since the U.S. was obligated to match ESA commitments). (Russell 2003, p. 62)

NASA and ESA approved the Cassini-Huygens program, and it proceeded under a traditional management approach. NASA built the Cassini orbiter (the largest and most complex unmanned space probe ever built) and the ESA constructed the Huygens lander. This partition of responsibility almost led to the failure (glossary) of the Titan survey portion of the mission. Cassini (which would conduct a variety of scientific surveys of the Saturn planetary system) was expected to relay transmissions from Huygens to NASA's Deep Space Network (DSN); however, the interface between the lander and orbiter was not well-managed and erroneous assumptions about how the orbiter/lander system would behave after separation nearly doomed the Titan exploration portion of the mission. (Oberg 2004)

**Purpose**

The intent of the Titan survey portion of the Cassini-Huygens mission was that the Huygens lander would separate from the Cassini orbiter and commence a one-way, 2.5 hour descent into Titan's atmosphere. Its modest transmitter would send data back to the orbiter, which would relay the information to Earth. (Oberg 2004, p. 30) This effectively made the radio link between the two spacecraft a single point of failure (SPOF) and one that was not well characterized.

Alenia Spazio SpA, the Italian communications vendor that built the radio system, overlooked the Doppler shift (approximately 38 kHz) (Oberg, 2004, p. 31) that would occur when Huygens separated from Cassini and began its descent (Oberg 2004, p. 38). The communications protocol was binary phase-key shifting: "[the] transmission system represents 1s and 0s by varying the phase of the outgoing carrier wave. Recovering these bits requires precise timing: in simple terms, Cassini's receiver is designed to break the incoming signal into 8192 chunks every second. It determines the phase of each chunk compared with an unmodulated wave and outputs a 0 or a 1 accordingly". (Oberg 2004, p. 31) The receiver was appropriately configured to compensate for the Doppler shift of the carrier wave but would be unable to adjust for the Doppler shift of the encoded data. "In effect, the shift would push the signal out of synch with the timing scheme used to recover data from the phase-modulated carrier." (Oberg 2004, p. 33) Therefore, the communications system would be unable to decode the data from the lander and would then relay scrambled information to NASA. Because of the failure mechanism involved, the data would be completely unrecoverable.

Both Cassini and Huygens had been tested before launch; however, none of the testing accurately reflected the Doppler shift that would be experienced at this critical phase of the mission. An opportunity to conduct a full-scale, high-fidelity radio test was ignored due to budget constraints; the testing would have required disassembly and subsequent recertification of the probes. (Oberg, 2004, p. 30) Correcting this latent issue would have been trivial before the spacecraft were launched (via a minor firmware upgrade); (Oberg 2004, p. 33) once they were on the way to Saturn any corrective action would be severely limited and expensive.

Once the mission was underway, the probe coasted along its seven-year trajectory to Saturn and its moons. Claudio Sollazzo, the ESA ground operations manager, was uncomfortable with the untested communications system. He tasked Boris Smeds, an engineer with radio and telemetry experience, with finding a way to test the communications system using an Earth-generated signal. (Oberg 2004, p. 30)

Smeds spent six months developing the test protocols that would use Jet Propulsion Laboratory (JPL) ground stations and an exact duplicate of Huygens. Simulated telemetry would be broadcast from Earth to Cassini and relayed back; the test signal would vary in power level and Doppler shift to fully exercise the communications link and accurately reflect the anticipated parameters during Huygens's descent into Titan's atmosphere. (ESA 2005)
Challenges

Smeds faced opposition to his test plans from those who felt it was unnecessary, but ultimately prevailed due to support from Sollazzo and Jean-Pierre Lebreton, the Huygens project (glossary) scientist. More than two years after the mission was launched, Smeds traveled to a DSN site in California to conduct the test. (Oberg 2004, p. 31)

A test signal was broadcast, received by Cassini, re-transmitted to the DSN site, and relayed to ESA's European Space Operation Centre (ESOC) in Darmstadt, Germany for analysis. Testing had to be conducted when the orbiter was in the correct relative position in the sky; it was more than a quarter of a million miles away with a signal round-trip time of nearly an hour. The test immediately exposed an issue; the data stream was intermittently corrupted, with failures not correlated to the power level of the test signal. The first of two days of testing concluded with no clear root cause identified. (Oberg 2004, p. 31)

Even though the probe was far from its ultimate destination, many science teams were competing for time to communicate with it using the limited bandwidth available. The communications team would not be able to conduct another set of trials for several months. Smeds diagnosed the root cause of the problem; he felt it was the Doppler shifts induced in the simulated signal. However, the test plan did not include unshifted telemetry (an ironic oversight). He modified his test plan overnight and shortened the planned tests by 60%; this recovered sufficient time for him to inject an unshifted signal into the test protocols. (Oberg 2004, p. 32)

This unshifted signal did not suffer from the same degradation; however, other engineers resisted the diagnosis of the problem. Follow-up testing using probe mockups and other equipment ultimately convinced the ESA of the issue; this took an additional seven months. (Oberg 2004, p. 33)

By late 2000, ESA informed NASA of the latent failure of the communications link between Cassini and Huygens. Inquiry boards confirmed that Alenia Spazio had reused timing features of a communications system used on Earth-orbiting satellites (which did not have to compensate for Doppler shifts of this magnitude). (Oberg, 2004, p. 33) In addition, because NASA was considered a competitor, full specifications for the communications modules were not shared with JPL. The implementation of the communications protocols was in the system’s firmware; trivial to correct before launch, impossible to correct after. (ESA 2005)

A 40-man Huygens Recovery Task Force (HRTF) was created in early 2001 to investigate potential mitigation actions. Analysis showed that no amount of modification to the signal would prevent degradation; the team (glossary) ultimately proposed changing the trajectory of Cassini to reduce the Doppler shift. (ESA 2005) Multiple studies were conducted to verify the efficacy of this remedy, and it ultimately allowed the mission to successfully complete the Titan survey.

Systems Engineering Practices

Space missions are particularly challenging; once the spacecraft is en route to its destination, it is completely isolated. No additional resources can be provided and repair (particularly for unmanned mission) can be impossible. Apollo 13’s crew barely survived the notable mishap on its mission because of the resources of the docked Lunar Excursion Module (LEM) and the resourcefulness of the ground control team’s experts. A less well-known failure occurred during the Galileo mission to Jupiter. After the Challenger disaster, NASA adopted safety standards that restricted the size of boosters carried in the Space Shuttle. (Renzetti 1995) Galileo was delayed while the Shuttles were grounded and Galileo’s trajectory was re-planned to include a Venus fly-by to accelerate and compensate for a smaller booster. Galileo’s main antenna failed to deploy; lubricant had evaporated during the extended unplanned storage (Evans 2003) and limited computer space led to the deletion of the antenna motor-reversing software to make room for thermal protection routines. When the antenna partially deployed, it was stuck in place with no way to re-furl and redeploy it. Engineers ultimately used an onboard tape recorder, revised transmission protocols, the available low-gain antenna, and ground-based upgrades to the DSN to save the mission. (Taylor, Cheung, and Seo 2002)
The Titan survey was ultimately successful because simulation (glossary) techniques were able to verify the planned trajectory modifications and sufficient reaction mass was available to complete the necessary maneuvers. In addition, Smeds’s analysis gave the mission team the time it needed to fully diagnose the problem and develop and implement the remedy. If this test were conducted the day before the survey it would merely have given NASA and ESA advance warning of a disaster. The time provided enabled the mission planners to craft a trajectory that resolved the communication issue and then blended back into the original mission profile to preserve the balance of the Saturn fly-bys planned for Cassini. (Oberg 2004, p. 33)

Lessons Learned

The near-failure of the Cassini-Huygens survey of Titan was averted because a handful of dedicated systems engineers fought for and conducted relevant testing, exposed a latent defect, and did so early enough in the mission to allow for a recovery (glossary) plan to be developed and executed. Root causes of the issue included politically-driven partitioning, poor interface management, overlooked contextual information, and a lack of appreciation for single-points-of-failure (SPOFs).

The desire to use a joint space mission as a mechanism for bringing NASA and ESA closer together (with the associated positive impact in foreign relations) introduced an unnecessary interface into the system. Interfaces must always be managed carefully; interfaces between organizations (particularly those that cross organizational or political borders) require extra effort and attention. Boeing and Airbus experienced similar issues during the development of the Boeing 787 and A380; international interfaces in the design (glossary) activities and supply chains led to issues:

...every interface in nature has a surface energy. Creating a new surface (e.g., by cutting a block of steel into two pieces) consumes energy that is then bound up in that surface (or interface). Interfaces in human systems (or organizations), a critical aspect of complex systems such as these, also have costs in the effort to create and maintain them. Second, friction reduces performance. Carl von Clausewitz, the noted military strategist, defined friction as the disparity between the ideal performance of units, organizations, or systems, and their actual performance in real-world scenarios. One of the primary causes of friction is ambiguous or unclear information. Partitioning any system introduces friction at the interface. (Vinarcik 2014, p. 697)

Alenia Spazio SpA’s unclear understanding of the Doppler shift introduced by the planned relative trajectories of Huygens and Cassini during the Titan survey led it to reuse a component from Earth-orbiting satellites. Because it considered NASA a competitor and cloaked details of the communications system behind a veil of propriety, it prevented detection of this flaw in the design phase. (Oberg 2004, p. 33)

Because NASA and ESA did not identify this communication link as a critical SPOF, they both sacrificed pre-launch testing on the altar of expediency and cost-savings. This prevented detection and correction of the flaw before the mission was dispatched to Saturn. The resource cost of the later analysis and remedial action was non-trivial and if sufficient time and reaction mass had not been available the mission would have been compromised. It should be noted that a number of recent spacecraft failures are directly attributable to SPOFs (notably, the Mars Polar Lander (JPL 2000) and the Genesis sample return mission (GENESIS, 2005)). Effective SPOF detection and remediation must be a priority for any product development effort. More generally, early in the development process, significant emphasis should be placed on analyses focused on what might go wrong (“rainy day scenarios”) in addition to what is expected to go right (“sunny day scenarios”).

The success of the Huygens survey of Titan was built upon the foundation established by Boris Smeds by identifying the root cause of the design flaws in a critical communications link. This case study underscores the need for clear contextual understanding, robust interface management, representative testing, and proper characterization and management of SPOFs.
References

Works Cited

Primary References
None.

Additional References
None.

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.8 released 27 March 2017

SEBoK Discussion
Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.
Global Positioning System Case Study

The Global Positioning System (GPS) case study was developed by the United States Air Force Center for Systems Engineering (AF CSE) located at the Air Force Institute of Technology (AFIT). The GPS is a space-based radio-positioning system. A constellation of twenty-four satellites, including three spares, comprise the overall system which provides navigation and timing information to military and civilian users worldwide. GPS satellites, in one of six Earth orbits, circle the globe every twelve hours, emitting continuous navigation signals on two different L-band frequencies. The system consists of two other major segments: a world-wide satellite control network, and the GPS user equipment that can either be carried by a human user or integrated into host platforms such as ships, vehicles, or aircraft.

This case study discussion is based on the original source (O’Brien and Griffin 2007) which provides useful insights into what we might consider a “traditional” SE application. A second Global Positioning System Case Study II looks at the same case study from the perspectives of System of Systems (SoS) (glossary) engineering and Enterprise Systems Engineering (ESE) (glossary).

Domain Background

When looking at the Global Positioning System (GPS), it would be difficult to imagine another system that relies so heavily upon such a wide range of domains, with the possible exception of the World Wide Web (WWW). Additionally, the various systems operating within these domains must all function together flawlessly to achieve success. It is evident from reading this case study that it directly relates to the following domains:

- aerospace;
- space;
- communications; and
- transportation.

This is also an example of systems of systems (SoS) and is considered an innovative technology.

The GPS case study includes a detailed discussion of the development of the GPS and its components, as well as other applicable areas. The reader of this study will gain an increased understanding of the effect that GPS has on military and commercial industries in the context of the systems engineering support required to achieve success.

Case Study Background

The United States Air Force Center for Systems Engineering (AF CSE), established in 2002 at the Air Force Institute of Technology (AFIT), was tasked to develop case studies focusing on the application of systems engineering principles within various aerospace programs. The GPS case study (O’Brien and Griffin 2007) was developed by AFIT in support of systems engineering graduate school instruction. The cases are structured using the Friedman-Sage framework (Friedman and Sage 2003; Friedman and Sage 2004, 84-96), which decomposes a case into contractor, government, and shared responsibilities in the following nine concept areas:

1. Requirements Definition and Management
2. Systems Architecture Development
3. System/Subsystem Design
4. Verification/Validation
5. Risk Management
6. Systems Integration and Interfaces
7. Life Cycle Support
8. Deployment and Post Deployment
9. System and Program Management

The Friedman-Sage framework (2004) is provided in Appendix A of the case study. This case study is an example where the government - specifically the JPO Systems Engineering Directorate - bore the responsibility for systems integration and configuration management. That is, the government played more than an oversight role in the systems engineering of the GPS system of systems. As mentioned in the case study, JPO developed the CONOPs, mission analysis, requirements and design analysis including security, and developed their own approach to the cryptology methodology. JPO coordinated the Configuration Control Board (CCB) chaired by the Program Director. JPO was also responsible for Level I ICDs and system design configurations; where the contractors were responsible for the system architecture and ICDs within their segment.

Case Study Description

The “Global Positioning System - Systems Engineering Case Study” describes the application of systems engineering during the concept validation, system design and development, and production phases of the GPS program (O'Brien and Griffin 2007). The case examines the applied systems engineering processes, as well as the interactions of the GPS joint program office (JPO), the prime contractors, and the plethora of government agencies that were associated with the program’s development and fielding. The systems engineering process is traced from the initiation of studies and the development of key technologies, which established the vision of a satellite navigation system in the 1960s, through to the multiphase joint-program that resulted in a fully operational capability release in 1995. This case study does not cover system enhancements incorporated through Blocks IIM, IIF, and III.

The GPS case study derived four learning principles (LPs) that explain the more broadly applicable areas of systems engineering knowledge that are addressed by the case study. These four LPs relate strongly to the SEBoK in the following areas:

• enabling individuals (LP1);
• configuration management (LP2);
• enabling the organization (LP3); and
• risk management (LP4).

Additionally, the GPS case study contains a thorough overview of life cycle management and exemplifies systems thinking principles.

Enabling Individuals

Learning Principle 1: Programs must strive to staff key positions with domain experts.

From the program management team, to the systems engineering, design, manufacturing, and operations teams, the individuals on the program were well-versed in their disciplines and all possessed a systems view of the program. While communications, working relationships, and organization were important, it was the ability of the whole team at all levels to understand the implications of their work on the system that was vital. Their knowledge-based approach for decision making had the effect of shortening the decision cycle because the information was understood and the base and alternative solutions were accurately presented.
Configuration Management

Learning Principle 2: The systems integrator must rigorously maintain program baselines.

The joint program office (JPO) retained the role of managing and controlling the system specification and, therefore, the functional baseline. The JPO derived and constructed a mutually agreed to set of system requirements that became the program baseline in 1973. While conducting the development program, the GPS team was able to make performance, risk, cost, and trade analyses against the functional baseline to control both risk and cost. The JPO was fully cognizant of the implications of the functional requirements on the allocated baseline because they managed the interface control working group process. Managing that process gave them first-hand knowledge and insight into the risks at the lowest level. The individual with the system integrator role must rigorously maintain the system specification and functional baseline. There must be appropriate sharing of management and technical responsibilities between the prime contractor and their government counterparts to ensure success.

Enabling the Organization

Learning Principle 3: Achieving consistent and continuous high-level support and advocacy helps funding stability, which impacts systems engineering stability.

Consistent, continuous high-level support provides the requirements and assists funding stability. In this role, the Office of the Secretary of Defense (OSD) provided advocacy and sourced the funding at critical times in the program, promoted coordination among the various services, and reviewed and approved the GPS JPO system requirements. The OSD played the central role in the establishment and survivability of the program. The GPS JPO had clear support from the Director of Defense Development, Research, and Engineering, Dr. Malcolm Currie, and program support from the Deputy Secretary of Defense, Dr. David Packard. Clearly, the armed services—particularly the Navy and the Air Force early on, and later the Army—were the primary users of GPS and the eventual customers. However, each armed service had initial needs for their individual programs, or for the then-current operational navigation systems. Additionally, the secretary of the Air Force provided programmatic support to supply manpower and facilities.

Risk Management

Learning Principle 4: Disciplined and appropriate risk management must be applied throughout the life cycle.

The GPS program was structured to address risk in several different ways throughout the multiphase program. Where key risks were known up front, the contractor and/or the government utilized a classic risk management approach to identify and analyze risk, as well as develop and track mitigation actions. These design (or manufacturing/launch) risks were managed by the office who owned the risks. Identified technical risks were often tracked by technical performance measures (such as satellite weight and software lines of codes) and addressed at weekly chief engineer’s meetings.

Serving in the clear role of program integrator allowed the JPO to sponsor risk trade studies at the top level. The JPO would issue study requests for proposals to several bidders for developing concepts and/or preliminary designs. Then, one contractor would be down-selected and the process would continue. This approach provided innovative solutions through competition, as well as helped in defining a lower risk, more clearly defined development program for the fixed-price contracts approach that was being used for development and production.

As the system integrator, the JPO was also closely involved with technical development. To identify unforeseeable unique technical challenges, the JPO would fund studies to determine the optimal approaches to new issues. There were schedule risks associated with the first launch due to unforeseen Block II issues with respect to the space vehicle and control segments (software development). Although a catastrophic event, the Challenger accident actually provided much needed schedule relief. Using decision analysis methodology led the JPO to an alternative approach to develop the expendable launch vehicle for the Block II satellites.
Good communication, facilitated by cooperative working relationships, was a significantly positive (though intangible) factor in the success of the GPS program, regardless of whether it was between the contractors and the government (JPO or other agencies), or between contractors and sub-contractors. A true team environment also played a significant role in reducing risk, especially considering the plethora of government agencies and contractors that were involved in the effort.

**Life Cycle Management**

The GPS case study takes the reader through the initial concept of GPS (March 1942) all the way to the development, production, and operational capability of the system. The current GPS program traces its heritage to the early 1960s when Air Force Systems Command initiated satellite-based navigation systems analyses conducted by The Aerospace Corporation. The case study follows the execution of the GPS program from the inception of the idea to the full operational capability release on April 27th, 1995. The concentration of the case study is not limited to any particular period, and the learning principles come from various times throughout the program's life.

**Systems Thinking**

The GPS case study highlights the need for systems thinking throughout. GPS satellites, in one of six Earth orbits, circle the globe every twelve hours. These satellites emit continuous navigation signals on two different L-band frequencies. The system consists of two other major segments: a world-wide satellite control network and the GPS user equipment that can either be carried by a human user, or integrated into host platforms such as ships, vehicles, or aircraft. The ability to conceive, develop, produce, field, and sustain the GPS demands the highest levels of systems thinking.

**Summary**

The GPS case study is useful for global systems engineering learning and provides a comprehensive perspective on the systems engineering life cycle. The study is applicable for detailed instruction in the following areas:

- enabling individuals;
- configuration management;
- enabling the organization;
- risk management;
- life cycle management; and
- systems thinking.

The GPS case study revealed that key Department of Defense personnel maintained a clear and consistent vision for this unprecedented, space-based navigation capability. The case study also revealed that good fortune was enjoyed by the JPO as somewhat independent, yet critical, space technologies matured in a timely manner. Although the GPS program required a large degree of integration, both within the system and external to the system amongst a multitude of agencies and contractors, the necessary efforts were taken to achieve success.

Lastly, the reader of the GPS case study will gain an increased understanding of the effect that GPS has on the military and commercial industries in the context of the systems engineering support required to achieve success. The system was originally designed to help "drop five bombs in one hole" which defines the accuracy requirement in context-specific terms. The GPS signals needed to be consistent, repeatable, and accurate to a degree that, when used by munitions guidance systems, would result in the successful delivery of multiple, separately-guided munitions to virtually the identical location anywhere at any time across the planet. Forty to fifty years ago, very few outside of the military recognized the value of the proposed accuracy and most non-military uses of GPS were not recognized before 1990. GPS has increasingly grown in use and is now used every day.
References

Works Cited


Primary References


Additional References

none.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.8 released 27 March 2017

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox[^1].

