Guide to the Systems Engineering Body of Knowledge (SEBoK) v. 1.3.1

Part 6: Related Disciplines

Contents

Articles

References

Letter from the Editor

A very warm welcome to you if you are a returning SEBoK user, and in particular to anyone new to the SEBoK. As BKCASE Editor in Chief (EIC) I have overall responsibility for the continuing review and update of the SEBoK. Many thanks to the BKCASE Governors and the current members of the Editorial Board for their support during my first year in the job.

SEBoK v. 1.3.1

SEBoK v. 1.3.1 is a micro release which continues our commitment to regular review of the information referenced in our *Guide to the Systems Engineering Body of Knowledge*. Some of the updates planned for a minor update to create SEBoK v. 1.4 have been delayed by external factors. In particular updates to key external sources such as ISO/IEC/IEEE. Systems and Software Engineering -- System Life Cycle Processes and the INCOSE Systems Engineering Handbook v. 4.0 will have a significant knock on effect for the SEBoK. Other activities within our sponsoring organizations on key topics such as model based systems engineering (MBSE), systems of systems, systems engineering leadership, etc. must also be carefully considered before they are incorporated into our guide. Work has already begun on new and revised material to reflect these changes within the wider knowledge base from which the SEBoK is drawn. As these updates are completed and reviewed you will begin to see them included from SEBoK in v. 1.4, now planned for Spring 2015.

While work on new content continues we also have our ongoing activity of lower level review and engagement with the community. From this we have identified a number of smaller updates to references, terminology and organization of knowledge. These changes have been implemented in v 1.3.1. For full details of all articles affected by this update go to Acknowledgements and Release History.

Future Direction for SEBoK

In the foreword to SEBoK v. 1.3 I described the "core group of dedicated and knowledgeable contributing authors and reviewers" who make up the BKCASE community. It has been my privilege over the last 12 months to continue working with and grow this community and to expand our relationships with key organizations and groups both within systems engineering and outside of it.

The role of the Editorial Board is to work with this community of interest on an ongoing review of the current SEBoK content and structure and to develop plans for its maintenance and evolution. Our overall goals in evolving the SEBoK remain broadly the same as those outlined in the previous SEBoK updates. I have restated and slightly modified those goals below:

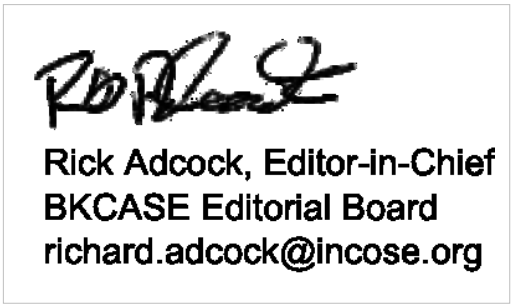
- Improve the ways in which Part 1 (SEBoK Introduction) provides a starting point for different SEBoK users to find and navigate knowledge relevant to them. This will include consideration of some of the SEBoK Use Cases which were not expanded in previous releases.
- Review of Part 2 (Systems) with help 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.
- Continue the alignment and co-evolution of Part 3 (Systems Engineering and Management) with other systems engineering life cycle documentation, in particular the planned new release of ISO/IEC/IEEE. Systems and Software Engineering -- System Life Cycle Processes and the INCOSE Systems Engineering Handbook v. 4.0.
- Assess our coverage of knowledge on systems engineering application and practices. This may include expansion of the Service and Enterprise knowledge areas in Part 4 (Applications of Systems Engineering). It will also consider how systems engineering practices such as architecting, life cycle tailoring and model based systems engineering are addressed across the SEBoK.

- 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.

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 made a good start on gathering 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,



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**
  - Paul Frenz, William Miller (Governing Board Chair from Jan 2015)
- **IEEE Computer Society**
  - Richard Fairley, Ken Nidiffer
- **SERC**
  - David Olwell, Art Pyster (Past Governing Board Chair)

Past INCOSE governors Kevin Forsberg, David Newbern, David Walden, and Courtney Wright. 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.

SEBoK Part 1 SEBoK Introduction

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

Janet Singer (USA)

Jointly responsible for the Systems Fundamentals, Systems Science and Systems Thinking knowledge areas

Mike Yearworth

University of Bristol (UK)

Jointly responsible for the Systems Fundamentals, Systems Science and Systems Thinking knowledge areas

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.