[^1]: ENCODED_CONTENT

NDgwODMPGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHwdHlwZT0idGV4dC9qYXZhc2Ny

END ENCODED_CONTENT
Global Positioning System Case Study II

This article highlights some of the differences between the so-called classical, traditional, or conventional systems engineering (SE) approaches and the newer, and, as yet, less defined principles of system of systems (SoS) engineering (SoSE), enterprise systems engineering (ESE), and/or complex systems engineering (CSE) or complex adaptive systems engineering (Gorod et al. 2015). The topic is still somewhat controversial, especially considering those that are sceptical that broader views of SE might work better when one is immersed in trying to cope with our most difficult problems. Indeed, the lack of a unified theory of SE is one of the prime motivations for producing and analysing case studies to develop more knowledge of what seems to work, what does not seem to work, and reasons why, really challenging SE environments.

For additional information, refer to Systems Engineering: Historic and Future Challenges, Systems Engineering and Other Disciplines, Enterprise Systems Engineering, and System of Systems Engineering.

Rather than modifying the previous discussion of the Global Positioning System Case Study in SEBoK, the focus is on comparing and contrasting the older and newer forms of SE by commenting on quotations from the original case study source documents (O'Brien and Griffin 2007).

Preface

The original case study begins by describing systems engineering (SE) principles. For example,

System requirements are critical to all facets of successful system program development. First, system development must proceed from a well-developed set of requirements. Second, regardless of the evolutionary acquisition approach, the system requirements must flow down to all subsystems and lower-level components. And third, the system requirements must be stable, balanced, and must properly reflect all activities in all intended environments. However, system requirements are not unchangeable. As the system design proceeds, if a requirement or set of requirements is proving excessively expensive to satisfy, the process must rebalance schedule, costs, and performance by changing or modifying the requirements or set of requirements. (O'Brien and Griffin 2007, p. 9)

The Global Positioning System (GPS), including its multi-various applications, was developed over many years as the result of the efforts of a host of contributors. It is very difficult to believe that the classical, traditional or conventional systems engineering approach described in the above paragraph (especially those phrases highlighted in bold by the present authors) was truly responsible for this remarkable achievement that so profoundly impacts our lives. Rather, some more advanced form of systems engineering (SE), that might be called, system of systems engineering (SoSE), enterprise systems engineering (ESE), or complex (adaptive) systems engineering (CSE), or a blend and/or combination of these approaches or methodologies, had to be responsible. This premise is supported explicitly and repeatedly in the following case study revision using bold font.

Continuing, the following quoted paragraphs seem flawed in several places highlighted in bold. The bold phrases might be replaced by the phrases in brackets […]. Such brackets might also include other editorial comments of the present authors.

Systems engineering includes making key system and design trades early in the process to establish the system architecture. These architectural artifacts This architecture can depict any new system, legacy system, modifications thereto, introduction of new technologies, and overall system-level behavior and performance. Modeling and simulation are generally employed to organize and assess architectural system alternatives at this stage. System and subsystem design follows the functional [system] architecture [as defined from a functional point of view]. System architectures designs are modified if elements are too risky, expensive, or time-consuming. (O'Brien and Griffin 2007, p. 9)
A good architecture, once established, should guide systems development, and not change very much, if at all, at least compared to possible changes in the system design, which, of course, can evolve as one learns more about the problem and potential solutions that may increase the system's capability. Thus, it is crucial to not confuse architecture with designs instantiating the architecture, contrary to what seems to be the case in (Ricci, et al. 2013).

Important to the efficient decomposition and creation of functional and physical architectural designs are the management of interfaces and the integration of subsystems. Interface management and integration is applied to subsystems within a system or across a large, complex system of systems. Once a solution is planned, analyzed, designed, and constructed, validation and verification take place to ensure satisfaction of requirements. Definition of test criteria, measures of effectiveness (MOEs), and measures of performance (MOPs) are established as part of the requirements process, taking place well before any component/subsystem assembly design and construction occurs. (O'Brien and Griffin 2007, p. 10)

In the quoted paragraph just above bold phrases note the emphasis on a reductionist approach, reductionism, where great attention is paid to the subsystems and managing the interfaces among them. This is the antithesis of a holistic approach where one concentrates on the whole system, recognizing that it is difficult to identify overall system behavior as depending on any particular subsystem or set of subsystems. In a truly complex system that is continually evolving, the above-mentioned requirements process is flawed because the system is continually changing, i.e., the system is evolutionary; the requirements are either ill-defined at the outset, or are modified because stakeholders change their minds, or become somewhat irrelevant because the system environment changes.

There are several excellent representations of the [usual traditional or conventional] systems engineering process presented in the literature. These depictions present the current state of the art in maturity and evaluation of the systems engineering process. One can find systems engineering process definitions, guides, and handbooks from the International Council on Systems Engineering (INCOSE), European Industrial Association (EIA), Institute of Electrical and Electronics Engineers (IEEE), and various Department of Defense (DoD) agencies and organizations. They show the process as it should be applied [Really? In all situations?] by today's experienced practitioner. One of these processes, long used by the Defense Acquisition University (DAU), is [a model] not accomplished in a single pass. This iterative and nested process gets repeated to the lowest level of definition of the design and its interfaces. (O'Brien and Griffin 2007, p. 10)

The above description appears to be written with pride without any acknowledgement that this SE methodology might fail to work if applied according to these guidelines, or that there might be new SE techniques that could be more effective in some situations. Again, this reflects a reductionist approach that ignores holism and emergent properties that might not be explained even when thoroughly understanding the systems components and their interactions. On the positive side, the next paragraph suggest how the world is changing and hints that something more is needed. Nevertheless, the advice seems to be oriented toward applying the existing SE discipline more vigorously instead of seeking new methods that might be more effective.

The DAU model, like all others, has been documented in the last two decades, and has expanded and developed to reflect a changing environment. Systems are becoming increasingly complex internally and more interconnected externally. The process used to develop aircraft and systems of the past was effective at the time. It served the needs of the practitioners and resulted in many successful systems in our inventory. Notwithstanding, the cost and schedule performance of the past programs are replete with examples of well-managed programs and ones with less-stellar execution. As the nation entered the 1980s and 1990s, large DoD and commercial acquisitions experienced overrunning costs and slipping schedules. The aerospace industry and its organizations were becoming larger and were more geographically and culturally distributed. Large aerospace companies have worked diligently to establish common systems engineering practices across their enterprises. However, because of the mega-trend of teaming in large (and some small) programs, these common practices must be understood
and used beyond the enterprise and to multiple corporations. It is essential that the systems engineering process govern integration, balance, allocation, and verification, and be useful to the entire program team down to the design and interface level. (O’Brien and Griffin 2007, p. 11)

Finally, in the next paragraph there is a suggestion that SE could be made more sophisticated but there is no mention of addressing people problems or advocating a broader transdisciplinary approach.

Today, many factors overshadow new acquisition; including system-of-systems (SoS) context, network centric warfare and operations, and rapid growth in information technology. These factors are driving a more sophisticated systems engineering process with more complex and capable features, along with new tools and procedures. One area of increased focus of the systems engineering process is the informational systems architectural definitions used during system analysis. This process, described in DoD Architectural Framework (DoDAF), emphasizes greater reliance on reusable architectural views describing the system context and concept of operations, interoperability, information and data flows, and network service-oriented characteristics. (O’Brien and Griffin 2007, p. 11)

The last two sections of the systems engineering principles portion of the original case study address case studies themselves, mainly for academic purposes, to help people appreciate systems engineering principles, and the framework used in the case study, namely the rather narrowly defined Friedman-Sage framework that will be discussed briefly in Section II below.

The treatment of the reason for case studies is quite good in that it talks about the benefits of applying systems engineering principles, as highlighted from real-world examples of what works and what does not. Except near the end, where there is allusion to the possibility of new endeavor systems engineering principles, the principles espoused tend to be traditional or conventional.

On the other hand, based upon the original case study (O’Brien and Griffin 2007), if one views the boundary of the GPS system to include primarily the technology associated with the GPS space segment and its controlling ground network, then it can be assumed that system was likely implemented primarily by following traditional or conventional systems engineering processes. If one takes this viewpoint, then all of the above criticism which attempts to point out some of the shortcomings of conventional systems engineering, may seem vacuous at best, or politically incorrect at worst. It may well be that many would rather not denigrate the original GPS case study by exposing it to the possibilities of a broader system engineering approach.

Unless otherwise indicated, as the present authors have already been doing, unchanged quotations from the existing SEBoK are indented below. Modifications to such quotations are shown in brackets [...]; deletions are not necessarily shown explicitly.

**Background**

The Global Positioning System (GPS) case study was developed by the United States Air Force Center for Systems Engineering (AF CSE) located at the Air Force Institute of Technology (AFIT). The GPS is a space-based radio-positioning system. A constellation of twenty-four satellites, including three spares, comprise the overall system which provides navigation and timing information to military and civilian users worldwide. GPS satellites, in one of six Earth orbits, circle the globe every twelve hours, emitting continuous navigation signals on two different L-band frequencies. The system consists of two other major segments: a world-wide satellite control network, and the GPS user equipment that can either be carried by a human user or integrated into host platforms such as ships, vehicles, or aircraft.

A user needs to receive signals from at least four GPS satellites simultaneously (satellite orbital positions and terrestrial terrain blockage can be issues that degrade performance) to determine one's position in three dimensions; the altitude determination is typically less accurate than the other two dimensions.
When looking at [GPS], it would be difficult to imagine another system that relies so heavily upon such a wide range of [domains containing systems that must interact effectively to achieve successful GPS operation]. It is evident that [GPS directly relates to many domains and applications including:

- position location and tracking
- time synchronization
- navigation
- transportation
- times of arrival
- air traffic management
- situational awareness
- jam-resistant communications
- business and commerce
- farming
- aerospace
- sensing nuclear detonations from space
- military war-fighting
- targeting
- weapons delivery
- etc.]

[GPS is] an example of [a collaborative (Dahmann, et al. 2008) systems of systems (SoS)]. As such, no one is in charge, and the capabilities (not requirements) flow from the bottom-up, as opposed to top-down.

**Purpose**

The GPS case study includes a detailed discussion of the development of the GPS and its components, as well as other applicable areas. The reader of this study will gain an increased understanding of the effect that GPS has on military and commercial industries in the context of the systems engineering support required to achieve success.

This may be, but the principal purpose of this revised case study is to suggest a broader view of GPS that discusses signature aspects of SoS, enterprises, and complex systems, and emphasizes SoSE, ESE, and CSE.

[AF CSE] was tasked to develop case studies focusing on the application of [SE] principles within various aerospace programs. The GPS case study [was developed in support of SE] graduate school instruction using the Friedman-Sage framework (Friedman and Sage 2003) (Friedman and Sage 2004).

However, the Friedman-Sage framework involves only two contractual stakeholders, the Government and the contractor; further, the framework is limited to the traditional or conventional SE life cycle which mainly treats activities in a linear instead of nonlinear fashion; still further, only risks are considered, not a balance of risk and opportunity. Thus, the present authors believe a broader framework embracing SoSE, ESE, and CSE is more appropriate.

**Challenges**

In the original case study the first highly technical section (Section 2) was the system description. The original idea derived from trying to determine the precise orbital parameters of the first artificial satellites such as Sputnik launched by the Soviets in 1957. Researchers at Johns Hopkins realized the inverse, that if one knew precisely the orbital parameters, the locations of ground stations receiving satellite signals could be determined quite accurately. (O’Brien and Griffin 2007, p. 20)
GPS got its start in the early 70s (O’Brien and Griffin 2007, p. 19) building upon several previous satellite navigation systems. The primary motive was very accurate position information for the purposes of military applications. For example, the U.S. Air Force wanted to deliver nuclear weapons from bombers with unprecedented accuracy and precision. (O’Brien and Griffin 2007, p. 29)

With such an intense interest from the military, the first real challenge, other than the many technical challenges of making GPS work as well as envisioned, might have been the question of how to make GPS available to the civilian community so they could share the benefits. The study claimed that the system was always offered for civilian use, albeit with some charge. After the Korean airliner went astray and got shot down by a Soviet interceptor aircraft, President Reagan made GPS officially available for civilian use free of charge. (O’Brien and Griffin 2007, p. 14)

The second challenge could be associated with preserving precision capabilities for the military only, and relegating course acquisition (C/A) accuracy to the civilian community. (O’Brien and Griffin 2007, p. 15) Later this dichotomy was essentially eliminated with the realization that a differential GPS configuration involving a fixed ground station with a precisely known location will yield great accuracy. (Kee, et al. 1991)

The GPS satellites used space-borne atomic clocks. To alleviate the need for updating these clocks too often a successful effort was initiated to revise the international time standard which ended up using relatively infrequent “leap seconds”. (O’Brien and Griffin 2007, p. 23) Even these are still annoying for many other applications, such as the continual need to achieve precise synchronization of frequency hopping radios.

An organizational challenge of inter-service rivalries was overcome with the formation of the Joint Program Office (JPO). (O’Brien and Griffin 2007, p. 25)

In the early days of satellite communication systems, for example, the satellites were quite small and low powered while the terminals were large and high-powered. By the time GPS came along, the satellites are getting bigger and more sophisticated. Then the challenge to develop relatively low-cost terminals, particularly for mobile users, greatly increased. (O’Brien and Griffin 2007, p. 29)

A small but interesting challenge was the definition of system of systems (SoS). It was decided that GPS was an SoS because it involved three independent systems, namely, the space vehicle (SV), the control segment (CS), and the user equipment (UE), that “merely” had to interface with each other. (O’Brien and Griffin 2007, p. 30)

Continually changing requirements is usually a problem, although in this case the requirements did not change as often as they could have. (O’Brien and Griffin 2007, p. 31)

Difficulties of defining and updating the many GPS interfaces was largely overcome by the GPS program director, Col. Brad Parkinson, when he convinced his own management, Gen. Schultz at Space and Missile Systems Office (SAMSO) (which eventually became the Space Division) that GPS ought to be defined solely by the signal-structure-in-space and not the physical interfaces. (O’Brien and Griffin 2007, p. 31)

**Systems Engineering Practices**

Although the systems engineering process in Phase I has been discussed previously, this section will expand on the concepts. For example, one of the user equipment contractors was technically competent, but lacked effective management. The JPO strongly suggested that a systems engineering firm be hired to assist the contractor in managing program and they agreed. (O’Brien and Griffin 2007, p. 42)

There did not seem to be any mention of what SE firm was hired, if any. The Aerospace Corporation, a non-profit Federally Funded Research and Development Center (FFRDC), which had such a key role in the run-up to GPS was also prominently and centrally involved in development phase of this humungous project. (O’Brien and Griffin 2007, pp. 20, 22, 25, 33, 34, 40, 41, 44, 48, 50-52, 56, 57, 62, 63, 64, 66, 67, 71)
Lessons Learned

Communications was a key ingredient that was fostered throughout GPS development. (O’Brien and Griffin 2007, p. 71)

Yes, from reading the original case study there seems to have been a lot of cooperation among the various organizations, more so than might have been expected in a less compelling case.

Several precepts or foundations of the Global Positioning Satellite program are the reasons for its success. These foundations are instructional for today’s programs because they are thought-provoking to those who always seek insight into the program’s progress under scrutiny. These foundations of past programs are, of course, not a complete set of necessary and sufficient conditions. For the practitioner, the successful application of different systems engineering processes is required throughout the continuum of a program, from the concept idea to the usage and eventual disposal of the system. Experienced people applying sound systems engineering principles, practices, processes, and tools are necessary every step of the way. Mr. Conley, formerly of the GPS JPO, provided these words: “Systems engineering is hard work. It requires knowledgeable people who have a vision of the program combined with an eye for detail.” (O’Brien and Griffin 2007, p. 72)

In very complex systems engineering efforts of this type, it is also important to explore new techniques that attempt to deal with “soft” issues involving people. Those that seem to work can be added to the systems engineering process collection.

Systems engineering played a major role in the success of this program. The challenges of integrating new technologies, identifying system requirements, incorporating a system of systems approach, interfacing with a plethora of government and industry agencies, and dealing with the lack of an operational user early in the program formation required a strong, efficient systems engineering process. The GPS program embedded systems engineering in their knowledge-base, vision, and day-to-day practice to ensure proper identification of system requirements. It also ensured the allocation of those requirements to the almost-autonomous segment developments and beyond to the subcontractor/vendor level, the assessments of new requirements, innovative test methods to verify design performance to the requirements, a solid concept of operations/mission analysis, a cost-benefit analysis to defend the need for the program, and a strong system integration process to identify and control the “hydra” of interfaces that the program encountered. The program was able to avoid major risks by their acquisition strategy, the use of trade studies, early testing of concept designs, a detailed knowledge of the subject matter, and the vision of the program on both the government and contractor side. (O’Brien and Griffin 2007, p. 72)

This well summarizes the successful systems engineering approach utilized in GPS. Another element of achieving overall balance is the pursuit of opportunities as the “flipside” of risk mitigation.

Finally, here is the list of academic questions offered in original case study.

QUESTIONS FOR THE STUDENT (O’Brien and Griffin 2007, p. 73) The following questions are meant to challenge the reader and prepare for a case discussion.

• Is this program start typical of an ARPA/ DARPA funded effort? Why or why not?
• Have you experiences similar or wildly different aspects of a Joint Program?
• What were some characteristics that should be modeled from the JPO?
• Think about the staffing for the GPS JPO. How can this be described? Should it be duplicated in today’s programs? Can it?
• Was there anything extraordinary about the support for this program?
• What risks were present throughout the GPS program. How were these handled?
• Requirement management and stability is often cited as a central problem in DoD acquisition. How was this program like, or [un]like, most others?
Could the commercial aspects of the User Equipment be predicted or planned? Should the COTS aspect be a strategy in other DoD programs, where appropriate? Why or why not?

Other questions might be: What possible influences did the demand for or offering to the public of this GPS capability entail? What differences in the development of GPS might have emerged if the public was more aware of the potential applications for their benefit at the outset?

References

Works Cited


Primary References


Additional References

None
**Medical Radiation Case Study**

This case study presents system and software engineering issues relevant to the accidents associated with the Therac-25 medical linear accelerator that occurred between 1985 and 1988. The six accidents caused five deaths and serious injury to several patients. The accidents were system accidents that resulted from complex interactions between hardware components, controlling software, and operator functions.

**Domain Background**

Medical linear accelerators, devices used to treat cancer, accelerate electrons to create high energy beams that can destroy tumors. Shallow tissue is treated with the accelerated electrons. The electron beam is converted to X-ray photons to reach deeper tissues. Accidents occur when a patient is delivered an unsafe amount of radiation.

A radiation therapy machine is controlled by software that monitors the machine's status, accepts operator input about the radiation treatment to be performed, and initializes the machine to perform the treatment. The software turns the electron beam on in response to an operator command. The software turns the beam off whenever the treatment is complete, the operator requests the beam to shutdown, or when the hardware detects a machine malfunction. A radiation therapy machine is a reactive system in which the system's behavior is state dependent and the system's safety depends upon preventing entry into unsafe states. For example, the software controls the equipment that positions the patient and the beam. The positioning operations can take a minute or more to execute, thus it is unsafe to activate the electron beam while a positioning operation is in process.

In the early 1980s, Atomic Energy of Canada (AECL) developed the Therac-25, a dual-mode (X-rays or electrons) linear accelerator that can deliver photons at 25 meagelectron volts (MeV) or electrons at various energy levels. The Therac-25 superseded the Therac-20, the previous 20-MeV dual mode accelerator with a history of successful clinical use. The Therac-20 used a DEC PDP-11 (Digital Equipment Corporation Programmed Data Processor) minicomputer for computer control and featured protective circuits for monitoring the electron beam, as well as mechanical interlocks for policing the machine to ensure safe operation. AECL decided to increase the responsibilities of the Therac-25 software for maintaining safety and eliminated most of the hardware safety mechanisms and interlocks. The software, written in PDP-11 assembly language, was partially reused from earlier products in the Therac product line. Eleven Therac-25s were installed at the time of the first radiation accident in June 1985.
The use of radiation therapy machines has increased rapidly in the last 25 years. The number of medical radiation machines in the United States in 1985 was approximately 1000. By 2009 the number had increased to approximately 4450. Some of the types of system problems found in the Therac-25 may be present in the medical radiation devices currently in use. References to more recent accidents are included below.