Garry Roedler

Lockheed Martin (USA)

garry.j.roedler@lmco.com [7]

Responsible for the Concept Definition and System Definition knowledge areas.

Ricardo Valerdi

University of Arizona (USA)

rvalerdi@email.arizona.edu [8]

Responsible for the System Realization knowledge area.

Ken Zemrowski

TASC

kenneth.zemrowski@incose.org [9]

Responsible for the Systems Engineering Standards knowledge area.

SEBoK Part 4: Applications of Systems Engineering

Judith Dahmann

MITRE Corporation (USA)

jdahmann@mitre.org [10]

Jointly responsible for Product Systems Engineering and Systems of Systems (SoS) knowledge areas

Rick Hefner

California Institute of Technology (USA)

Responsible for the Service Systems Engineering knowledge area.

Michael Henshaw

Loughborough University (UK)

M.J.d.Henshaw@lboro.ac.uk [11]

Jointly responsible for Product Systems Engineering and Systems of Systems (SoS) knowledge areas

James Martin *The Aerospace Corporation*

james.martin@incose.org [12]

Responsible for the Enterprise Systems Engineering knowledge area.

SEBoK Part 5: Enabling Systems Engineering

Heidi Davidz

Aerojet Rocketdyne (USA)

heidi.davidz@rocket.com [13]

Jointly responsible for the Enabling Individuals and Enabling Teams knowledge area

Emma Sparks

Cranfield University

Jointly responsible for the Enabling Individuals and Enabling Teams knowledge area

SEBoK Part 6 Related Disciplines

Alice Squires

Washington State University (USA)

alice.squires@wsu.edu [14]

Responsible for Part 6

SEBoK Part 7 Systems Engineering Implementation Examples

Brian Sauser

University of North Texas (USA)

brian.sauser@unt.edu [15]

Responsible for Part 7: Systems Engineering Implementation Examples, which includes Case Studies and Vignettes

Brian White

CAU>SE (USA)

bewhite71@gmail.com [16]

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 [17]

The Editorial Board is further supported by the BKCASE staff.

BKCASE Staff

Kate Guillemette

IEEE Computer Society (USA)

kguillemette@computer.org [18]

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
- Systems Engineering and Procurement/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 [19].

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIC
END_ENCODED_CONTENT

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:rvalerdi@email.arizona.edu
[9] mailto:kenneth.zemrowski@incose.org
[10] mailto:jdahmann@mitre.org
[11] mailto:M.J.d.Henshaw@lboro.ac.uk
[12] mailto:james.martin@incose.org
[13] mailto:heidi.davidz@rocket.com
[14] mailto:alice.squires@wsu.edu
[15] mailto:brian.sauser@unt.edu
[16] mailto:bewhite71@gmail.com
[17] mailto:c.nielsen@cranfield.ac.uk
[18] mailto:kguillemette@computer.org
[19] mailto:bkcase.incose.ieeecs@gmail.com
[20] http://www.sebokwiki.org/sandbox/

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. 1.3.1, 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 1.3.1

Changes between versions 1.3 and version 1.3.1 included:

- New primary references have been added to the Security Engineering article.
- A read-through has been completed of Part 1, Part 2 and Part 4 focused on text errors, references, citations, and general formatting for alignment with the style guide.
- A number of minor descriptions and references related to military standards, DoD System Engineering Plan (SEP) and SPRDE-SE Competency Model / ENG Competency Model has been updated to reflect newer external material available on these subjects. This has had an impact on the following articles: Systems Engineering and Procurement/Acquisition, Technical Leadership in Systems Engineering ,Developing Individuals, Assessing Individuals, Roles and Competencies, Environmental Engineering, and Safety Engineering.

SEBoK Release History

There have been 10 releases of the SEBoK to date, collected into 4 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.

Click on the links above to read more information about each release.

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.
- Nicole Hutchison, Stevens Institute of Technology
- Kate Guillemette, IEEE Computer Society

The wiki is currently supported by Daniel Robbins of WikiWorks.

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIHF
END_ENCODED_CONTENT

Part 6: Related Disciplines

Related Disciplines

Systems engineering (SE), as a discipline, intersects with other disciplines across the practice of engineering and across the enterprise. Part 6 of the SEBoK presents knowledge that would be useful to systems engineers as they interact with these other fields and experts in those fields. 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 Procurement/Acquisition
- Systems Engineering and Specialty Engineering

References

Works Cited

None.