Case Study Background
The Therac-25 accidents and their causes are well documented in materials from the U.S. and Canadian regulatory agencies (e.g., the U.S. Food and Drug Administration (FDA) and the Canadian Bureau of Radiation and Medical Devices) and in depositions associated with lawsuits brought against AECL. An article by Leveson and Turner (1993) provides the most comprehensive, publicly available description of the accident investigations, the causes of the accidents, and the lessons learned relevant to developing systems where computers control dangerous devices.

Case Study Description
The Therac-25 accidents are associated with the non-use or misuse of numerous system engineering practices, especially system verification and validation, risk management, and assessment and control. In addition, numerous software engineering good practices were not followed, including design reviews, adequate documentation, and comprehensive software unit and integration tests.

The possibility of radiation accidents increased when AECL made the systems engineering decision to increase the responsibilities of the Therac-25 software for maintaining safety and eliminated most of the hardware safety mechanisms and interlocks. In retrospect, the software was not worthy of such trust. In 1983 AECL performed a safety assessment on the Therac-25. The resulting fault tree did include computer failures, but only those associated with hardware; software failures were not considered in the analysis.

The software was developed by a single individual using PDP-11 assembly language. Little software documentation was produced during development. An AECL response to the FDA indicated the lack of software specifications and of a software test plan. Integrated system testing was employed almost exclusively. Leveson and Turner (1993) described the functions and design of the software and concluded that there were design errors in how concurrent processing was handled. Race conditions resulting from the implementation of multitasking also contributed to the accidents.

AECL technical management did not believe that there were any conditions under which the Therac-25 could cause radiation overdoses, and this belief was evident in the company's initial responses to accident reports. The first radiation overdose accident occurred in June 1985 at the Kennestone Regional Oncology Center in Marietta, Georgia, where the Therac-25 had been operating for about 6 months. The patient who suffered the radiation overdose filed suit against the hospital and AECL in October 1985. No AECL investigation of the incident occurred and FDA investigators later found that AECL had no mechanism in place to follow up potential reports of suspected accidents. Additionally, other Therac-25 users received no information that an accident had occurred.

Two more accidents occurred in 1985, including a radiation overdose at Yakima Valley Memorial Hospital in Yakima, Washington that resulted in an accident report to AECL. The AECL technical support supervisor responded to the hospital in early 1986: "After careful consideration, we are of the opinion that this damage could not have been produced by any malfunction of the Therac-25 or by any operator error… there have apparently been no other instances of similar damage to this or other patients."

In early 1986 there were two accidents at the East Texas Cancer Center in Tyler, Texas, both of which resulted in the death of the patient within a few months. On March 21, 1986 the first massive radiation overdose occurred, though the extent of the overdose was not realized at the time. The Therac-25 was shut down for testing the day after the accident. Two AECL engineers, one from the plant in Canada, spent a day running machine tests but could not reproduce the malfunction code observed by the operator at the time of the accident. The home office engineer
explained that it was not possible for the Therac-25 to overdose a patient. The hospital physicist, who supervised the use of the machine, asked AECL if there were any other reports of radiation overexposure. The AECL quality assurance manager told him that AECL knew of no accidents involving the Therac-25.

On April 11, 1986 the same technician received the same malfunction code when an overdose occurred. Three weeks later the patient died; an autopsy showed acute high-dose radiation injury to the right temporal lobe of the brain and to the brain stem. The hospital physicist was able to reproduce the steps the operator had performed and measured the high radiation dosage delivered. He determined that data-entry speed during editing of the treatment script was the key factor in producing the malfunction code and the overdose. Examination of the portion of the code responsible for the Tyler accidents showed major software design flaws. Levinson and Turner (1993) describe in detail how the race condition occurred in the absence of the hardware interlocks and caused the overdose. The first report of the Tyler accidents came to the FDA from the Texas Health Department. Shortly thereafter, AECL provided a medical device accident report to the FDA discussing the radiation overdoses in Tyler.

On May 2, 1986 the FDA declared the Therac-25 defective and required the notification of all customers. AECL was required to submit to the FDA a corrective action plan for correcting the causes of the radiation overdoses. After multiple iterations of a plan to satisfy the FDA, the final corrective action plan was accepted by the FDA in the summer of 1987. The action plan resulted in the distribution of software updates and hardware upgrades that reinstated most of the hardware interlocks that were part of the Therac-20 design.

AECL settled the Therac-25 lawsuits filed by patients that were injured and by the families of patients who died from the radiation overdoses. The total compensation has been estimated to be over $150 million.

Summary

Leveson and Turner (1993) describe the contributing factors to Therac-25 accidents: “We must approach the problems of accidents in complex systems from a systems-engineering point of view and consider all contributing factors.” For the Therac-25 accidents, the contributing factors included

- management inadequacies and a lack of procedures for following through on all reported incidents;
- overconfidence in the software and the resulting removal of hardware interlocks (causing the software to be a single point of failure that could lead to an accident);
- less than acceptable software engineering practices; and
- unrealistic risk assessments along with over confidence in the results of those assessments.

Recent Medical Radiation Experience


The following quotations are excerpts from that series:

Increasingly complex, computer-controlled devices are fundamentally changing medical radiation, delivering higher doses in less time with greater precision than ever before.” But patients often know little about the harm that can result when safety rules are violated and ever more powerful and technologically complex machines go avery. To better understand those risks, The New York Times examined thousands of pages of public and private records and interviewed physicians, medical physicists, researchers and government regulators. The Times found that while this new technology allows doctors to more accurately attack tumors and reduce certain mistakes, its complexity has created new avenues for error — through software flaws, faulty programming, poor safety procedures or inadequate staffing and training. . . .

Linear accelerators and treatment planning are enormously more complex than 20 years ago,’ said Dr. Howard I. Amols, chief of clinical physics at Memorial Sloan-Kettering Cancer Center in New York. But
hospitals, he said, are often too trusting of the new computer systems and software, relying on them as if they had been tested over time, when in fact they have not.

Hospitals complain that manufacturers sometimes release new equipment with software that is poorly designed, contains glitches or lacks fail-safe features, records show. Northwest Medical Physics Equipment in Everett, Wash., had to release seven software patches to fix its image-guided radiation treatments, according to a December 2007 warning letter from the F.D.A. Hospitals reported that the company's flawed software caused several cancer patients to receive incorrect treatment, government records show.

References

Works Cited


Primary References

None.

Additional References

None.
FBI Virtual Case File System Case Study

This case study presents systems and software engineering issues encountered in the Federal Bureau of Investigation (FBI) Virtual Case File (VCF) project in the period between 2000-2005. VCF development was abandoned in 2005 after over $170 million had been spent.

Domain Background

The FBI is an organization within the United States Department of Justice (DoJ) consisting of 23 divisions, including counterintelligence, criminal investigation, and cyber crime. The Bureau's 12,400 agents investigate everything from counter-terrorism leads to kidnappings. They interview witnesses, develop informants, conduct surveillance, hunt for clues, and collaborate with local law enforcement to find and arrest criminals. Agents document every step and methodically build case files. They spend a tremendous amount of time processing paperwork. This system of forms and approvals stretches back to the 1920s when forms for all of the bureau's investigative reports were standardized.

In 2000, the Bureau had hundreds of standardized paper forms and obsolete information technology (IT) systems. The FBI's 13,000 computers could not run modern software. Most of the agency offices were connected to the FBI Intranet with links operating at about the speed of a 56 kilobits-per-second modem. Agents could not e-mail U.S. Attorneys, federal agencies, local law enforcement, or each other; instead, they typically sent case-related information by fax. The agency's problems in 2000 were summarized in the 9/11 Commission Report: "the FBI's information systems were woefully inadequate. The FBI lacked the ability to know what it knew; there was no effective mechanism for capturing or sharing its institutional knowledge" (National Commission on Terrorist Acts upon the United States 2004).

In September 2000, Congress approved $380 million over three years for what was then called the FBI Information Technology Upgrade Program. Eventually divided into three parts, the program became known as the Trilogy Information Technology Modernization Program. The first part would provide all 56 FBI field offices with updated computer terminals, as well as new hardware such as scanners, printers, and servers. The second part would re-implement the FBI Intranet to provide secure local area and wide area networks, allowing agents to share information with their supervisors and each other. The third part was intended to replace the FBI's investigative software applications, including the obsolete Automated Case Support (ACS) system.

In June 2001, the FBI awarded a contract to develop the investigative software applications of Trilogy to Science Applications International Corporation (SAIC) over a three year period. The purpose of the software to be developed was to

- provide the capability to find information in FBI databases without having prior knowledge of its location, and to search all FBI databases with a single query through the use of search engines;
- Web-enable the existing investigative applications;
- improve capabilities to share information inside and outside the FBI;
- provide access to authorized information from both internal and external databases; and
- allow the evaluation of cases and crime patterns through the use of commercial and FBI-enhanced analytical and case management tools.

After the September 11 terrorist attacks, the inability of FBI agents to share the most basic information about al Qaeda's U.S. activities was front-page news. Within days, the FBI's obsolete technology infrastructure was being discussed in Congress and the FBI was under intense pressure to improve its information sharing capabilities. On September 4, 2001, Robert S. Mueller III became FBI director, and, in the face of intense public and congressional pressure, Mueller accelerated the Trilogy program. The planned three year period to develop the investigative software was considered politically unacceptable. In January 2002, the FBI requested an additional $70 million to accelerate Trilogy; Congress went further, approving $78 million.
Providing web-enablement of the existing but antiquated and limited ACS system would not provide the investigative case management capabilities required to meet the FBI’s post-September 11 mission. In December 2001, the FBI asked SAIC to stop building a Web-based front end for the old programs. Instead, SAIC was asked to devise a new case management system, the Virtual Case File (VCF), to replace ACS. The VCF would contain a major new application, database, and graphical user interface. In order to make both criminal and terrorist investigation information readily accessible throughout the FBI, major changes to the standardized FBI processes would be required. This case study focuses on the VCF component of the Trilogy program.

Case Study Background

The most complete description of the development of the VCF is the report by the DoJ Office of the Inspector General (OIG). The OIG reports to the Attorney General and is independent of the FBI organizations responsible for the Trilogy program. The introduction to the report states, “We conducted this audit to assess the FBI’s progress in meeting cost, schedule, technical, and performance targets for the three components of Trilogy. We also examined the extent to which Trilogy will meet the FBI’s current and longer-term IT needs” (OIG 2004).

An IEEE Spectrum article complements the OIG audit report by providing detailing the development of the VCF requirements, the contractor’s activities, and the project management failures by both the FBI and the contractor. The contractor’s viewpoint is presented in testimony given before a subcommittee of the U.S. Senate Appropriations Committee.

These materials, in total, provide a comprehensive view of the VCF program and the reasons for its failure.

Case Study Description

In the political environment following the 9/11 attacks, funding for the VCF project was never a problem. By early 2002, SAIC and the FBI committed to creating an entirely new case management system in 22 months. High-level funding enabled the project to continue gaining momentum in spite of the problems it encountered. The scheduling for the VCF project focused on what was desired, not what was possible. Trilogy’s scope grew by approximately 80% from the initial project baseline (Moore 2010).

The reasons for the failure of the VCF project are associated with the non-use or misuse of numerous system engineering practices, especially within stakeholder requirements, system requirements, planning, assessment and control, and risk management. Given the political pressures following the 9/11 attacks, the schedule was accelerated to the point that it was nearly impossible for the developers to follow an appropriate systems engineering process. The FBI cycled through five people in the role of Chief Information Officer in four years and most decisions were made by committees. In order to compress the schedule, the FBI even proposed replacing the ACS with the VCF over a weekend using an IT procedure called a “flash cut-over.” In this proposed implementation, the ACS system would be taken offline and entirely replaced by VCF. Once the cut-over happened, there would be no mechanism to return to ACS, even if the VCF did not work properly.

SAIC worked under a cost-plus-award-fee contract for the VCF as the scope of the project was undefined in early 2002 when work began. Given the schedule pressures, the FBI believed that there was no time to develop formal requirements (glossary), validate them with the various FBI user communities, and then estimate the cost and time required to develop the VCF. The SAIC contract did not require specific completion milestones and the cost-plus contract allowed the scope to increase. VCF was a case of not getting the requirements sufficiently defined in terms of completeness and correctness. The continuous redefinition of requirements had a cascading effect on what had already been designed and produced. Once there was demonstrable software, change requests started arriving—roughly 400 from December 2002 to December 2003.

The new FBI Intranet was specified during 2001, before the start of the VCF project and with little understanding of the network traffic that would result from information sharing. By early 2003, the FBI began to realize how taxing
the network traffic would be once all 22,000 users came online. The requirements for the FBI Intranet were modified based on the best guesses for the bandwidth that would be required when the VCF was fully operational. By early 2004, the new FBI Intranet was in operation, although the VCF software was far from complete.

In reaction to the time pressure, SAIC broke its VCF development group into eight teams working in parallel on different functional elements of the program. However, this posed many integration challenges and the eight threads would later prove too difficult for SAIC to combine into a single system. By the time VCF was canceled, SAIC had developed over 700,000 lines of software based upon an incomplete set of requirements that were documented in an 800-page volume.

**Summary**

The OIG summarizes its conclusions as

> Various reasons account for the delays and associated cost increases in the Trilogy project, including:
>  
> • poorly defined and slowly evolving design requirements,
>  
> • contracting weaknesses,
>  
> • IT investment management weaknesses,
>  
> • lack of an Enterprise Architecture,
>  
> • lack of management continuity and oversight,
>  
> • unrealistic scheduling of tasks,
>  
> • lack of adequate project integration, and
>  
> • inadequate resolution of issues raised in our previous reports on Trilogy . . . .

According to the Government Accountability Office (GAO), an Enterprise Architecture is a set of descriptive models such as diagrams and tables that define, in business and technology terms, how an organization operates today, how it intends to operate in the future, and how it intends to invest in technology to transition from today’s operational environment to tomorrow’s. . . .

As of early 2005 the FBI’s operations remain significantly hampered due to the poor functionality and lack of information-sharing capabilities of its current IT systems. . . . (OIG 2005)

In May 2005, FBI director Mueller announced Sentinel, a four-phase, four-year project intended to fulfill the purpose of VCF and provide the Bureau with a web-based case and records management system. During the previous five years, commercial case management software had become available; as a result, Sentinel is intended to utilize commercial off-the-shelf (COTS) software. A report by the OIG in late 2009 describes Sentinel and its status at that time. Sentinel was put on line for all employees on July 1, 2012, and it ended up at $451 million and 2 1/2 years overdue (Yost 2012).

**References**

**Works Cited**


FBI Virtual Case File System Case Study


**Primary References**

None.

**Additional References**


SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [{1}](#).

ENCODED_CONTENT

NDQ2MDYPGRpdiBDiZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHg4dHlwZT0idGV4dC9qYXZhc2Ny END_ENCODED_CONTENT
MSTI Case Study

The Miniature Seeker Technology Integration (MSTI) spacecraft was the first of its kind: a rapid development spacecraft, designed and launched in one year. As an aerospace example for a satellite application, the case study, "M.S.T.I.: Optimizing the Whole System" (Grenville, Kleiner, and Newcomb 2004), describes the project's systems engineering approach. Driven by an aggressive schedule, MSTI optimized over the whole project, rather than allowing sub-optimizations at the component level. As a partnership with Phillips Laboratories, the Jet Propulsion Laboratory (JPL), and Spectrum Astro, MSTI went into orbit on November 21, 1992. The MSTI-1 succeeded in meeting all primary mission objectives, surpassing the 6-day data collection mission requirement.

Domain Background

There are many case study examples for aerospace systems. This case is of particular interest because it highlights mechanisms which enabled successful performance following an aggressive schedule. Since this rapid development spacecraft was designed and launched in one year, new ways of structuring the project were necessary. Within this domain, the MSTI project used an innovative approach. Practices from this project led to the Mission Design Center and the System Test Bed at JPL.

Case Study Background

This case study was developed in support of the National Aeronautics and Space Administration (NASA) Program and Project Management Initiative by authors at the Virginia Polytechnic Institute and State University and Scientific Management, Inc. The case study was developed in the interest of continuously improving program and project management at NASA (NASA 2010). Research for this case included comprehensive literature review and detailed interviews. The project was selected based on the potential for providing lessons learned.

Case Study Description

The MSTI case study illustrates many principles described in the Systems Engineering Body of Knowledge (SEBoK). The MSTI team had to make adjustments to the traditional approach to spacecraft development in order to stay within budget and to meet the aggressive timeline of bringing a spacecraft from conception to launch within one year. The team realized that they were "building Porsches not Formula 1s" (Grenville, Kleiner, Newcomb 2004). Meeting the schedule was a crucial factor that affected all decisions. The SEBoK knowledge area on life cycle models describes life cycle design in more detail.

The team took advantage of existing hardware architectures for their architectural design to expedite the project. In addition, at each design phase, the whole system was optimized instead of optimizing subsystems, and the level of optimization at the subsystem level was reduced. A hardware-in-the-loop test bed was used throughout the project, which expedited system integration.

The schedule was maintained only at a high level in the project management office, and the costs were managed using a cost reporting technique for "cost to completion." Rather than report on past spending, the Responsible Engineering Authorities (REAs) were expected to continually evaluate their ability to complete their tasks within projected costs. Faster procurement was achieved using the Hardware Acquisition Team, where a technical team member was matched with a procurement representative for each design function. This pair wrote the specifications together and initiated the purchase requisitions.

From the organizational perspective, increased responsibility and accountability were given to each team member. Individuals took ownership of their work and the decision process was streamlined. The team made more "good decisions," rather than optimal decisions. The team was collocated, and daily meetings were used to assign daily
tasks and keep the team focused on the launch. The standard Problem Failure Report (PFR) was streamlined and electronic reports provided snapshots of the resolved and outstanding PFRs. The report helped REAs stay on top of potential problem areas. REAs were responsible for looking forward on the project horizon and notifying the team of any potential problem areas.

The first satellite in the MSTI series, MSTI-1, was launched on November 21, 1992. The spacecraft weighed 150 kg and was built for $19M in less than 12 months. Over 200,000 photographs were returned from the spacecraft. From a project management standpoint, all mission objectives were completed.

In addition, MSTI had a lasting legacy. Faster procurement developed into an approach JPL now calls "Fast Track Procurement." Hardware acquisition teams are used often in JPL projects. The hardware-in-the-loop test bed was the precursor to the Flight System Test Bed at JPL. Team members moved up quickly in JPL due to the increased responsibility and authority they were given on the MSTI project.

Summary

MSTI demonstrated that an aggressive schedule can be used to design low earth-orbiting spacecraft to optimize the full system. The MSTI experience changed JPL’s culture and their approach to spacecraft development and mission management. The insights from this case study example can help both students and practitioners better understand principles described in the SEBoK.

References

Works Cited


Primary References

None.

Additional References

None.
Next Generation Medical Infusion Pump Case Study

This case study summarizes the systems engineering aspects of the next-generation Symbiq™ IV (intravenous) medical pump development. Symbiq™ was developed by Hospira Inc. and documented in detail in Chapter 5 of the National Research Council book, *Human-System Integration in the System Development Process*. As described in the book, Symbiq™’s purpose was “to deliver liquid medications, nutrients, blood and other solutions at programmed flow rates, volumes and time intervals via intravenous and other routes to a patient, primarily for hospital use with secondary limited feature use by patients at home” (Pew 2007).

Domain Background

This case study provides insight into the use of systems engineering practices in a medical application.

Case Study Background

The project that is the subject of this report was approved by the Governing Board of the National Research Council, whose members are drawn from the councils of the National Academy of Sciences, the National Academy of Engineering, and the Institute of Medicine.

The study was supported by Award Nos. W911NF-05-0150 and FA5650-06-1-6610 between the National Academy of Sciences, the U.S. Department of the Army, and the U.S. Department of the Air Force.