Primary References

Abran, A. and J.W. Moore (exec. eds); P. Borque and R. Dupuis (eds.). 2004. *SWEBOK: Guide to the Software Engineering Body of Knowledge*. Piscataway, NJ, USA: The Institute of Electrical and Electronic Engineers, Inc. (IEEE). Available: http://swebok.org.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense. February 19, 2010.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogK
END_ENCODED_CONTENT

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) (Abran et al. 2004).

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.

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 Software
- An Overview of the SWEBOK Guide
- Key Points a Systems Engineer Needs to Know about Software Engineering
- Key Points a Systems Engineer Needs to Know about Managing a Software Team

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 component specialists (for example, user interface, database, computation, and communication specialists) who construct or otherwise obtain the needed software components.
- may be component specialists for certain kinds of components; they engage with other kinds of software component specialists as necessary.
- sometimes 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 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 Systems Engineering and Software Engineering (Fairley and Willshire 2011). Reprinted with permission of Dick Fairley and Mary Jane Willshire. All other rights are reserved by the copyright owner.

Systems Engineering Methods Adapted to Software Engineering	Software Engineering Methods Adapted to Systems Engineering
• Stakeholder Analysis	• Model-Driven Development
• Requirements Engineering	• UML-SysML
• Functional Decomposition	• Use Cases
• Design Constraints	• Object-Oriented Design
• Architectural Design	• Iterative Development
• Design Criteria	• Agile Methods
• Design Tradeoffs	• Continuous Integration
• Interface Specification	• Process Modeling
• Traceability	• Process Improvement
• Configuration Management	• Incremental V&V
• Systematic Verification And Validation	

References

Works Cited

Abran, A. and J.W. Moore (exec. eds); P. Borque and R. Dupuis (eds.). 2004. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Piscataway, NJ, USA: The Institute of Electrical and Electronic Engineers, Inc. (IEEE). Available at: http://www.computer.org/portal/web/swebok

Primary References

Abran, A. and J.W. Moore (exec. eds); P. Borque and R. Dupuis (eds.). 2004. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Piscataway, NJ, USA: The Institute of Electrical and Electronic Engineers, Inc. (IEEE). Available at: http://www.computer.org/portal/web/swebok

Brooks, F. 1995. *The Mythical Man-Month,* Anniversary Edition. Boston, MA, USA: Addison Wesley Longman Inc.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, NJ, USA: John Wiley and Sons.

Additional References

Pressman, R. 2009. *Software Engineering: A Practitioner's Approach.* 7th Ed. New York, NY, USA: McGraw Hill.

Schneidewind, N. 2009. *Systems and Software Engineering with Applications*. New York, NY: Institute of Electrical and Electronics Engineers.

Sommerville, I. 2010. *Software Engineering.* 9th Ed. Boston, MA, USA: Addison Wesley.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIHF
END_ENCODED_CONTENT

The Nature of Software

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, and
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 1987; Brooks 1995, 182)

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 and complex algorithms and heuristics.

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

Conformity

Software must conform to exacting specifications in the representation of each part, 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.

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

Changeability is the third of Brooks' factors that make software development difficult. 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

The fourth of Brooks' factors is 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 use different representations at different levels of abstraction in an attempt to visualize the inherently invisible entity.

Teams

In addition to the four essential properties of software identified by Brooks, one additional factor distinguishes software from other kinds of engineering artifacts. This factor is that *software projects are team-oriented, intellect-intensive endeavors* (Fairley 2009, 6).

Other kinds of engineers (including systems engineers) engage in team-oriented problem solving. Unlike other engineering artifacts, however, there is no fabrication phase for software. Software is composed from the thoughts of software engineers that flow through their fingers onto a keyboard and into a computer. Software teams are necessary because it would take too much time for one person to develop a modern software system. It is also unlikely that a single individual would possess the full range of skills required for this task.

It has also been observed that the issues of team-oriented software development are similar to the issues that would be encountered if a team of authors were to write a novel as a collaborative project (Fairley 2009).

Uniqueness

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

Brooks, F. P. 1987. "No Silver Bullet: Essence and Accidents of Software Engineering," IEEE *Computer*. 20(4) (April 1987): 10-19.

Brooks, F. 1995. *The Mythical Man-Month,* Anniversary Edition. Boston, MA, USA: Addison Wesley Longman Inc.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, New Jersey: John Wiley and Sons.

Primary References

Abran, A. and J.W. Moore (exec. eds); P. Borque and R. Dupuis (eds.). 2004. *Guide to the Software Engineering Body of Knowledge (SWEBOK).* Piscataway, NJ, USA: The Institute of Electrical and Electronic Engineers, Inc. (IEEE). Available at: http://www.computer.org/portal/web/swebok

Brooks, F. 1995. *The Mythical Man-Month,* Anniversary Edition. Boston, MA, USA: Addison Wesley Longman Inc.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, New Jersey: John Wiley and Sons.

Additional References

MITRE. 2011. "Design Patterns." *Systems Engineering Guide.* Accessed 6 March 2012 at [the MITRE site. [1]]

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKiBS
END_ENCODED_CONTENT

References

[1] http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/engineering_info_intensive_enterprises/design_patterns.html

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

Version 3 of the SWEBOK Guide (SWEBOK V3) was released at the end of 2013. The purposes of SWEBOK V3 are

- 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 describe 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
- Mathematics
- Project Management
- Quality Management
- Systems Engineering

References

Works Cited

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

IEEE. 1991. *IEEE Standard computer dictionary. A Compilation of IEEE Standard computer glossaries.* The Institute of Electrical and Electronic Engineers (IEEE). IEEE Standard 610.

Primary References

Bourque, P. and R.E. Fairley (eds.). 2014. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Los Alamitos, CA, USA: IEEE Computer Society. Available at: http://www.swebok.org

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKgpEND
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. Chapter 4 of the SWEBOK [1] (Abran and Moore 2004) discusses minimizing complexity as part of the software construction fundamentals. Zuse (1991) has a highly cited article on software complexity measures and methods. Chapter 4 of the SWEBOK also has 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 5 of the SWEBOK [2] 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 3 on Software Design [3] 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. IEEE Standard 1517-99 addresses software reuse, and Chapter 4 of the SWEBOK provides additional references. [4]
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 under Chapter 2 [5] 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 7 of the SWEBOK [6] with 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 11 of the SWEBOK [7] 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 2 of the SWEBOK discusses this here [8] and here, [9] with 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. 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 (Moore, 1965), 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.

References

Works Cited

Belady, L. and M. Lehman. 1979. "Characteristics of Large Systems." In P. Wegner (ed.), Research Directions in Software Technology. Cambridge, MA, USA: MIT Press.

Boehm, B. and V. Basili. 2001. "Software defect reduction Top 10 List." *Computer*. 34(1):135-137.

Bullock, J. 2000. "Calculating the Value of Testing." *Software Testing and Quality Engineering*, May-June, 56-62.

Fairley, R.E. and M.J. Willshire. 2011. "Teaching software engineering to undergraduate systems engineering students." *Proceedings of the 2011 American Society for Engineering Education (ASEE) Annual Conference and Exposition*. 26-29 June 2011. Vancouver, BC, Canada.

Moore, G.E. 1965. "Cramming more components onto integrated circuits," *Electronics Magazine*, April 19, 4.

Patterson, D. 2010. "The Trouble With Multicore." *IEEE Spectrum*, July, 28-32, 52-53.

Zuse, Horst. 1991. *Software Complexity: measures and methods*. Hawthorne, NJ, USA: Walter de Gruyter and Co.

Primary References

Abran, A. and J.W. Moore (exec. eds); P. Bourque and R. Dupuis (eds.). 2004. *Guide to the Software Engineering Body of Knowledge (SWEBOK)*. Piscataway, NJ, USA: The Institute of Electrical and Electronic Engineers, Inc. (IEEE). Available at: http://www.computer.org/portal/web/swebok.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, NJ, USA: John Wiley & Sons.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

Pyster, A., M. Ardis, D. Frailey, D. Olwell, A. Squires. 2010. "Global workforce development projects in software engineering." *Crosstalk - The Journal of Defense Software Engineering*, Nov/Dec, 36-41. Available at: http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA535633.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIC
END_ENCODED_CONTENT

References

[1] http://www.computer.org/portal/web/swebok/html/ch4#Ref22
[2] http://www.computer.org/portal/web/swebok/html/ch5
[3] http://www.computer.org/portal/web/swebok/html/ch3
[4] http://www.computer.org/portal/web/swebok/html/ch4#Ref18
[5] http://www.computer.org/portal/web/swebok/html/ch2#ch2-7.1
[6] http://www.computer.org/portal/web/swebok/html/ch7
[7] http://www.computer.org/portal/web/swebok/html/ch11
[8] http://www.computer.org/portal/web/swebok/html/ch2#ch2-3.2
[9] http://www.computer.org/portal/web/swebok/html/ch2#ch2-6.2

Key Points a Systems Engineer Needs to Know about Managing a Software Team

Software is a major component of many complex (glossary) systems. Because of this, a systems engineer's involvement in the technical management of a project involves leading, coordinating, and directing the work activities of the software team(s). This article summarizes the key aspects of managing software projects that a systems engineer must take into account and provides references for further reading.