Case Study Description

In creating a next-generation product, Hospira proposed to introduce new IV pump features, such as:

- multi-channel vs. single-channel liquid delivery;
- the ability to group multi-channeled devices together;
- associated user-programming capabilities and programmable drug libraries for specifying parallel delivery of liquids;
- use of color touchscreen devices;
- integration with numerous types of hospital information systems;
- ease of use for both medical personnel and patients at home;
- handling of potential hardware, software, and human-user faults;
- compliance with U.S. and international safety standards;
- use of alternating-current or battery power; and
- the ability to be cost-competitive and attractive to traditional medical and hospital administration personnel.

Many of these features are highly coupled, such as the multi-channel hardware controls, concurrent software synchronization, distinctive displays and alarms for multi-channel devices, and rigorous medical safety standards.
Views of the resulting medical infusion pump can be found as Figures 5-5 and 5-6 in Chapter 5, page 107 of the Pew and Mavor (2007) book. Systems engineering for the device involved a great deal of concurrent analysis and engineering of its hardware, software, human factors, operational, business, and safety aspects. It has been a commercial success and won the 2006 Human Factors and Ergonomics Society’s User-Centered Product Design Award and the 2007 Medical Design Excellence Award.

Not only were there numerous technical challenges in the development of Symbiq™, but there were also challenges in the systems engineering of a product with a life-cycle operational concept that would produce satisfactory outcomes for a wide variety of product and operational stakeholders whose value propositions were often in some conflict. Some stakeholders wanted numerous features that would require a complex user interface, while others wanted a simple and easy to learn interface. Some users wanted the most advanced color touchscreen displays available, while others wanted a simpler, less-expensive product that was harder to misuse due to inadvertent screen commands. Some organizations felt that a minimal interpretation of the required safety features would be acceptable, while others advocated ultrahigh assurance levels. Some marketing personnel wanted a quick development and fielding of the basic product to capture market share, while maintainers wanted initial built-in life cycle support, maintenance, and diagnostic capabilities.

In such situations, many organizations focus on making quick requirement decisions and rapidly proceed into development. However, Hospira's understanding of the uncertainties and risks caused them to pursue a risk-driven, incremental commitment course of buying information to reduce risk, as emphasized in the SEBoK Part 3 knowledge area on Risk Management. As described in Pew and Mavor (2007), Hospira used a version of the Incremental Commitment Spiral Model (ICSM) summarized in the SEBoK Part 3 Knowledge Area on representative systems engineering process models. The following sections describe the project’s incremental system definition progress through the ICSM exploration, valuation, foundations, and Development phases. Some evolution of terminology has occurred, the Pew and Mavor (2007) version uses ICM instead of ICSM and "architecting phase" instead of "foundations phase".

**Symbiq™ Exploration Phase Summary**

In the exploration phase, the project carried out numerous analyses on stakeholder needs, technical opportunities, and business competition. Using these analyses, the project team determined ranges of preferred options. Stakeholder needs analyses included contextual inquiry via shadowing of nurses using IV pumps and followup interviews, as well as creating task flow diagrams, use environment analyses, and user profiles analyses. Technical opportunity analyses included initial conceptual designs of multi-channel pump configurations, evaluation of commercially available single-color and multicolor display devices with touchscreen capabilities, and software approaches for specifying multi-channel delivery options and synchronizing concurrent processes.

Business competition analyses included hiring a management and marketing planning firm to examin next-generation pump competitor strengths and weaknesses with respect to such capabilities as the number of pump channels, therapies, programming options, air-in-line management, battery and alternating current capabilities, biomedical domain expertise, and alarms. Several key competitive advantages of a next-generation pump were identified, such as the ability to read bar-codes, small size, light weight, stand-alone functional channels, an extensive drug library, a high level of reliability, and clear mapping of screen displays and pumping channels.

Market research and market segment analyses also identified market windows, pricing alternatives, hospital purchasing decision-making trends, and safety aspects. These were iterated by focus groups of key thought leaders in critical care. The results were factored into a product concept plan, cost analysis, and business case analysis. These were independently reviewed by experts as part of the ICSM Valuation Phase Commitment Review process, which resulted in a go-ahead decision with an identification of several risks to be managed.
**Symbiq™ Valuation Phase Summary**

The valuation phase focused on the major risks highlighted in the Valuation Commitment Review, such as the multi-channel pump options, the types of programmable therapies, the need for tailorable medication libraries, the display screen and user interface options, and the safety considerations. The valuation phase also elaborated the product concept plan for the most attractive general set of options, including a development plan and operations plan, along with an associated cost analysis, risk analysis, and business case for review at the Foundations Commitment Review.

The multi-channel pump options were explored via several hardware industrial design mockups and early usability tests of the mockups. These included evaluation of such desired capabilities as semi-automatic cassette loading, special pole-mounting hardware, stacking of and total number of channels, and tubing management features. The evaluations led to the overall all choice to use a semi-automatic cassette loading capability with a red-yellow-green LED display to indicate concerns with the loading mechanism and with the pump in general.

Field exercises with prototypes of the pole mountings indicated the need for quick release and activation mechanisms, which were subsequently implemented. Risk analyses of alternative stacking mechanisms and the potential number of channels available established a preference for side-by-side stacking, a decision to develop one and two channel units, and to support a maximum of four channels in a stacked configuration.

The types of programmable therapies considered included continuous delivery for a specified time period, patient weight-based dosing, piggyback or alternating delivery between the two channels, tapered or ramped-rate delivery, intermittent-interval delivery, variable-time delivery, and multistep delivery. These were evaluated via prototyping of the software on a simulated version of the pump complexes and were iterated until satisfactory versions were found.

Evaluation of the tailorable medication libraries addressed the issue that different hard and soft safety limits were needed for dosages in different care settings (e.g., emergency room, intensive care, oncology, pediatric care, etc.) which creates a need for hospitals to program their own soft limits (overridable by nurses with permission codes) and hard limits (no overrides permitted). Stakeholder satisfaction with the tailoring features was achieved via prototype exercises and iteration with representative hospital personnel.

A literature review was conducted to determine the relative advantages and disadvantages of leading input and display technologies, including cost and reliability data. After down-selecting to three leading vendors of touch screen color LCD displays and further investigating their costs and capabilities, a business risk analysis focused on the trade-offs between larger displays and customer interest in small-footprint IV pumps. The larger display was selected based on better readability features and the reduced risk of accidental user entries since the larger screen buttons would help to avoid these occurrences. Extensive usability prototyping was done with hardware mockups and embedded software that delivered simulated animated graphic user interface (GUI) displays to a touchscreen interface that was integrated into the hardware case.

The safety risk analysis in the valuation phase followed ISO 14971:2000 standards for medical device design, focusing on Failure Modes and Effects Analyses (FMEAs). This analysis was based on the early high-level design, such as entry of excessive drug doses or misuse of soft safety limit overrides. Subsequent-phase FMEAs would elaborate this analysis, based on the more detailed designs and implementations.

As in the exploration phase, the results of the valuation phase analyses, plans, budgets for the succeeding phases, the resulting revised business case, evidence of solution feasibility, and remaining risks with their risk management plans were reviewed by independent experts and the ICSM Foundations Commitment Review was passed, subject to a few risk level and risk management adjustments.
Symbiq™ Foundations Phase Summary

During the foundations phase, considerable effort was focused on addressing the identified risks such as the need for prototyping the full range of GUI usage by the full range of targeted users, including doctors, home patients, the need for interoperability of the Symbiq™ software with the wide variety of available hospital information systems, and the need for fully detailed FMEAs and other safety analyses. Comparable added effort went into detailed planning for development, production, operations, and support, providing more accurate inputs for business case analyses.

GUI prototyping was done with a set of usability objectives, such as

- 90% of experienced nurses will be able to insert the cassette the first time while receiving minimal training;
- 99% will be able to correct any insertion errors;
- 90% of first time users with no training will be able to power the pump off when directed; and
- 80% of patient users would rate the overall ease of use of the IV pump three or higher on a five-point scale (with five being the easiest to use).

Similar extensive evaluations were done on the efficacy and acceptability of the audio alarms, including the use of a patient and intensive care unit simulator that included other medical devices that produced noises, as well as other distractions such as ringing telephones. These evaluations were used to enable adjustment of the alarms and to make the visual displays easier to understand.

Software interoperability risk management involved extensive testing of representative interaction scenarios between the Symbiq™ software and a representative set of hospital information systems. These resulted in several adjustments to the software interoperability architecture. Also, as the product was being developed as a platform for the next generation of infusion pump products, the software design was analyzed for overspecialization to the initial product, resulting in several revisions. Similar analyses and revisions were performed for the hardware design.

As the design was refined into complete build-to specifications for the hardware and the operational software, the safety analyses were elaborated into complete FMEAs of the detailed designs. These picked up several potential safety issues, particularly involving the off-nominal usage scenarios, but overall confirmed a high assurance level for the safety of the product design. However, the safety risk assessment recommended a risk management plan for the development phase to include continued FMEAs, thorough off-nominal testing of the developing product’s hardware and software, and extensive beta-testing of the product at representative hospitals prior to a full release.

This plan and the other development and operations phase plans, product feasibility evidence, and business case analysis updates were reviewed at a Development Commitment Review, which resulted in a commitment to proceed into the development phase.

Symbiq™ Development Phase Systems Engineering Summary

The development phase was primarily concerned with building and testing the hardware and software to the build-to specifications, but continued to have an active systems engineering function to support change management; operations, production, and support planning and preparation; and further safety assurance activities as recommended in the risk management plan for the phase.

For hospital beta-testing, thoroughly bench-tested and working beta versions of the IV pump were deployed in two hospital settings. The hospitals programmed drug libraries for at least two clinical care areas. The devices were used for about four weeks. Surveys and interviews were conducted with the users to capture their “real world” experiences with the pump. Data from the pump usage and interaction memory was also analyzed and compared to the original doctors’ orders. The beta tests revealed a number of opportunities to make improvements, including revision of the more annoying alarm melodies and the data entry methods for entering units of medication delivery time in hours or minutes.
Usability testing was also conducted on one of the sets of abbreviated instructions called TIPS cards. These cards serve as reminders for how to complete the most critical tasks. Numerous suggestions for improvement in the TIPS cards themselves, as well as the user interface, came from this work, including how to reset the “Air-in-Line” alarm and how to check all on-screen help text for accuracy.

The above mentioned usability objectives were used as Acceptance Criteria (glossary) for the validation usability tests. These objectives were met. For example, the calculated task completion accuracy was 99.66% for all tasks for first time nurse users with minimal training. There were a few minor usability problems uncovered that were subsequently fixed without major changes to the GUI or effects on critical safety related tasks.

The risk analysis was iterated and revised as the product development matured. FMEAs were updated for safety critical risks associated with three product areas: the user interface, the mechanical and electrical subsystems, and the product manufacturing process. Some detailed implementation problems were found and fixed, but overall the risk of continuing into full-scale production, operations, and support was minimal. Systems engineering continued into the operations phase, primarily to address customer change requests and problem reports, and to participate in planning for a broader product line of IV pumps.

Overall, customer satisfaction, sales, and profits from the Symbiq™ IV pump have been strong and satisfaction levels from the management, financial, customer, end user, developer, maintainer, regulatory, and medical-community stakeholders have been quite high (Pew 2007).

Summary

In summary, the Symbiq™ Medical Infusion Pump Case Study provides an example of the use of the systems engineering practices discussed in the SEBoK. As appropriate for a next-generation, advanced technology product, it has a strong focus on risk management, but also illustrates the principles of synthesis, holism, dynamic behavior, adaptiveness, systems approach, progressive entropy reduction, and progressive stakeholder satisfying discussed in Part 2 of the SEBoK. It provides an example of an evolutionary and concurrent systems engineering process, such as the incremental commitment spiral process, and of other knowledge areas discussed in SEBoK Parts 3 and 4, such as system definition, system realization, system engineering management, and specialty engineering.

References

Works Cited


Primary References

None.

Additional References

None.
**Design for Maintainability**

This article describes an example of where systems thinking led to a much more practical solution to a common problem. For additional information, refer to Systems Thinking.


**Background**

In the late 1870s a Parisian obstetrician named Stephane Tarnier was visiting the Paris Zoo where they had farm animals. While there he conceived the idea of adapting a chicken incubator to use for human newborns, and he hired "the zoo’s poultry raiser to construct a device that would perform a similar function for human newborns." At the time infant mortality was staggeringly high "even in a city as sophisticated as Paris. One in five babies died before learning to crawl, and the odds were far worse for premature babies born with low birth weights." Tarnier installed his incubator for newborns at Maternité de Paris, and embarked on a quick study of five hundred babies. "The results shocked the Parisian medical establishment: while 66 percent of low-weight babies died within weeks of birth, only 38 percent died if they were housed in Tarnier’s incubating box. … Tarnier’s statistical analysis gave newborn incubation the push that it needed: within a few years the Paris municipal board required that incubators be installed in all the city’s maternity hospitals." …

**Purpose**

"Modern incubators, supplemented with high-oxygen therapy and other advances, became standard equipment in all American hospitals after the end on World War II, triggering a spectacular 75 percent decline in infant mortality rates between 1950 and 1998."… "In the developing world, however, the infant mortality story remains bleak. Whereas infant deaths are below ten per thousand births throughout Europe and the United States, over a hundred infants die per thousand (births) in countries like Liberia and Ethiopia, many of them premature babies that would have survived with access to incubators."
Challenges
But modern incubators are complex, expensive things. A standard incubator in an American hospital might cost more than $40,000 [about €30,000]. But the expense is arguably the smaller hurdle to overcome. Complex equipment breaks and when it breaks you need the technical expertise to fix it, and you need replacement parts. In the year that followed the 2004 Indian Ocean tsunami, the Indonesian city of Meulaboh received eight incubators from a range on international relief organizations. By late 2008, when an MIT professor named Timothy Prestero visited the hospital, all eight were out of order, the victims of power surges and tropical humidity, along with the hospital staff’s inability to read the English repair manual. The Meulaboh incubators were a representative sample: some studies suggest that as much as 95 percent of medical technology donated to developing countries breaks within the first five years of use.

Systems Engineering Practices
“Prestero had a vested interest in those broken incubators, because the organization he founded, Design that Matters, had been working for several years on a scheme for a more reliable, and less expensive, incubator, one that recognized complex medical technology was likely to have a very different tenure in a developing world context than it would in an American or European hospital. Designing an incubator for a developing country wasn’t just a matter of creating something that worked; it was also a matter of designing something that would break in a non-catastrophic way. You couldn’t guarantee a steady supply of spare parts, or trained repair technicians. So instead, Prestero and his team decided to build an incubator out of parts that were already abundant in the developing world. The idea had originated with a Boston doctor named Jonathan Rosen, who had observed that even the smaller towns of the developing world seemed to be able to keep automobiles in working order. The towns might lack air conditioning and laptops and cable television, but they managed to keep their Toyota 4Runners on the road. So Rosen approached Prestero with an idea: What if you made an incubator out of automobile parts?”

Lessons Learned
“Three years after Rosen suggested the idea, the Design that Matters team introduced a prototype device called NeoNurture. From the outside, it looked like a streamlined modern incubator, but its guts were automotive. Sealed-beam headlights supplied the crucial warmth; dashboard fans provided filtered air circulation; door chimes sounded alarms. You could power the device via an adapted cigarette lighter, or a standard-issue motorcycle battery. Building the NeoNurture out of car parts was doubly efficient, because it tapped both the local supply of parts themselves and the local knowledge of automobile repair. These were both abundant resources in the developing world context, as Rosen liked to say. You didn’t have to be a trained medical technician to fix the NeoNurture; you didn’t even have to read the manual. You just needed to know how to replace a broken headlight.”
References

Works Cited

Primary References

Additional References
None.

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].

ENCODED_CONTENT
OTk1MjYPGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2Nya3lubG9nIGNvbW1lbnRzIHBvd2VyZWQgYnkgPHNwYW4gY2xhc3M9ImxvZ28tZGlzcXVzIj5EaXNxdXM8L3NwYW4+PC9hPg==
END_ENCODED_CONTENT
Complex Adaptive Operating System Case Study

This article is based on the Complex Adaptive Operating System: Creating Methods for Complex Project Management case study (Findlay and Straus, 2015). The case study focuses on creating tools and methods that project managers can use in managing complex adaptive systems projects.

Background

The International Centre for Complex Project Management (ICCPM) is a non-profit organization that was created to address "the international community's ability to successfully deliver very complex projects and manage complexity across all industry and government sectors" (ICCPM, 2012).

In an ongoing effort to help member organizations successfully undertake major complex projects, ICCPM conducted a bi-annual series of international round-tables. The purpose of the round-tables was to better understand what contributes to the success of complex projects and to identify and develop new and improved tools and approaches. The round-tables were facilitated using a computer-assisted collaborated meeting process that leverages the features of complex adaptive systems—described below—to help people with diverse viewpoints and experience create new collective understanding and plans for action.

Complex major projects are known for being unsuccessful in on-time and on-budget completion. An (IBM, 2008) survey of 1,500 change managers found that only 40% of projects finished on time and on budget. Barriers to success were the inability to change attitudes or mindsets (58%), dysfunctional culture (40%) lack of senior management support (32%) and underestimating the complexity of a project (35%).

However, several new systemic approaches show considerable promise as a way to think about and manage projects. Six frameworks help inform these approaches: systems thinking, the features of complex adaptive systems (CASs), complexity theory, the Complexity Model of Change (Findlay and Straus, 2011); polarity thinking as a way of thinking about and leveraging wicked problems, cognitive complexity and adult development theory.

Systems thinking recognizes whole systems and the interdependencies of their parts. A system may be defined as "a set of things, organisms, and people that, as a result of their interconnection, produce their own patterns of behavior over time" (Meadows, 2008, p.2). A system cannot be understood by focusing on its parts alone (Wheatley, 1999).

Leaders of complex projects would also be wise to consider three fundamental theorems of complexity theory, which apply to CASs and which are critical to project success. These are a robust model of the system, requisite variety and adopting solutions which act at an appropriate scale.

- The robust model axiom considers that "no one can effectively influence a system until they have a thorough understanding of its scope and the connections and interdependencies" (Conant and Ashby, 1970).
- The law of requisite variety contents that "complexity can only be dealt with by equal or greater complexity" (Ashby, 1956, p. 2). In other words, in order to deal effectively with the diversity of problems presented by a complex project, one must have a repertoire of responses which is (at least) as nuanced as the problems themselves (Requisite Variety, 2015).
The scale condition requires that those who wish to exercise leverage over a system must recognize that "highly complex situations can best be addressed by greater degrees of freedom at the local scale so that innovation and adaptability are maximized" (Bar-Yam, 2004).

A third framework, the Complexity Model of Change, is a model of socio-technological change, comprising a series of growth and decay curves or waves, which helps project managers better understand how to influence systems and design new ones, so the roles, methods, rules of interaction or engagement, technologies and relationships between people are better aligned with each other and the desired outcome. The overlapping waves represent large-scale eras, for example, the Industrial Age and the Information Age, which have at their core a metaphor, for example the machine and the computer. The current wave, the Knowledge Age, is based on a network metaphor. The wave we are now entering is the Wisdom Age, which began in 2010. Its core metaphor is the complex adaptive system and the main thrust of this period is the wise application of knowledge.

Another area that project managers may now address differently is that of wicked problems or paradoxes: problems that are ill-defined and recurrent, and which, when attempts are made to solve them with single optimal solutions, create another problem. Polarity thinking regards wicked problems as sets of interdependent values or ideas—like centralizing for efficiency and decentralizing for adaptability—that persist together over time and need each other for the success of the system. If we pay attention to one pole at the expense of the other we achieve sub-optimal results. When we manage polarities as a system, we realize the benefits of both poles and achieve high performance over time with a minimum of vacillation and the need for correction.

Other disciplines that are critical to project success are understanding and making best use of new ways of thinking about issues and relating to others in more flexible and adaptive ways. Theories of cognitive complexity and adult development theory can contribute to how we think about this problem. For example, "triple-loop learning" (Gragert, 2013), helps us think about issues from a higher level of cognitive complexity. Instead of asking are we doing things right (single loop), we might ask are we doing the right things (double loop) or how do we decide what is the most effective paradigm to use to influence and create benefit for the system (triple)?

**Purpose**

The purpose of the ICCPM roundtables was to help project managers develop new and better ways to lead complex major projects, by bringing together people from both the buy-side and the supply-side to share their knowledge and experience and to grow a network of practitioners, professionals, researchers, and educators able to deliver leading edge complex project management solutions to client organizations and partners around the world (Findlay and Straus, 2015, p. 489).

**Challenges**

There are many challenges to be addressed in the complex project management environments. The three top contenders are 1) developing new ways of thinking, acting, and interacting; 2) developing more robust models of the system by getting everyone in the room—the project management team and their stakeholders; and 3) steering projects through multiple disruptive socio-technological shifts using the feature of complex adaptive systems.

"People, their organizations, and their projects need to be capable of reorganizing into new forms, which are a better fit with the new context" (Findlay and Straus, 2015, p. 494).
Systems Engineering Practices

One of the tools Findlay and Straus use to deal with all three challenges in the context of group interaction, such as the ICCPM round-tables, is the Zing complex adaptive meeting process. The process is used to guide conversation in the room, to capture, simultaneously display ideas and to help participants integrate and make meaning from the ideas. The tool was used for the round-tables to help people work together in new ways, develop new and better models of the system together and to design and pilot new and better decision and learning methods.

The technology “provide[s] a container for a suitably representative sample of the people in the system to meet and conceptualize a robust model of the system and develop strategies for how to leverage the system” (Findlay and Straus, 2015, p. 492).

A “talk-type-read-review” (Findlay and Straus, 2015, p. 492) etiquette was employed to organize the session, which, in complexity theory terms, is a simple rule of interaction. Rich, open-ended questions guide the conversations, the ideas are read out aloud and the common themes or stand out ideas are recorded by the facilitators.

The open-ended questions are asked one at a time to explore all possibilities and reduce complexity. Although, round-table participants often held opposing views at the beginning, of the session, through a processes of continual, iterative feedback, they ultimately arrived at similar or complementary conclusions by the end of the round-table.

The process “automates”—or helps participants engage in—ways of interacting that incorporate a higher level of cognitive complexity than the participants might engage in individually, thus facilitating a shift in the group to a higher level of system performance.

Lessons Learned

The role of leaders of complex projects is to help their organization systems successfully deliver on time and on budget amidst constant change. Their mandate is to deliver amazing new solutions while making few or no mistakes—a challenging goal even in far less complex environments. In order to be successful, project leaders (and their teams) need new systems structures—new tools and methods—that reliably get better results. They need to have a robust and fresh understanding of the systems over which they preside and how they might influence them to greatest effect.

No longer can the complex project leader go off into a corner and design a project and then try to sell it to the community and political leaders, for example. Leaders now need to involve the whole system in the design of a project from its inception through to completion. They need to deal with wicked problems not by looking for the one best solution, but by integrating and leveraging competing ideas. This requires a shift in perspective: from attempting to “control” a complex project system as one might control a mechanical device, to understanding projects as highly complex and interconnected “living” systems that evolve over time. While we do not have “control” over our systems in the classical sense, we can exert influence very effectively, provided that we constantly update our understanding of what is going on and learn new ways to act and interact that are more likely to achieve our desired outcomes.

To achieve this, leaders need to develop the capacity to “anticipate the skills, leadership and coordination roles, technologies, methods, and processes that will be required to successfully surf the waves of change…” (Findlay and Straus, 2015, p. 501).

The 2012 ICCPM round-table series discussion paper (ICCPM, 2012) uses the example of a system undergoing transformation of many levels to illustrate the difficulty that complex project leaders face:

“The issue has been characterized as learning to fly a plane, while the plane is already in the air, and being re-assembled into another kind of transportation technology altogether. And, at the same time, the current passengers are disembarking and another group is boarding that demands a better quality of service or experience at lower cost than
This case study illustrates the need, in times of accelerating change, of "a real-time, systems-wide approach to the development of the methods and tools for managing complex projects" (Findlay and Straus, 2015, p. 500) so leaders can deal successfully and creatively with uncertainty and ambiguity.

References

Works Cited

Primary References

Additional References
None.
SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].

Complex Adaptive Project Management System Case Study

This article is based around a Russia space agency Project Management Systems case study (Rzevski and Skobelev, 2014). The case study focuses on the development of a Real-Time Complex Adaptive Project Management Systems capable of effectively managing multiple related projects collectively contributing to a defined Enterprise Value.

Background

The essence of every management system should be the same: the allocation of human, physical, financial and intellectual (knowledge) resources to demands (tasks) with the aim of increasing the specified Enterprise Value, where the Enterprise Value is a system of values such as profit, market share, business sustainability, quality of service to customers, quality of working conditions for employees, quality of life in the community, etc. The key difference between the management of a business and management of a project is that businesses are continuously evolving processes whilst projects have specified beginnings and ends. Standard challenges for project management are:

- Stringent budgets and deadlines
- High competition for limited resource availability such as up-to-date domain knowledge, skills and advanced productivity tools
- Functional organisation, which inevitably impedes interdepartmental cooperation
- Bureaucratic management, which is more concerned with lines of command and reporting than with the full use of project member’s initiative and creativity and which negatively affects their motivation
- Rigid project planning, which leads to a rapid divergence between the project plan and reality

Large enterprises commonly operate several projects concurrently. What is best for an individual project it is not always best for the enterprise and therefore it is necessary to implement coordination of concurrently run projects with the objective of significantly increasing Enterprise Value.

There are two new key problems, which the 21st century brought to us.

The first is the rapidly increasing complexity of the Internet-based global market, which creates frequent unpredictable disruptive events. This requires real-time adaptive project management, for which there is at present
no precedent.
The second is the replacement of capital with knowledge as the key business resource in the economy in which the wealth created by knowledge services is greater than wealth created by producers of goods. There is at present no management system capable of discovering, processing, storing and allocating knowledge to project tasks.

**Purpose**
The client for this case study was one of the key space technology organisations in Russia, their equivalent of NASA, which operates, at any time, many concurrent mission critical projects.
The purpose of the project management system was to enable the client to effectively manage several related projects (at least ten), collectively contributing to the specified Enterprise Value, with the following requirements.

- Each project may consist of up to 5,000 constituent tasks
- Project members may have different backgrounds and skills and may belong to diverse business cultures
- Project members must have an opportunity to contribute to decision making processes, which affects their domain of work (distributed decision making)
- Both project management and project members must have readily available and up-to-date information on, respectively, project and individual progress, productivity and achievements of goals
- The allocation of resources to tasks must consider 4 types of resources, namely, human, physical, financial and knowledge
- Availability of resources for and constituent tasks of each project may change with short notice and these changes must be rapidly incorporated into the system
- Projects will be subjected to frequent disruptive events such as non arrival of expected orders, arrival of unexpected orders, sudden and unforeseen emergence of external/internal competitors, cancellations, changes in task specifications, delays, failures, no-shows, etc.
- Rules and regulations governing projects are likely to change rather frequently and any change in rules and regulations must be incorporated immediately and easily into the relevant project management system
- Any discrepancy between project plans and reality in the field must be continuously monitored and rapidly detected and reported
- Projects may cooperate and/or compete for resources in order to increase specified Enterprise Value

A thorough analysis of the client’s requirements led to the conclusion that it is necessary to develop a real-time, complex adaptive project management system capable of cooperating and/or competing with other systems, with the overarching goal to continuously increase specified Enterprise Value.
The new system would replace a number of stand-alone, manual or semi-automated project management systems with inadequate monitoring of progress and productivity.

A thorough analysis of contemporary practices showed that such a transformation had never before been achieved. To the best of the team’s knowledge, there were no real-time project management systems in existence anywhere in the world.

**Challenges**
This particular undertaking had a number of challenges.
The most important challenge was the resistance to change by client’s managers. The new system with its progress and productivity monitoring capabilities threatened to expose inefficiencies and was not universally welcomed. Two approaches were planned to manage this challenge: the first was education of participants and the second, a proposal for a new payment structure which related salaries to meeting of targets, as reported by the new system.
The scale of the proposed network of systems was an even more important challenge, especially because all projects were mission critical. To manage this challenge the plan was made to adopt an evolutionary development strategy.
The first step was planned to be a fully engineered prototype with a limited functionality, which would be extended into the first project management system only after a complete acceptance by the client that the prototype was capable of delivering its limited functionality as specified. The network of project management systems would be grown step by step.

The multi-agent technology, which underpinned the system, was well understood by the team, and a methodology for managing complexity (Rzevski and Skobelev, 2014) of the task was in place.

**Systems Engineering Practices**

**Overview**

The complexity of client’s projects ruled out all conventional project management practices and systems. Instead, for every project, the team designed a complex adaptive project management system, based on multi-agent technology, capable of meeting client requirements.

The system consisted of the following major components (Rzevski and Skobelev, 2014):

1. Knowledge Base containing domain knowledge relevant to the client’s project management processes
2. Multi-agent Virtual World which models the Real World of projects and is capable of managing real-world complexity
3. Interfaces between the Virtual and Real Worlds, which enable the Virtual World to, in-effect, manage the Real World, with or without human intervention

**Knowledge Base**


Examples of attributes are, for Task: Content, Cost, Duration, Deadline and Preferences; and for Project Member: Organisational Unit, Competences, Profile, Schedule, Current Task, Salary, Achievements and Preferences.

A fragment of enterprise ontology is shown below.
Virtual World

Examples of agents that populate the Virtual World include:

- Task Agent, whose objective is to search for the best resources capable to meet its requirements
- Human Resource Agent, whose objective is to get the best possible task, which will keep the project participant fully occupied, provide opportunities for bonuses and/or enable the participant to learn new skills or get new experience
- Physical Resource Agent, whose objective is to maximise resource utilisation
- Project Agent, whose objective is to maximise Project Value
- Enterprise Agent, whose objective is to maximise Enterprise Value

All decisions are made through agent negotiations, as exemplified by the following process:

Task Agents send messages to Human Resource Agents with required competences inviting them to contribute to task fulfilments. Human Resource Agents that are available send their bids. Task Agents offer project participation to those Agents that sent the best bids. Bids are subject to negotiations between affected agents.

A new Task Agent is created whenever a new task is formulated or a previously allocated task is modified. The new Task Agent consults ontology to find out what are its objectives and how to achieve them, and proceeds to send messages to selected Human Resource Agents inviting them to bid for project participation. It is very likely that this invitation will result in re-scheduling, giving an opportunity to Human Resource Agents that were not fully satisfied with their previous allocations to improve their positions. Remuneration, including bonuses, if any, is calculated on the basis of project member participation and achieved results. Enterprise members may participate in several projects.

The allocation of physical, software and knowledge resources is done in an analogous manner. Advanced methods (Rzevski and Skobelev, 2014) have been employed to maximise effectiveness of agent negotiation, such as, virtual microeconomics, agent satisfaction, agent proactivity, enterprise agents, swarm cooperation, etc.

Decisions on allocation of resources to project tasks are made using multiple criteria, for example, decreases in completion time, increases in quality and reducing identified risks, as illustrated below.

Figure 2. Decisions on allocation of resources to tasks are made using multiple criteria. This material is reproduced with permission of G. Rzevski and publisher WitPress. All other rights are reserved by the copyright owner.
**Connecting Virtual and Real Worlds**

The project member dashboard is the key link between the system and the project member. The exchange of information that could be conducted using the dashboard include:

- Negotiations of task content, risks, deadlines and budgets
- Acceptance/rejection by the project member of offered project tasks
- Inputs by project members of unexpected disruptive events and comments during the project
- Reports by the system on the project member performance in carrying out accepted tasks

The system may decide to engage two or more available project members in competition with each other to secure an agreement on the acceptable task performance.

The other key link between the virtual and real worlds is the project managers dashboard, which displays detailed project performance data in the form of various diagrams and Gant charts and allows the manager to examine or modify decisions made autonomously by the system.

**Lessons Learned**

The first real-time complex adaptive project management system was commissioned and deployed by the client in 2014 achieving the following results:

- 10% to 15% increase in project member productivity;
- 3 to 4 times reduction in manpower required for project scheduling, monitoring and coordination;
- 2 to 3 times reduction of response time to unpredictable disruptive events;
- 15% to 30% increase in the number of projects completion on budget and in time;
- A significant increase in project member motivation;
- A possibility to increase the number of projects operating in parallel without increasing the number of employees.

**References**

**Works Cited**


**Primary References**


**Additional References**

None.
Complex Adaptive Taxi Service Scheduler Case Study

This article is based around a London Taxi Service Case Study (Rzevski and Skobelev, 2014). The case study focuses on the development of a Real-Time Complex Adaptive Scheduler for a London Taxi Service capable of managing the complexity of many hundreds of taxi journeys in an unpredictable and changing environment, while fitting into the goals and values of the Enterprise.

Background

When this project was initiated, our client, the largest and the best-known minicab (taxi) operator in London had a fleet of more than 2,000 vehicles, each with a Global Positioning System (GPS) navigation system. The fleet comprised a variety of vehicles, including minivans and Sport Utility Vehicles (SUVs), some with equipment to match special customer requirements. Typically approximately 700 drivers worked concurrently, competing with each other for customers.

The company had a modern Enterprise Resource Planning (ERP) system and a call centre with over 100 operators receiving orders concurrently. Some orders were received through the company website. A large team of skilled dispatchers allocated vehicles to customers.

Main characteristics of the taxi service were as follows:

- More than 13,000 orders per day
- Occasionally more than 1,500 orders per hour (1 order every 2.4 seconds)
- Unpredictable order arrival times and locations
- Various clients, e.g., personal, corporate, Very Important Persons (VIPs), a variety of discounted tariffs, special requirements suitable for the disabled, small children (child seats), transportation of pets, etc.
- Many freelance drivers who leased cars from the company and were allowed to start and finish their shifts at times that suited them, which may have differed from day to day
- Clients in central London were guaranteed pick up times within 15 minutes of order placement
- Fundamentally the company tried to find the best economic match of vehicle to every client
- However, dynamic exceptions to this basic requirement included: matching drivers going to and from home with passengers travelling in the same direction (to reduce drivers’ idle runs); and giving priority to drivers with less work during a particular day (to increase drivers' satisfaction with working conditions)
No pre-planned taxi schedule was viable because any of the following unpredictable "Disruptive Events" occurred every 2 to 10 seconds:

- Order arrival, change, or cancellation
- Changes in driver profile, status, or location
- Client no-show
- Vehicle failure
- Delays due to traffic congestion, or queues at airports, railway stations, etc.

**Purpose**

Rescheduling up to 700 independent entities, travelling in London under unpredictable conditions that change every few seconds represented an exceedingly complex task, which was not feasible to accomplish using any known mathematical method.

Manual scheduling, as practiced, could not handle the frequent disruptive events. Many perturbations, such as unexpected delays, had to be ignored by the human dispatchers.

Therefore the project’s objective was to provide effective, real-time, automated assistance to accommodate the disruptions that drove the scheduling. Thus, the project purpose became the development of a complex adaptive software system capable of managing the taxi operation complexity described above with the aim of substantially improving: (1) operational profitability; (2) customer service quality; and (3) driver working conditions.

The planned transformation was from a manual to semi-automated managed taxi operation that facilitated optional human dispatcher interactions with a complex adaptive system scheduler. A thorough analysis of contemporary practices showed that such a transformation has never been achieved before. To the best of the team knowledge, there were no real-time schedulers of taxi operations in existence anywhere in the world.

**Challenges**

The team undertaking the development of a new real-time scheduler for this client had vast experience of designing and implementing complex adaptive software, and therefore no particular challenges were anticipated. The multi-agent technology, which underpinned the system, was well understood by the team, and a methodology for managing complexity (Rzevski and Skobelev, 2014) of the task was in place.

**Systems Engineering Practices**

**Overview**

The complexity of the taxi service ruled out all conventional systems engineering practices. The real-time adaptive scheduler for the client’s taxi service was developed using multi-agent software technology.

The scheduler design consisted of the following major components (Rzevski and Skobelev, 2014):

1. Knowledge Base containing domain useful information relevant to the client’s taxi service
2. Multi-agent Virtual World which models the Real World of the taxi service and is capable of managing its complexity
3. Communication channels between the Virtual and Real Worlds which enable the Virtual World management of the Real World with or without human intervention.

The system was designed to behave as follows. In reaction to every disruptive event, Order Agents, assigned to every received order, and Driver Agents, assigned to every working driver, negotiate, through the exchange of messages, the most suitable Order-Driver match. Before starting negotiations these software agents consult the Knowledge Base for the current negotiation rules. Once the best possible match (under prevailing circumstances) is agreed, the
Complex Adaptive Taxi Service Scheduler Case Study

result is communicated to Drivers, who are free to accept or reject the task (Glaschenko et al. 2009). All this is depicted simply in the figure below.

After a successful prototype implementation, a basic version of the complex adaptive scheduler was developed as described below.

![Figure 1. Essence of the event driven, real-time, adaptive scheduling of a taxi service.](image)

This material is reproduced with permission of G. Rzevski and publisher WitPress. All other rights are reserved by the copyright owner.

**Knowledge Base**

The Knowledge Base consisted of: (1) Ontology, containing conceptual knowledge as a semantic network; and (2) Values, in standard databases.

The basic Ontology contained two Object Classes: Order and Driver. Order attributes were

- Location of pick-up and drop-off
- Pick-up: urgent or booked in advance (for a certain date and time)
- Type of service (standard car, minivan, VIP, etc.)
- Importance of service (a number from 0 to 100 depending upon the client)
- Special requirements (pet, child chair, etc.)

Driver attributes were

- Type of vehicle
- Capability to complete special jobs
- Driver experience (novice or experienced)
- Domicile of driver
- Current vehicle location (GPS coordinates)
- Driver status (unavailable, break, working, free, will be free in 5/10 minutes, home transit)

Factual data on Object Instances (Individual Orders and Drivers/Vehicles statuses) were stored in client’s databases, including Scenes (i.e., instantaneous models of the taxi service yielding every vehicle location and driver availability).

**Virtual World**

In the basic version of the scheduler, the allocation of taxis to customers was done by the negotiation between Order Agents, assigned to customers, and Driver Agents, assigned to taxi drivers. Order Agents were active: they compiled lists of available vehicles and initiated negotiations with Driver Agents. Driver Agents, in this first version of the system were designed to be only reactive: they only replied to requests from Order Agents and implemented the option selected by an Order Agent.

In the extended version, hereafter described, Order Agents and Driver Agents competed with each other or co-operated, depending on what was best for the whole enterprise. In this version, in addition to Order and Driver
Agents, some new types of agents, namely, External Events Agents, Regional Loading Agents, and Orders Allocation Agents were also used.

Agents were designed to use flexible decision-making criteria instead of direct priorities, which is valuable when there is a need to deal with different categories of clients. For example, if a VIP order arrived and there was only one driver that fully corresponded to the specified requirements and if that driver was already assigned to another job, the system would nevertheless allocate the VIP order to this driver and initiate re-scheduling of the previously agreed matches, if required.

The system first attempted to maximize company profit. Then, other criteria that are important for the business were considered, such as the service level and driver working conditions. For example, when choosing from two approximately equal options the system allocated the order to the driver who had not received orders for a longer time, thus ensuring relatively fair distribution of orders.

This virtual agent-based scheduling system was designed to work effectively with human dispatchers. In a situation where one dispatcher takes a new order and schedules a vehicle to come from north to south to pick up a client, and another dispatcher independently schedules another vehicle to go from south to north for another order, the virtual agents can spot this schedule anomaly and recommend dispatchers change their decisions to be more effective.

To enable improved performance, the taxi allocation system functioned in short cycles rather than as an immediate reaction to every event. Between the cycles the system collected the events and placed them in a queue. During each cycle, the events from the queue were processed, one by one, and appropriate agents, in turn, were given control by a designated human system dispatcher. Each event thus initiated a chain of negotiations among virtual agents. When all events were processed and the system dispatcher was satisfied that the best possible schedule was produced for that cycle, the schedule perturbation was implemented in the real world, and the system fell asleep (was idled) until a new event arrived causing the initiation of the next cycle.

To decrease the dimensions of the decision space, a pre-matching mechanism was used, which determined the suitability of Order-Driver matching. This mechanism cuts off unpromising options.

The Order-Driver pairs were evaluated before the final decision was made. An evaluation mark was given to each option and good options were remembered so that the evaluations did not need to be repeated later. The evaluation mark was determined using a multi-criteria model and calculated as a sum of all criteria values multiplied by their (variable) weights.

The following criteria were used for option evaluation: distance to the order, predicted delay of the pick-up, if any, preferences of the driver, driver experience, distance of the driver to overloaded area (to utilize drivers from outlying districts), service level conformity, importance and priority of the order, driver’s place in a queue (if he is waiting at an airport), driver’s home address (if he is looking for an order to or from home).

Scheduling workflow included the following steps:

1. New order arrives and joins the event queue
2. Possibility of order scheduling is checked
3. A software agent is assigned to the order
4. All drivers that can complete this order are included in pre-matching
5. Evaluation of all Order-Driver pairs is done according to agreed criteria
6. The Order Agent requests order completion costs from selected Driver Agents. This cost includes the cost of transferring the order from the previously allocated driver, if any
7. The Driver Agent receives the information on the reallocation costs by sending a request to its current Order Agent
8. If the revised decision is better than the previous one, it is applied
9. Step 6 continues for all candidate drivers, for whom the initial evaluation (without transfers) was better than the current evaluation
10. If no further changes occur during the cycle, the event processing is considered finished

In order to achieve the best possible solution, the system continued to search for improvements in previously agreed Order-Driver matches until the last moment when it had to issue the instruction to a driver to fulfil an order (commitment time). During this time interval the Driver was considered to be available for new allocations, but only if the new allocation improved specified performance indicators.

When required, Driver Agents attempted “to come to an agreement” with each other about proposed re-allocation of orders. Occasionally, compensation was offered to the Driver Agent who lost a good client in order to improve overall value of the business, and Driver Agent satisfaction, in particular. Very often the re-scheduling of allocated resources caused a wave of negotiations aimed at the resolution of conflicts between new and old orders. The length of the re-scheduling chain was limited only by the time required for a taxi to reach a customer in a busy city such as London, which normally was sufficient for several changes of the schedule.

To summarise, the system built a schedule and perpetually reviewed it, attempting to improve key performance indicators as long as the time for essential re-scheduling was still available.

The Commitment Time was dynamically calculated for each order taking into account the priority and service type of the order and some other parameters. The introduction of the dynamic Commitment Time resulted in the increase of the fleet effectiveness by reducing the average task completion time per driver.

An option was introduced for the system to distribute the fleet according to the order-flow forecasts. Having information about the current order-flow and distribution of orders in the past, the expected order-flow was extrapolated, enabling the system to generate short-term (30 minutes) forecasts, which were normally reasonably correct. Based on the forecast, the system sent text messages to unoccupied drivers with recommendations to stay in, or move to the region where an increased order flow was expected. This feature enabled an improved distribution of the fleet, reducing response times and idle miles and increasing the number of pick-ups.

In cases when forecasts envisaged a probability of a VIP order arrival at a significant distance from the point where drivers were advised to congregate, the system would recommend that a proximate driver to move closer to the likely order point, offering him/her a guaranteed next order in exchange for compliance. This was an important feature because there were usually enough proximate drivers to complete available orders in areas that were not overloaded, and productivity of work in overloaded areas determined the actual fleet effectiveness. The system was also designed with an option to temporary amend criteria for the allocation of orders to drivers (for example, to extend the area where drivers are allowed to search for orders) to enable drivers to reach critical locations without being intercepted by less important orders from nearby locations.

The forecasting functionality was supported by an agent-based dynamic data mining system, which was, in fact, another complex adaptive system cooperating with the complex adaptive scheduler.

In later versions the system was designed to detect and identify drivers that cheat, i.e., by deliberately providing the scheduler with false information to gain personal advantages. Recorded cases include attempts to

- Reduce their ultimate waiting time by reporting that they were already waiting in an airport queue when, in reality, they may still have been tens of miles from the airport
- Get an earlier next order by indicating “free in 10 minutes” at or near the beginning of a long assignment
- Receive orders in their home direction by indicating “going home” several times during a day.

To reduce cheating Driver Agents were designed to monitor drivers’ schedules and ignore their messages, when judged inappropriate.

The final version of the complex adaptive taxi service scheduler negotiated only with agents that were affected by a disruptive event and then modified only affected parts of the schedule. This capability was a key feature that improved overall effectiveness.
Connecting Virtual and Real Worlds

The Virtual World, which is a model of reality and resides in the scheduler, is connected with the Real World of customers, dispatchers, and drivers, as follows.

As customers ring the call centre or visit the website to place, modify or cancel an order, dispatchers enter the pertinent information into the system. Drivers communicate with the system using GPS, mobile phones, or specialised handheld devices, conveying information on their location, direction of travel, availability, etc., and, in turn, receive instructions to pick up customers.

Lessons Learned

The system began its operation and maintenance phase in March 2008, only 6 months from the beginning of the project.

Results were extremely good: 98.5% of all orders were allocated automatically without dispatcher's assistance; the number of lost orders was reduced by up to 2%; the number of vehicles idle runs was reduced by 22.5%. Each vehicle was able to complete two additional orders per week spending the same time and consuming the same amount of fuel, which increased the yield of each vehicle by 5 – 7%.

Time required to repay investments was 2 months from the beginning of the operation and maintenance phase. During the first month of operation the fleet utilization effectiveness was increased by 5 – 7%, which represents potential additional revenue of up to 5 million dollars per year. Such realized additional income has benefited both the company and the taxi drivers. According to available statistics, since 2008 driver wages have increased by 9%, and there is a possibility for an overall fleet growth.

Delayed pick-ups were reduced by a factor of 3 which considerably improved customer service. Urgent order average response time (from booking until taxi pick-up arrival) decreased to 9 minutes which is the best time among all taxi services in London. For high priority orders the response time is 5 – 7 minutes or less. Response time reductions are especially noticeable in overloaded areas.

Implementation of "on the way home" orders, an improved allocation mechanism, when compared with a previous system, gives 3 – 4 thousand miles reduction in daily fleet run, greatly benefiting both drivers and the city's ecology. Further developments targeting business effectiveness improvements may include an analysis of vehicle movements to determine actual vehicle velocities that could improve courier service by increasing the number of orders per courier.

References

Works Cited


Primary References

Additional References
None.

SEBoK Discussion
Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].
Submarine Warfare Federated Tactical Systems Case Study

This article describes the transformation of the systems engineering and integration program that produces the common combat system used across the United States Navy (USN) submarine fleet from traditional document-based systems engineering (DBSE) to model-based systems engineering (MBSE). The topic may be of particular interest to those dealing with programs in the sustainment and evolution phase of their life cycle. For additional information, refer to the links provided in Section V, Lessons Learned below.

Background

Modern submarines are typically in service for 20 - 40 years. Submarines and their internal systems are typically state-of-the-practice at launch, but most navies find it necessary to upgrade the ship's combat system at least once during the operational lifetime. The evolution of threats, technology and interoperability drives the USN to upgrade their submarine combat systems and key components including the sonar (Fages 1998) (Ford and Dillard 2009) and tactical control systems continuously (Jacobus and Barrett 2002).

Over the last three decades submarine combat systems have evolved from multiple independent systems (sonar, combat control, imaging, electronic warfare, weapon control, etc.) with manual or point-to-point interfaces into networked federations of systems (FoS). Confusingly, these component systems are often referred to as subsystems in the literature. Typically, each new class of submarines has been equipped with a new combat system, with a corresponding logistics and sustainment tail unique to that class.

In the USN, each of these component systems has its own acquisition program, customer, and contractor team. Starting as legacy military systems hosted on traditional military-unique computational platforms, these systems have evolved to utilize Commercial Off-The-Shelf (COTS) computational and networking platforms, and leverage large amounts of COTS software.
As the component systems became more tightly interconnected, the acquisition customers established and collaboratively funded a systems engineering and integration (SE&I) program to manage the interfaces between systems, manage technology insertion and the obsolescence of common COTS hardware, and to integrate and test the production systems (Cooper, Sienkiewicz and Oliver 2006). Starting with the Virginia class, this SE&I program was expanded to encompass both new-production and in-service submarine modernization efforts. Over time, the combat systems of the various submarine classes were converged into variants of a single product line (Zingarelli, et al. 2010). In addition, the independent system-to-system interfaces were rationalized where practical into common interfaces shared between multiple systems.

**Purpose**

The submarine combat system SE&I program delivers an updated production baseline annually, along with product line variants for each submarine class or subclass being built or upgraded that year. Production systems implementing this baseline are delivered to new-build submarines, and to in-service submarines being upgraded on a roughly six-year cycle. The common combat system product line is referred to as the Submarine Warfare Federated Tactical Systems (SWFTS). SWFTS is deployed by the USN on submarines of the Los Angeles (SSN 688), Ohio (SSGN 726, SSBN 730), Seawolf (SSN 21), and Virginia (SSN 774) classes, and by the Royal Australian Navy on the Collins (SSG 73) class. SWFTS is also planned for the next-generation USN Columbia (SSBN) and RAN Future Submarine classes. Compared to the submarine combat systems that it replaced, SWFTS significantly reduces development, maintenance and training costs while delivering enhanced combat capabilities and facilitating the rapid insertion of new or improved capabilities (Zingarelli, et al. 2010).

![Networked architecture of the initial VIRGINIA Class Non-Propulsion Electronic Systems](Produced by Lockheed Martin for US Navy. Approved for Public Release by US Navy, #16-348, June 2016)
Submarine Warfare Federated Tactical Systems Case Study

Challenges
The USN submarine fleet encompasses substantial platform variability between class, sub-classes, and even individual ships within a sub-class. The RAN Collins class contributes additional variability. Platform variability drives combat system variability.

SWFTS is a Federation of Systems (FoS), with each platform hosting a subset of 40 systems produced by 20 different program offices. As is common with System of Systems (SoS) and FoS, there is no central program office that can command the compliance of all of the component system programs. Instead, the evolution of SWFTS is executed through negotiation and consensus.

Many baselines must be produced each year: new common hardware baselines are introduced in odd years, while new common software baselines are introduced in even years (Jacobus, Yan and Barrett 2002). In addition, multiple incremental developmental baselines are established each year. Once the annual production baseline for the product line is defined, variants must be developed for each submarine class or subclass built or upgraded that year (Mitchell 2012).

Like most other defense programs, the SWFTS SE&I program is under constant pressure to accomplish more with decreasing resources. There has been steady increase in SE scope despite decreasing budgets. Program leadership has responded in part through continuous SE process improvement. Improvements have included test automation, changes in the requirements management processes and tools (spreadsheets to IBM® Rational® DOORS® to OMG® SysML®), refined tooling for change management, and the DBSE to MBSE transition that is the focus of this case study. Substantial Return on Investment (ROI) has been achieved with each major SE process or tooling improvement (Mitchell 2014).

Systems Engineering Practices
During 2009 the SWFTS SE&I program conducted a Model Driven Architecture (MDA) study to determine if MDA should be the next step in the program’s continuous SE process improvement. The MDA study predicted a positive Return on Investment (ROI) from converting to MBSE. In January 2010 the SWFTS SE&I program office initiated a three-year effort to develop and validate a SWFTS FoS model and MBSE process. In 2013 a SWFTS baseline was developed using both DBSE and MBSE in parallel. The DBSE products were used to validate the MBSE products. Based on that successful validation, the SWFTS SE&I program transitioned to MBSE for all ongoing work.

Until the transition in 2013, SWFTS SE was performed using traditional DBSE. Requirements were managed in DOORS, and reviewed and used by engineers in the form of massive spreadsheets with hundreds of columns and thousands of rows. The design of each variant was documented in Microsoft Office files. Baseline change requests (BCR) were documented in briefings, and analyzed by all component system programs in parallel for potential impact. Approved BCRs were manually merged into DOORS and into revised baseline documents for each effected variant.

Starting in 2010, the customer community invested in a three-year MBSE transformation effort. The engineering team performed an in-depth tool trade study to select and set up the MBSE environment. That trade study resulted in the selection of MagicDraw™ for system modeling, using Teamwork™ as the model repository.

Once the modeling environment was installed, the MBSE transformation team architected, developed and populated a SysML-based model of the SWFTS FoS interfaces. The team updated the SWFTS SE process to take advantage of the new MBSE environment, with the constraint that the MBSE process produce SE products that were effectively identical to those produced by the DBSE process.

In 2013 the new MBSE environment and process was used to produce a set of SWFTS baseline SE products. In parallel, the DBSE process produced equivalent products. The two sets of baseline products were compared in detail, with all differences traced back to root causes.
This analysis identified a significant number of minor differences that were all traced to errors in the DBSE products. This validated the MBSE process, and demonstrated that while those initial MBSE baseline products were more labor-intensive than the DBSE baseline products, the new MBSE process produced higher quality products. After this validation, the SWFTS SE&I program switched over to MBSE as their basic process.

Since that transformation, SE process improvement has continued apace. Requirements management has moved from DOORS to the system model in MagicDraw. As of 2016, the system model is the baseline for requirements, architecture and the FoS design. BCR impact analysis is now performed in model, leveraging capabilities of the toolset for automated assistance. Variants are documented in the system model as system configurations. Most SE products are generated from system model on demand.

While the initial scope of MBSE was limited to managing the interfaces between component systems, once the transition was successful MBSE started expanding out to encompass additional SWFTS SE&I tasks. MBSE is now beginning to spread into the component system programs as well as the overall submarine combat system SE&I program.

**Lessons Learned**

Seven learning principles (LPs) (Friedman and Sage 2005) were derived that address the more broadly applicable areas of systems engineering knowledge, and inform the areas of the SEBoK that are most strongly related to the case study. They are:

- Requirements traceability (LP1);
- Communications (LP2);
- Productivity (LP3);
- Quality (LP4);
- Managing Change (LP5);
- Managing variants (LP6); and
- Life cycle (LP7).

**Requirements traceability** LP1: MBSE improves traceability in multiple dimensions, but maintaining requirements traceability between a traditional database and the MBSE system model can be challenging. While DOORS, MagicDraw and Teamwork can interoperate to provide requirements management and traceability, the combination is fragile. Without careful configuration management, synchronization can lead to database corruption. If DOORS and Teamwork are on separate servers, maintaining the connection can run afoul of ever-evolving corporate information assurance (IA) policies.

Requirements can be managed using the SysML language inside the system model quite effectively. This approach can reduce the resources needed to keep the system model in sync with a traditional requirements database system and increase overall SE productivity.

**Communications** LP2: Tailored SE products generated from the system model can substantially enhance communications both within the technical team and between customer stakeholders. Graphical depictions of the system model often communicate better to human stakeholders than massive spreadsheets and textual documents. Further, the enhanced precision driven by modeling can reduce miscommunications between both technical and programmatic stakeholders.

Having the architecture and design in a system model makes it affordable to generate specialized SE products on demand for particular communications needs while keeping all SE products in concordance. Even technical stakeholders who thought they understood the design can find new insights by looking at it in different representations.

**Productivity** LP3: MBSE increases productivity by enhancing communications within the team, automation of routine tasks and through cost avoidance. Processes used in DBSE often require substantial revision to achieve the
potential productivity gains of MBSE. In particular, review processes must be modified to take advantage of the tooling.

The selected modeling tools constrain how you can practically re-engineer SE processes. Automation can replace a great deal of routine SE work (document generation, identifying potential impacts of changes, etc.). Developing strong modeling style guidelines and specialized representations, along with training materials to indoctrinate new team members as they join the program, is worth the investment. MBSE does require a trained cadre of modelers, but not all systems engineers have to become skilled modelers.

To effectively quantify the benefits of MBSE, a program needs to plan metrics collection carefully, and then stick to the plan long enough to collect meaningful data.

**Quality LP4:** Much of the ROI from the MBSE transition can be in improved quality. Improving the quality of SE products reduces latent defects in systems delivered to the customer, reducing maintenance costs and increasing customer satisfaction

Models are less tolerant of imprecision than documents. The increased precision improves SE product quality, both in reduced defect generation and in reduced defect escape. The automation of product generation can make specialized SE products affordable, further enhancing system quality.

**Managing change LP5:** How a proposed system change is understood and executed is fundamentally different between a model- and a document-centric approach. In the document-centric approach, the focus is on “What should my final output look like?” In the model-based approach, the focus is on “What does this change mean to the system? Which other parts of the system are impacted by this change?”

Change management is hard. When moving from DBSE to MBSE you need to think carefully up front about what approach you are going to take, and then design the system model to facilitate that approach. Change management also impacts tool selection, since different tools align better with different approaches.

**Managing variants LP6:** The most common process for managing variants in DBSE is ‘clone and own’, where each new product family member takes the then-current baseline and ‘forks’ the baseline for evolution of the variant. This makes synchronizing changes to the core baseline across the product family a very labor-intensive process. Treating variants as deviations from a core baseline in a model greatly reduces the cost of managing variation in a product family.

Variant management is hard. You need to think about what approach you plan to take up front, and design your system model to accommodate it. The selected approach impacts tool selection and tailoring.

Design the system model to treat variants as deviations from the core baseline. Then changes to the core baseline are automatically shared among all variants, and impact to product family members is limited to any impact of core baseline change on specific variant deviations. This also facilitates commonality between variants, a key customer goal as commonality reduces logistics and training costs.

**Life cycle LP7:** MBSE can be applied early or late in the product family life cycle. While most projects using MBSE start off model-based, a program can transition to MBSE late in the life cycle.

Getting from DBSE to MBSE requires serious engineering, careful thought, planning and implementation. The SWFTS MBSE transition required three years of investment by the customer. That time and budget was spent primarily in designing and developing the system model and in re-engineering the SE processes.

Start the transition with carefully defined scope. Once that is accomplished you can expand the scope of MBSE from there.
References

Works Cited
Cooper, Dennis J., Joan Sienkiewicz, and Mike Oliver, "System Engineering and Integration for Submarine Combat Systems in the COTS Environment", NDIA Systems Engineering Conference, San Diego, CA, 23-26 October 2006


Primary References


Additional References


< Previous Article | Parent Article | Next Article>

SEBoK v. 1.8 released 27 March 2017

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and
Northwest Hydro System

This case study presents the Northwest Hydro System (NwHS), which is situated in the northwestern United States (the Northwest). NwHS comprises a large number of loosely coupled autonomous elements (hydroelectric dams) that operate in a complex environment of social, regulatory, and ecological contexts. It is instructive to note that the NwHS is characterized as a project that is concerned with planning, developing, and maintaining a hydroelectric system that has evolved, and continues to evolve, over time. Each of the hydroelectric dams within the NwHS is also referred to as a project, which indicates that the individual elements of the NwHS are also evolving over time (FWEE 2016).

Background

As is shown in this case study, NwHS is an adaptive and reconfigurable system that exists within a framework of policies, rules, regulations, and agreements. The NwHS is analyzed using each of the four provisioning paradigms in SEBoK Part 4: product system engineering; service system engineering; enterprise system engineering; and system of systems engineering.

NwHS encompasses the Columbia River and its tributaries. The headwaters of the Columbia are in the Rocky Mountains of British Columbia, Canada. The river flows south into the United States and then east to west; it forms the north-south border between the states of Washington and Oregon in the U.S. and empties into the Pacific Ocean near Astoria, Oregon. The Columbia River drainage basin (Columbia and tributaries) is roughly the size of France and extends across seven U.S. states and British Columbia (Col 2016).

The Columbia River has 14 hydroelectric dams (hydro dams) on its main stem. In total, there are more than 250 hydro dams in the Columbia River Basin, each of which has a generating capacity of five or more megawatts of electricity. The NwHS produces approximately one-third of all hydroelectric power generated in the United States - more than any other North American hydroelectric system. The amount of electrical power generated by the NwHS fluctuates between 50% and 75% of all electricity used in the Northwest; other sources include coal, natural gas, nuclear, wind, and solar. Excess electrical energy generated by NwHS is sold to other electrical grids. There are more small hydro dams than large dams. Small hydro dams have a generating capacity of 100 kilowatts to 30 megawatts; large hydro dams have a capacity of more than 30 megawatts. In addition, there are numerous micro dams that each can generate fewer than 100 kilowatts. Most micro dams provide power to isolated homes or small communities but some are elements of the NwHS and sell power to utilities (DOE 2016). Utility companies own some of the dams, some are locally and privately owned, and some are owned and operated by federal agencies.

The Bonneville Power Administration (BPA) provides about one-third of the electrical power generated by NwHS and used in the Northwest; BPA is a federal nonprofit agency – it is part of the U.S. Department of Energy. It is self-funded and covers its costs by selling electrical power generated by 31 federal hydro dams in the Columbia River Basin. The U.S. Army Corps of Engineers and the Bureau of Reclamation operate the dams.

The Bureau of Reclamation also operates a network of automated hydrologic and meteorologic monitoring stations located throughout the Northwest. This network and its associated communications and computer systems are collectively called Hydromet. Remote data collection platforms transmit water and environmental data via radio and
satellite to provide near-real-time water management capabilities. Other information, as available, is integrated with Hydromet data to provide timely water supply status for river and reservoir operations (USBR 2016).

Distinguishing characteristics of individual hydro dams are their capacity to generate electrical energy and their physical structure. Several factors contribute to the differences in generating capacity of hydro dams: the number of turbines/generators; the flow rate of a river or tributary; the amount of elevation that water falls in order to spin the turbines; and environmental factors such as providing for fish passage and regulating water flow to provide irrigation water and maintain downstream ecosystems.

Dam structures include storage dams, run-of-river dams, and pumped storage dams (DOE 2016). A storage dam retains water in a reservoir for future use. A run-of-river dam harvests the energy in a river as it flows through the dam but does not impede the flow. Pumped storage dams use generated electricity to pump water back up to a storage reservoir during times of low demand for use during times of high demand. Pumped storage dams are not common in the NwHS.

**Purpose**

NwHS has three primary goals (NwHS 2016):

1. To provide most of the Northwest’s "firm-energy" needs.
2. To maximize "non-firm" energy production.
3. To maintain the ecological environment.

Firm energy is the amount of electricity the Northwest will need each year. Planners rely on NwHS and other sources to produce enough firm energy to ensure that sufficient electricity will be generated to meet estimated needs (energy sources include hydroelectric, nuclear, coal, natural gas, solar, and wind). NwHS “firm energy” is the amount of electricity than can be generated by NwHS when the amount of available water is at a historical low, thus guaranteeing the amount of energy the NwHS can provide.

Non-firm energy is the electricity generated when the annual hydrologic cycle makes more water available for power generation than in a historically low-water year. Non-firm electricity generated by hydro dams is generally sold at a lower price than the alternatives of electricity generated by nuclear, coal, or natural gas thus making it more attractive to customers. Excess non-firm electricity is also sold to interconnected regional grids when the demand on those grids exceeds supply.

Other goals for NwHS include flood control, navigation, irrigation, and maintaining the water levels of all reservoirs.

**Challenges**

The NwHS is a large, complex system that has many challenges to be met.

NwHS hydro dams have varying design details (e.g., the types of turbines, generators, control systems, and fish passage facilities used). This makes routine maintenance, retrofitting, and other sustainment issues unique for each dam.

Safety and security (both physical and cyber) are common challenges for all dams; cyber security is a growing concern. Smaller dams, having fewer resources, may be more susceptible to cyber attacks than larger ones. It has been reported that on 12 occasions in the last decade hackers gained top-level access to key power networks (HLS 2010) (Cyber 2015).

Run-of-river hydro dams do not store water. They may not be able to meet their firm energy commitments when rivers are lower than anticipated and flowing slowly.

The ways in which electricity generated by a dam is transmitted and sold to utilities and large industrial customers varies widely. For instance, the Bonneville Power Administration transmits and sells power generated by federal dams. Non-federal operators must manage transmission and sale of power produced by their dams.
Depending on factors such as size, structure, location, and ownership of each dam, a large number of policies, regulations, and agreements have dramatically different effects on how dams are operated.

Some of the most contentious environmental issues are associated with maintaining the ecology of the rivers, preserving salmon and other fish, and providing sufficient irrigation water while preserving sufficient reservoir water to meet firm-energy demands (Speakout 2016).

Preserving the salmon population endangered by dams is a continuing challenge. Salmon populations have been depleted because dams impede the return of salmon to upstream spawning beds. Native Americans advocate for their traditional fishing rights, which conflict with governmental policies intended to maintain healthy salmon populations in the Columbia River Basin (Impact 2016). The National Marine Fisheries Service has recently declared that salmon recovery is a higher priority than all other purposes except flood control at 14 federal dams.

**Systems Engineering Practices**

The Northwest Hydro System is a large, complex system composed of loosely coupled autonomous elements; each dam operates semi-independently within a large network of similar entities and contextual constraints. The NwHS evolves over time: some dams have been retrofitted to increase power generation capability or to reconfigure connections to electrical transmission lines; new dams have been constructed; and some existing dams have been decommissioned and removed.

Human elements of NwHS include: operators; maintainers; regulators; and inspectors. Others are suppliers to NwHS (vendors and contractors); some humans are users of the electricity generated by NwHS (businesses and home owners); and some are stakeholders who depend on and are impacted by the NwHS (utilities, large industries, farmers, ranchers, homeowners, ecology advocates, Native Americans, towns).

The context of NwHS includes natural elements (rivers, terrain, weather systems, fish); elements purposefully built by humans (transmission lines, electrical grids); cyber connections (both wired and Internet); and rules, regulations, and agreements at the federal, regional, state, and local levels.

Given the complexity of the NwHS and its context, it is instructive to analyze the NwHS by applying each of the four application paradigms of systems engineering presented in SEBok Part 4: the product, service, enterprise, and system-of-systems application paradigms.

**Product system provisioning**

Product system provisioning applies systems engineering processes, methods, tools, and techniques to conceive, develop, and sustain the purposefully developed elements of a system (e.g., a hydro dam or a hydro system). In addition, some of the naturally occurring physical elements of a system may be shaped and configured (e.g., a river channel).

Major product elements of a hydro dam include the physical structure of the dam (including the spillway), the penstock (used to direct water into the turbines), the generating plant (i.e., the turbines used to turn generator rotors, generator stators and rotors that generate the electricity, step-up transformers used to increase the voltage level of electricity produced by the generators, and connections to transmission lines).

Cyber elements sense, measure, regulate, and control water flow, power generation, safety, security, and the structural integrity of the dam. Some turbines, for example, have adjustable vanes that are controlled to harvest maximum energy from the water, depending on the flow rate, power demand, and other factors. The cyber elements include: computing devices; supporting software (operating systems, databases, spreadsheets); data management software (collection, analysis, reporting); application software (displays of monitored status and interfaces for controlling operation of a dam); and communication interfaces to wired linkage and Internet-enabled links. In addition, software support is provided for the analog and digital devices needed to sense, measure, regulate, and control the purposefully built and naturally occurring elements of a dam and its environment.
Product system provisioning is also concerned with other issues that apply to individual dams, elements of dams, and the overall NwHS. They include issues such as: manufacturability/producingility; logistics and distribution; product quality; product disposal; conformance to policies, laws, regulations, agreements, and standards; value added for stakeholders; and meeting customer's expectations. Many different technologies and engineering disciplines are needed to develop and sustain a hydro dam and the overall Northwest Hydro System. Product system provisioning can provide the coordination and control of systems engineering needed to develop, reconfigure, adapt, analyze, and sustain the hydro dams and the NwHS.

Service system provisioning
A service is an activity performed by an entity to help or assist one or more other entities. Service system provisioning can be applied within the various contexts of services provided by the NwHS to meet stakeholders' requirements, users' needs, and system interactions with operators, users, and maintainers, plus the interactions with the contextual elements that determined services provided by the NwHS in the social, business, regulatory, and physical environments.

The NWHS provides electricity to a grid that serves commercial, industrial, governmental, and domestic customers. Stakeholders in addition to customers served include those who affect or will be affected by development, operation, and sustainment of a dam. Downstream stakeholders served, for example, include, Native Americans, farmers and ranchers, and communities that receive the service of water released by the dam.

Additional service attributes include: the services that enable operators and maintainers to efficiently and effectively operate and maintain the physical and cyber elements of a dam; release water from the dam in a manner that services the upstream and downstream ecosystems; manage sharing of electrical power with other regional grids; provide emergency responses to power demands that result from electrical brownouts, blackouts, and overloads; and handle system failures that might be caused by earthquakes, terrorist attacks, and other catastrophic events.

Enterprise system provisioning
An enterprise, such as the NwHS, consists of one or more organizations that share a mission, goals, and objectives to offer an output such as a product or service. The mission and goals of the NwHS are to provide most of the Northwest's firm energy needs and to maximize non-firm energy production while serving stakeholders and preserving affected environmental ecosystems. To meet those goals, the Northwest Hydroelectric Association (NWHA) coordinates the planning, design, improvement, and operation of the hydro dams that constitute the NwHS enterprise.

NWHA members represent all segments of the hydropower industry – independent developers and energy producers; public and private utilities; manufacturers and distributors; and local, state and regional governments including water and irrigation districts. Other NWHA members include contractors, Native American tribes, and consultants: engineers, financiers, environmental scientists, attorneys and others (NWHA 2016).

Note that an enterprise may consist of multiple organizations that are engaged in a common endeavor. The NwHS is a large complex enterprise that has many constituent organizations; namely, the organizations that own and operate the hydro dams and the other stakeholder members of the NWHA. Differences in ownership, structure, location, and size of hydro dams, the special interests of various NWHA members, and a complicated regulatory process, are some of the distinguishing characteristics of the NwHS that can be analyzed by enterprise systems provisioning.
System of Systems provisioning

Many systems are composed of autonomous elements that are combined to provide increased capabilities that cannot be provided by the elements operating in isolation. The Northwest Hydro System is a system of systems comprised of autonomous hydro dams that have different owners, different operators, different stakeholders, and different regulators. The autonomous hydro dams could not provide the NwHS capabilities without the overall coordination and control that can be managed by applying system of systems provisioning.

Lessons Learned

NwHS is a collection of many interrelated ongoing projects that have shared common goals and shared constraints. The unique characteristics of NwHS make it a useful case study to illustrate how the four provisioning paradigms in SEBoK Part 4 provide essential viewpoints for analyzing large complex systems comprised of loosely coupled, autonomous elements.

Product systems engineering allows the collection of physical and purposefully built NwHS elements and their interconnections to be analyzed by applying systems product engineering processes and methods.

Service systems engineering supports analysis of the NwHS services provided to customers, users, farmers, ranchers, Native Americans, and other stakeholders who rely on NwHS for those services.

Enterprise systems engineering considers the broad scope and impact of the NwHS enterprise, both positive and negative on the northwestern United States within the context of economic, social, physical, and regulatory environments.

System of Systems engineering applies the principles of planning, coordination, and operation to a collection of semi-autonomous hydro dams that form the Northwest Hydro System. The complexity of adding new dams as well as modifying and decommissioning existing dams in a seamless manner can best be understood by applying system of systems engineering processes and methods.

Taken together, the four provisioning paradigms in SEBoK Part 4 present a comprehensive view of a very large complex system whose many dimensions would be otherwise difficult, if not impossible, to comprehend when the NwHS is examined using only one of the paradigms.

References

Works Cited


**Primary References**


**Additional References**


---

< Previous Article | Parent Article | Next Article>

SEBoK v. 1.8 released 27 March 2017

**SEBoK Discussion**

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox[^1].

[^1]: ENCODED_CONTENT

MTg2OTIPGRpdIpZD0iZGlzczXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXJldnQg

END_ENCODED_CONTENT
Vignettes

Systems engineering (SE) principles described in the SEBoK Parts 1-6 are illustrated in Part 7, Systems Engineering Implementation Examples. These examples describe the application of systems engineering practices, principles, and concepts in real settings. These examples can be used to improve the practice of systems engineering by illustrating to students and practitioners the benefits of effective practice and the risks of poor practice.

The SEBoK systems engineering implementation examples are grouped in two categories: case studies and vignettes. The SEBoK examines case studies previously published by external sources and demonstrates the real world examples of systems engineering principles that are present in these studies. The vignettes are short wiki articles written specifically for the SEBoK. These vignettes were developed to illustrate the applicability of systems engineering principles in a broader range of domains and geographic locations.

A matrix is used to map the implementation examples to topics in the SEBoK. This matrix maps each implementation example to the discussion of the specific systems engineering principles illustrated.

List of Vignettes

The following vignettes are included:

- Denver Airport Baggage Handling System Vignette
- Virginia Class Submarine Vignette
- UK West Coast Route Modernisation Project Vignette
- Singapore Water Management Vignette
- FAA Advanced Automation System (AAS) Vignette
- Standard Korean Light Transit System Vignette

References

None.

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.

If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox [1].

ENCODED_CONTENT

MTU4MDYpGRdpiBpZH0/ZGlzcXVzX3RoemVhZCI+PC9kaXY+CjxzY3JpcgdHlwZT0idGV4dC9qYXZhc2Ny
END_ENCODED_CONTENT
Denver Airport Baggage Handling System Vignette

This vignette describes systems engineering (SE) issues related to the development of the automated baggage handling system for the Denver International Airport (DIA) from 1990 to 1995. The computer controlled, electrical-mechanical system was part of a larger airport system.

Vignette Description

In February 1995, DIA was opened 16 months later than originally anticipated with a delay cost of $500 million (Calleam Consulting Ltd. 2008). A key schedule and cost problem—the integrated automated baggage handling system—was a unique feature of the airport. The baggage system was designed to distribute all baggage automatically between check-in and pick-up on arrival. The delivery mechanism consisted of 17 miles of track on which 4,000 individual, radio-controlled carts would circulate. The $238 million system consisted of over 100 computers networked together, 5,000 electric eyes, 400 radio receivers, and 56 bar-code scanners. The purpose of the system was to ensure the safe and timely arrival of every piece of baggage. Significant management, mechanical, and software problems plagued the automated baggage handling system. In August 2005, the automated system was abandoned and replaced with a manual one.

The automated baggage system was far more complex than previous systems. As planned, it would have been ten times larger than any other automated system, developed on an ambitious schedule, utilized novel technology, and required shorter-than-average baggage delivery times. As such, the system involved a very high level of SE risk. A fixed scope, schedule, and budget arrangement precluded extensive simulation or physical testing of the full design. System design began late as it did not begin until well after construction of the airport was underway. The change management system allowed acceptance of change requests that required significant redesigns to portions of work already completed. The design did not include a meaningful backup system; for a system that required very high mechanical and computer reliability, this increased failure risks. The system had an insufficient number of tugs and carts to cope with the volume of baggage expected and this, along with severely limited timing requirements, caused baggage carts to jam in the tracks and for them to misalign with the conveyor belts feeding the bags. This resulted in mutilated and lost bags (Neufville 1994; Gibbs 1994).

The baggage system problems could be associated with the non-use or misuse of a number of systems engineering (SE) concepts and practices: system architecture complexity, project scheduling, risk management, change management, system analysis and design, system reliability, systems integration, system verification and validation/testing, and insufficient management oversight.

Summary

The initial planning decisions, such as the decision to implement one airport-wide integrated system, the contractual commitments to scope, schedule, and cost, as well as the lack of adequate project management (PM) procedures and processes, led to a failed system. Attention to SE principles and practices might have avoided the system’s failure.

References

Works Cited


**Primary References**

None.

**Additional References**


---

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

*If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox*[1].

[1] ENCODED_CONTENT
OTc2NDIPGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXEND_ENCODED_CONTENT
Virginia Class Submarine Vignette

Prior to the Virginia class submarine, sonar systems were comprised of proprietary components and interfaces. However, in the mid-1990s the United States government transitioned to the use of commercially developed products - or commercial off the shelf (COTS) products - as a cost-saving measure to reduce the escalating costs associated with proprietary-based research and development. The Virginia class submarine system design represented a transition to COTS-based parts and initiated a global change in architectural approaches adopted by the sonar community. The lead ship of the program, Virginia, reduced the number of historically procured parts for nuclear submarines by 60% with the use of standardization. The Virginia class submarine sonar system architecture has improved modularity, commonality, standardization, and reliability, maintainability and testability (RMT) over historical sonar systems.

Architectural Approach: Standardization

Based on the new architectural approach and the success of the transition, system architecture experts developed an initial set of architecture evaluation metrics:

- Commonality
  - Physical commonality (within the system)
    - Hardware (HW) commonality (e.g., the number of unique line replaceable units, fasteners, cables, and unique standards implemented)
    - Software (SW) commonality (e.g., the number of unique SW packages implemented, languages, compilers, average SW instantiations, and unique standards implemented)
  - Physical familiarity (with other systems)
    - Percentage of vendors and subcontractors known
    - Percentage of HW and SW technology known
- Operational commonality
  - Percentage of operational functions which are automated
  - Number of unique skill codes required
  - Estimated operational training time (e.g., initial and refresh from previous system)
  - Estimated maintenance training time (e.g., initial and refresh from previous system)
- Modularity
  - Physical modularity (e.g., ease of system element or operating system upgrade)
  - Functional modularity (e.g., ease of adding new functionality or upgrading existing functionality)
- Orthogonality
  - Level to which functional requirements are fragmented across multiple processing elements and interfaces
  - Level to which throughput requirements span across interfaces
  - Level to which common specifications are identified
- Abstraction (i.e., the level to which the system architecture provides an option for information hiding)
- Interfaces
  - Number of unique interfaces per system element
  - Number of different networking protocols
  - Explicit versus implicit interfaces
  - Level to which the architecture includes implicit interfaces
  - Number of cables in the system
- Standards-based openness
• Interface standards
  • Ratio of the number of interface standards to the number of interfaces
  • Number of vendors for products based on standards
  • Number of business domains that apply/use the standard (e.g., aerospace, medical, and telecommunications)
  • Standard maturity
• Hardware standards
  • Ratio of the number of form factors to the number of line replaceable units (LRUs)
  • Number of vendors for products based on standards
  • Standard maturity
• Software standards
  • Number of proprietary and unique operating systems
  • Number of non-standard databases
  • Number of proprietary middle-ware
  • Number of non-standard languages
• Consistency orientation
  • Common guidelines for implementing diagnostics and performance monitor/fault location (PM/FL)
  • Common guidelines for implementing human-machine interface (HMI)
• Reliability, maintainability, and testability
  • Reliability (fault tolerance)
  • Critical points of fragility (e.g., system loading comprised of percent of processor, memory, and network loading)
  • Maintainability (e.g., expected mean time to repair (MTTR), maximum fault group size, whether the system can be operational during maintenance)
  • Accessibility (e.g., space restrictions, special tool requirements, special skill requirements)
  • Testability
    • Number of LRUs covered by built-in tests (BIT) (BIT coverage)
    • Reproducibility of errors
    • Logging/recording capability
    • Whether the system state at time of system failure can be recreated
    • Online testing (e.g., whether the system is operational during external testing and the ease of access to external test points)
    • Automated input/stimulation insertion

Other Points
The Virginia class submarine acquisition exhibited other best practices. These are discussed by Schank (2011), GAO (2008), and General Dynamics (2002).

These best practices included stringent design trades to keep costs under control, careful consideration of technical maturity of components, and the importance of program stability.

Summary
In summary, the work on the Virginia class submarine prompted a change in the traditional architectural approach used in the sonar community to design submarine sonar and validated the cost savings in both research and development (R&D) and in component costs when transitioning from proprietary interfaces to industry standard interfaces. The identification of a list of feasible architecture evaluation metrics was an added benefit of the effort.
References

Works Cited


Primary References

None.

Additional References

None.

SEBoK Discussion

Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication*. *Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

**If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox**[1].

References

UK West Coast Route Modernisation Project Vignette

The West Coast Main Line (WCML) is a principal United Kingdom (UK) railway artery serving London, the Midlands, the North West and Scotland. The Line is responsible for over 2,000 train movements each day, with more than 75 million rail journeys made each year on the route. It accounts for 43% of Britain's UK freight market (Railway People 2011). In 1998, the British government embarked on a modernisation program called the West Coast Route Modernisation (WCRM) project, to carry out a significant volume of modernization work between 1998 and 2008, delivering increased capacity and reduced journey times as well as replacing worn-out parts of the railway. It was a challenging job involving 640 kilometers of track—much of which was incapable of carrying high-speed rail cars. Some sections were seriously dilapidated, and new trains would require a complete overhaul of signaling, power supply, and switching systems.

This vignette is based on information from an INCOSE publication on systems engineering case studies (INCOSE 2011) and a report of the UK National Audit Office (NAO 2006).

Vignette Description

Early on, the WCRM upgrade had serious problems. A major complicating factor was the introduction of a new signaling technology that was designed to allow improved services for new trains running at 140 miles per hour. By 2001, neither the rail infrastructure upgrade nor the new trains were on course for delivery as expected in the 1998 agreement. By May 2002 the projection of the program’s final cost had risen from £2.5 billion (in 1998) to £14.5 billion, but had delivered only a sixth of the original scope.

In January 2002, the UK Secretary of State instructed the Strategic Rail Authority (SRA) to intervene and find a way to renew and upgrade the WCML. An SRA analysis identified the following issues:

- The program lacked direction and leadership before 2002.
- The project did not have a delivery strategy and there was no central point for responsibility and communication.
- There was a lack of openness and communication regarding the program with interested parties before 2002 and a lack of stakeholder management.
- Scope changes arose because WCRM did not have an agreed-upon specification that matched required outputs with inputs.
- There was inadequate knowledge about the West Coast asset condition.
- Technology issues related to the decision to replace conventional signaling with unproven moving block signaling introduced major risk into deliverability and cost before 2002. These technology issues caused scope changes and program delay.
- Project management (PM) was weak, with a lack of senior management skills, too many changes in personnel, and ill-defined and fragmented roles and responsibilities. There was no integrated delivery plan and there was limited oversight of contractors. Poor management of contracts added to costs.

In order to remedy the situation the SRA initiated the following actions, which align with generally accepted systems engineering (SE) practice:

- A clear direction for the project was developed and documented in the June 2003 West Coast Main Line Strategy, specifying desired goals and outcomes.
- A clear, measurable set of program outputs was established, along with more detailed infrastructure requirements, which were then subject to systematic change control and monitoring procedures fixing scope. Contractors were invited to tender complete detailed designs and deliver the work to a fixed price.
- Clear program governance structures were instituted.
The SRA consulted widely with stakeholders and, in turn, kept stakeholders informed. A National Audit Office (NAO) report concluded that the new arrangements worked well and that there were benefits to this approach. (NAO 2006) Until this time, one of the program's key constraints and cost drivers had been the ability to access certain areas of the track. The new approach facilitated the ability to obtain possession of the track for engineering work, which was crucial to delivery. The new approach also enabled the program to identify opportunities to reduce the total cost by over £4 billion.

The NAO report also discussed a business case analysis by the SRA that identified the following benefits (NAO 2006):

- benefit:cost ratio for the enhancements element was 2.5:1;
- journey times and train frequencies exceeded the targets set out in the 2003 West Coast Strategy;
- growth in passenger numbers exceeded expectations (e.g., by 2005-06, following Phase 1 of the West Coast program, annual passenger journeys on Virgin West Coast grew by more than 20%); and
- punctuality improved (e.g., by September 2006, average time delays on Virgin West Coast trains have been approximately 9.5 minutes, a 43% improvement on the average delay of 17 minutes in September 2004).

The WCRM problems could be associated with a number of systems engineering concepts and practices: stakeholders requirements, planning, analysis of risks and challenges of new technology and associated risk management, decision management, configuration or change management, information management, and management oversight.

**Summary**

The WCRM project illustrates that when SE concepts and practices are not used or applied properly, system development can experience debilitating problems. This project also demonstrates how such problems can be abated and reversed when SE principles and methods are applied.

**References**

**Works Cited**


Singapore Water Management Vignette

This vignette describes a systems engineering approach in the development of a sustainable National Water Management System for the Republic of Singapore. It demonstrates the successful outcome of long term planning and a systems approach to preempt a critical water shortage. The vignette is primarily based on information taken from a paper presented at the INCOSE International Symposium in 2008. (Chia 2008)

Vignette Description

When Singapore achieved independence in 1965, water supply depended on water catchment in local reservoirs and two bilateral water agreements with its closest neighbor, Malaysia. These water agreements are registered with the United Nations. The first agreement expired in August 2011, and the second agreement will expire in 2061 (Singapore 2012). After several failed attempts to renegotiate the extension of the first water agreement, Singapore determined that it was necessary to achieve full water self sufficiency by 2060 in case the second water agreement also could not be extended. An intermediate goal was to match the supply of the first water agreement before it expired. This was achieved in several ways. In 2001, the Public Utilities Board (PUB), the national water agency responsible for treating raw water in Singapore, was charged to also begin managing wastewater and stormwater, allowing for an integrated and holistic approach to water management.

This vignette examines Singapore’s water management system from a large-scale systems engineering perspective, particularly focusing on the goals, boundaries (see Concepts of Systems Thinking), stakeholders (see Stakeholder Needs and Requirements), and complexities involved in this type of a national system. This approach illustrates how Systems Thinking (illustrated through causal loop diagrams) and other systems engineering tools may be used to
understand systems complexities. Concepts and methodologies of learning organizations were applied to enable understanding of behavioral complexities. Lean thinking facilitated a long term strategic philosophy, built on the premise of continuous improvements.

Perhaps more importantly, it shows that while systems engineering, especially the Systems Approach, is necessary for the conceptualization and planning of such a complex system, it is not sufficient for success. It is the systemic structures that have been put in place over decades, political will, leadership, people, and culture that make such tasks realizable.

The supply of water in Singapore is managed in totality. Collecting rainwater, purchasing water, purifying water utilizing reverse osmosis and desalination were all considered. Approaches included even incentivising consumers to change their habits by making drains and canals recreational areas to encourage the public not to dispose of waste in their drains. By managing sewage and drainage together with water, environmental considerations are taken into account as well. By carefully adjusting organizational boundaries, Singapore has managed to reduce silo thinking and parochial interests. The relationships between the industry innovators, government, suppliers and users, and technology innovators create opportunities for Singapore’s water management. This demonstrates how multiple stakeholder interests can be combined to create a viable water management solution. Continuous improvements through the use of technology and elimination of waste, such as reducing water that is not accounted for in the system, help to assure the sustainability of an adequate supply of water for a growing Singapore population. The Singapore Water Management system is already in successful operation and is being studied by the Organisation for Economic Co-operation and Development (OECD) and by other nations.

Summary

The supply of water in Singapore is managed in totality through a systems approach, i.e., water catchment, supply, sewage and drainage. The importance of relationships between the stakeholders is also recognized. Industry innovators, political leadership, suppliers, and consumers are all involved; the project has been able to incentivize this diverse group to work together for a common goal, i.e., assuring the sustainability of an adequate supply of water for Singapore into the future.

Utilizing systems engineering and taking into consideration the systemic structures and culture required has helped Singapore achieve its first milestone of supplying its own water resources by 2010. Singapore has been able to overcome the shortfall that would have come about with the expiry of the first water agreement with Malaysia in 2011.

References

Works Cited


Primary References

None.
**FAA Advanced Automation System (AAS) Vignette**

In 1981 the Federal Aviation Administration (FAA) announced the Advanced Automation Program, which was established to modernize air traffic control (ATC) computer systems. A centerpiece of the project was the Advanced Automation System (AAS). AAS was the largest project in FAA’s history to modernize the nation’s ATC system. AAS would replace computer hardware and software as well as controller work stations at tower, terminal, and en-route facilities and allow the ATC system to accommodate forecasted large increases in traffic through the use of modern equipment and advanced software functions. (GAO 1992)

**Vignette Description**

The FAA originally proposed AAS in 1982 as a project that would cost $2.5 billion and be completed in 1996. However, substantial cost increases and schedule delays beset the AAS project over it history, caused by numerous problems in AAS development:

- The project began with a design competition between Hughes and IBM. The competition involved numerous extensions and took four years to complete. Analysis by the FAA and others pointed to inadequate consideration of user expectations and improper assessment of the technology risks. (Barlas 1996)
- The FAA pushed for 99.99999% reliability, which was considered by some “more stringent than on any system that has ever been implemented” and extremely costly. (DOT 1998)
- The program created unworkable software testing schedules - “Testing milestones were skipped or shortcutted and new software was developed assuming that the previously developed software had been tested and performed.” (Barlas 1996)
• There were an extraordinary number of requirements changes. For example, for the Initial Sector Suite System (ISSS), a key component of AAS, there were over 500 requirements changes in 1990. Because of these changes, 150,000 lines of software code had to be rewritten at a cost of $242 million. (Boppana et al. 2006)

• IBM's cost estimation and development process tracking used inappropriate data, were performed inconsistently, and were routinely ignored by project managers. The FAA conservatively expected to pay about $500 per line of computer code - five times the industry average. The FAA ended up paying $700 to $900 per line for the AAS software. (Gibbs 1994)

• In 1988, FAA estimated that the AAS program - both contract and supporting efforts - would cost $4.8 billion. By late 1993, the FAA estimated that it would cost $5.9 billion. Before the program was dramatically restructured in 1994, estimates had risen to as much as $7 billion, with key segments expected to be behind schedule by as much as 8 years. In 1994, with significant cost and schedule overruns, as well as concerns about adequate quality, usability, and reliability, the AAS program ceased to exist as originally conceived, leaving its various elements terminated, restructured, or as parts of smaller programs. (DOT 1998)

The AAS problems could be associated with the non-use or misuse of a number of systems engineering (SE) concepts and practices: system requirements, system architecture complexity, project planning, risk management, change management, system analysis and design, system reliability, system integration, system verification and system validation/testing, and management oversight.

Summary

The AAS program was the centerpiece of an ambitious effort begun in the 1980s to replace the computer hardware and software throughout the ATC system - including controller workstations, and en-route, terminal, and tower air traffic control facilities. AAS was intended to provide new automated capabilities to accommodate increases in air traffic. After sustaining serious cost and schedule problems, FAA dramatically restructured the program into more manageable pieces. This action included terminating major segments of the contract. (DOT 1998)

References

Works Cited


Primary References
None.

Additional References

SEBoK Discussion
Please provide your comments and feedback on the SEBoK below. You will need to log in to DISQUS using an existing account (e.g. Yahoo, Google, Facebook, Twitter, etc.) or create a DISQUS account. Simply type your comment in the text field below and DISQUS will guide you through the login or registration steps. Feedback will be archived and used for future updates to the SEBoK. *If you provided a comment that is no longer listed, that comment has been adjudicated. You can view adjudication for comments submitted prior to SEBoK v. 1.0 at SEBoK Review and Adjudication. Later comments are addressed and changes are summarized in the Letter from the Editor and Acknowledgements and Release History.*

*If you would like to provide edits on this article, recommend new content, or make comments on the SEBoK as a whole, please see the SEBoK Sandbox* [1].

Standard Korean Light Transit System Vignette

This vignette deals with systems engineering (SE) concepts and guidelines applied to the development of the Standard Korean Light Transit System (SKLTS). In Korea, local authorities had historically been interested in light transit to help resolve their transportation problems. The SKLTS was a joint effort between local authorities and the central government. It was built to provide a standard platform on which any local authority could construct its own light transit system. The issues of stakeholder requirements, safety, and reliability, availability, and maintainability were critical to the success of this system.

Vignette Description

The elements of the SKLTS were classified into four groups (as shown in Figure 1): trains, signal systems, electric and machinery (E&M) systems, and structures. Trains and vehicles were to be automatically operated, without need for human operators. Operation systems and their interfaces were based on digital signals and communications. For SKLTS, SE-based design activities focused on reliability, availability, maintainability, and safety (RAMS), and were integrated into project management (PM) activities during all phases.
The project life cycle for the SKLTS is summarized in Figure 2. It consisted of 7 phases: concept studies, concept development, preliminary design, design, system production and testing, performance evaluation, and operation/maintenance/close-out (OMC) - please see (Choi 2007) and (Chung et al. 2010) for further details. These phases, with the exception of the production and test phases, are completed through an evaluation and decision point (EDP) (milestone), depicted as a colored circle in Figure 2. These EDPs correspond to common life cycle artifacts such as requests for proposal (RFPs), proposals, preliminary design reviews (PDRs), and critical design reviews (CDRs).

During the SKLTS development, SE activities were focused on RAMS as summarized in Table 1.

**Table 1. The SE Framework of the SKLTS (Ahn 2005). Reprinted with permission of the Journal of the Korean Society for Railway. All other rights are reserved by the copyright owner.**

<table>
<thead>
<tr>
<th>Phases</th>
<th>Safety</th>
<th>Reliability</th>
<th>Function</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concept studies</td>
<td>Requirements analysis</td>
<td>Identifying RAM conditions</td>
<td>System configuration</td>
<td>Performance simulation</td>
</tr>
<tr>
<td>Concept development &amp; pre-design</td>
<td>Safety planning</td>
<td>RAM planning</td>
<td>Defining scenarios and alarm procedure</td>
<td>Interface analysis</td>
</tr>
<tr>
<td></td>
<td>Defining safety procedures &amp; levels</td>
<td>Initial availability analysis</td>
<td>Pre-designing command rooms</td>
<td></td>
</tr>
</tbody>
</table>
In the "concept studies" and "concept development" phases, requirements included the RAMS objectives. Planning activities in this phase included the scheduling of various tests and evaluations to be conducted after system design. The basic layout of rails and command rooms was also proposed. Finally, it was during this phase that interface management procedures and relationships between requirements and systems were defined. For RAMS engineering, it was also important to establish associated plans and criteria (e.g., RAM plans, safety plans, service availability, etc.).

During the pre-design phase, the basic architecture of the system was determined for safety planning, RAMS planning, and operational scenarios. Interfaces among subsystems were defined as well as management procedures for contractors and legal regulations. The functional analysis dealt with timeline, accuracy of stop points, and trip times. Pre-design activities also included the specifications of major system elements such as signal systems, trains, and interfaces. For RAMS engineering, safety scenarios were defined, and the hazard and risk analyses were performed.

During the design and performance evaluation phases, hazard log and RAMS analyses were performed to ensure that each subsystem met safety requirements. The specifications of alarm systems and stations were also defined. In addition, V&V and test procedures were determined for performance evaluation. During the design phase, a design/construction interface manual (D/CIM) was developed and applied to ensure integrated and consistent design. (Bombardier, 2005)

Because SKLTS was designed as an automatically-driven system, RAMS issues were critical to its success. The safety and reliability of the SKLTS were evaluated on a test railway that was constructed to standard specifications. Data was gathered from existing Korean light rail systems, as well as the light rail systems from other countries, to support V&V activities.

Various methods were applied for achieving the RAMS objectives, including RAMS requirements analysis, safety and RAMS planning, utilization of systems scenarios, and construction risk analysis.

Initial operation of SKLTS was allowed only after the system was formally accepted and operators were properly certified. During test operation, RAMS performance was continuously monitored and system scenarios were used successfully to evaluate the dynamic behavior of the system. A failure reporting and corrective action system (FRACAS) was used to gather accident and failure data. Continuous improvement when the system is in normal operation was identified as a requirement; the results from the FRACAS will be used to support improvement of the system, maintenance, and improvement of procedures.

*FRACAS: Failure Reporting & Corrective Action System
Summary
Korean local authorities have successfully introduced the SKLTS to their precincts with some modifications. Successful examples include the Inchun Airport Line and the Seoul 9th Subway Line. One lesson learned identified was that requirement analysis, especially in the first few phases, should have been more complete.

References

Works Cited

Primary References
None.

Additional References
END ENCODED CONTENT
Article Sources and Contributors

Related Disciplines

Systems Engineering and Software Engineering

Software Engineering in the Systems Engineering Life Cycle

The Nature of Software

An Overview of the SWEBOK Guide

Key Points a Systems Engineer Needs to Know about Software Engineering

Software Engineering Features - Models, Methods, Tools, Standards, and Metrics

Systems Engineering and Project Management

The Nature of Project Management

An Overview of the PMI/BOK Guide

Relationships between Systems Engineering and Project Management

The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships

Procurement and Acquisition

Systems Engineering and Industrial Engineering

Systems Engineering and Specialty Engineering

Reliability, Availability, and Maintainability

Human Systems Integration

Safety Engineering

Security Engineering

Electromagnetic Interference/Electromagnetic Compatibility

Resilience Engineering

Manufacturability and Productivity

Environmental Engineering

Systems Engineering Implementation Examples

Matrix of Implementation Examples

Case Studies

Successful Business Transformation within a Russian Information Technology Company

Federal Aviation Administration Next Generation Air Transportation System

How Lack of Information Sharing Jeopardized the NASA/ESA Cassini/Huygens Mission to Saturn

Global Positioning System Case Study

Global Positioning System Case Study II

Medical Radiation Case Study
Article Sources and Contributors

**FBI Virtual Case File System Case Study**  
*Contributors:* Apyster, Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Skmackin, Smenck2, Wikiexpert, Zamoses

**MSTI Case Study**  
*Contributors:* Apyster, Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Skmackin, Smenck2, Wikiexpert, Zamoses

**Next Generation Medical Infusion Pump Case Study**  
*Contributors:* Apyster, Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Smenck2, Wikiexpert

**Design for Maintainability**  
*Contributors:* Bkcase, Radcock

**Complex Adaptive Operating System Case Study**  
*Contributors:* Bkcase, Radcock

**Complex Adaptive Project Management System Case Study**  
*Contributors:* Bkcase, Radcock

**Complex Adaptive Taxi Service Scheduler Case Study**  
*Contributors:* Bkcase, Radcock

**Northwest Hydro System**  
*Contributors:* Bkcase

**Vignettes**  
*Contributors:* Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Skmackin, Smenck2, Thilburn, Wikiexpert, Zamoses

**Denver Airport Baggage Handling System Vignette**  
*Contributors:* Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Kpailletmette, Qwang, Skmackin, Smenck2, Thilburn, Wikiexpert, Zamoses

**Virginia Class Submarine Vignette**  
*Contributors:* Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Mhenshaw, Rturner, Skmackin, Wiener, Zamoses

**UK West Coast Route Modernisation Project Vignette**  
*Contributors:* Alee, Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Rturner, Skmackin, Smenck2, Thilburn, Wikiexpert, Zamoses

**Singapore Water Management Vignette**  
*Contributors:* Alee, Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Rturner, Skmackin, Wikiexpert, Zamoses

**FAA Advanced Automation System (AAS) Vignette**  
*Contributors:* Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hlavicz, Jgercken, Rturner, Thilburn, Wikiexpert

**Standard Korean Light Transit System Vignette**  
*Contributors:* Apyster, Asquires, Bkcase, Cralvano, Dhenry, Dholwell, Hlavicz, Jgercken, Mhenshaw, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses
Image Sources, Licenses and Contributors


File:Fig_1_A_generic_life_cycle_KF.png Source: http://sebokwiki.org/w/index.php?title=File:Fig_1_A_generic_life_cycle_KF.png License: unknown Contributors: Bkcase


File:Fig1_The_Organizational_Continuum_KN.png Source: http://sebokwiki.org/w/index.php?title=File:Fig1_The_Organizational_Continuum_KN.png License: unknown Contributors: Mhenshaw, Smenck2


File:Fig_1_Integration_Process_for_Specialty_Engineering.png Source: http://sebokwiki.org/w/index.php?title=File:Fig_1_Integration_Process_for_Specialty_Engineering.png License: unknown Contributors: Mhenshaw, Smenck2


File:Figure 1.png Source: http://sebokwiki.org/w/index.php?title=File:Figure 1.png License: unknown Contributors: Mhenshaw, Smenck2

File:Figure 2.png Source: http://sebokwiki.org/w/index.php?title=File:Figure 2.png License: unknown Contributors: Mhenshaw, Smenck2


File:ChoeKimFigure1.png Source: http://sebokwiki.org/w/index.php?title=File:ChoeKimFigure1.png License: unknown Contributors: Mhenshaw, Smenck2

File:ChoeKimFigure2.png Source: http://sebokwiki.org/w/index.php?title=File:ChoeKimFigure2.png License: unknown Contributors: Mhenshaw, Smenck2