Key Concepts for a Systems Engineer Managing a Software Team

1. **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 8 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.
2. **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 9. [2]
3. **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.
4. **Communication among team members** - Coordination of work activities among software development team members requires continuous communication. As famously observed by Brooks (1995), the number of communication paths among a group of individuals who must closely coordinate their work activities grows geometrically with the number of individuals: $(n*(n-1))/2$. For this reason, software development teams are usually limited to ten or fewer members. Large software projects are typically decomposed into a set of small loosely coupled teams, each of which is highly cohesive. A related key observation in (Brooks 1995) is that, contrary to many physical-system construction projects, software project communication and learning-curve effects relate to Brooks' Law: Adding manpower to a late software project will only delay the project further.
5. **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 produce an explicit design document. Martin (2003) gives a highly cited overview.

6. **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 (glossary) 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 5 of the SWEBOK [3] discusses software testing, and provides numerous references, including standards. Much has been written on the subject; a representative article is (Wallace and Fujii 1989).
7. **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.
8. **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.
9. **Software metrics include product measures and process measures** - The metrics used to measure and report progress of software projects include product (glossary) measures and process (glossary) 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 9 [4] and Chapter 8 [5] of the SWEBOK address this.
10. **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, in

the form of working software, is one of the primary advantages of the iterative development of working software increments.

11. **Software team members are not interchangeable parts** - As emphasized in (DeMarco and Lister 1987), the common practice in physical construction projects of pulling people off one project to work another generally has disastrous effects on software projects, as the productivity of software teams depends strongly on the software knowledge workers' shared knowledge and mutual trust.

References

Works Cited

Boehm, B. 1988. "A spiral model of software development and enhancement." *Computer*. 21(5):61-72.

Boehm, B. 1991. "Software Risk Management: Principles and Practices." *IEEE Software*. 8(1):32-41.

Brooks, F. 1995. *The Mythical Man-Month*. Anniversary Edition. Boston, MA, USA: Addison Wesley Longman Inc.

DeMarco, T. and T. Lister. 1987. Peopleware: Predictive Projects and Teams. New York, NY, USA: Dorset House.

Elssamadisy, A. and G. Schalliol. 2002. "Recognizing and Responding to 'Bad Smells' in Extreme Programming." Proceedings, ICSE 2002, ACM-IEEE, 617-622.

Highsmith, J.A. 2000. *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*. New York, NY, USA: Dorset.

Kitchenham, B. 1997. "Estimates, Uncertainty, and Risk." *IEEE Software*. 14(3): 69-74.

Larman, C. and V.R. Basili. 2003. "Iterative and incremental developments: a brief history." *Computer*. 36(6): 47-56.

Lederer, A.L. and J. Prasad. 1995. "Causes of inaccurate software development cost estimates." *Journal of Systems and Software*. 31(2):125-134.

Magazinius, A. 2012. "Exploring Software Cost Estimation Inaccuracy." Doctoral Dissertation. Chalmers University of Technology. Goteborg, SE.

Martin, R.C. 2002. *Agile Software Development: Principles, Patterns and Practices.* Upper Saddle River, NJ, USA: Prentice Hall.

Royce, W.W. 1970. "Managing the development of large software systems." *Proceedings of IEEE WESCON.* August, 1970. Available at http:/ / leadinganswers. typepad. com/ leading_answers/ files/ original_waterfall_paper_winston_royce.pdf [6].

Wallace, D.R. and R.U. Fujii. 1989. "Software verification and validation: an overview." *IEEE Software*. 6(3):10-17.

Primary References

Abran, A. and J.W. Moore (exec. eds); P. Bourque and R. Dupuis (eds.). 2004. *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide)*. Piscataway, NJ, USA: The Institute of Electrical and Electronic Engineers, Inc. (IEEE). Available at: http://www.computer.org/portal/web/swebok.

Fairley, R.E. 2009. *Managing and Leading Software Projects*. Hoboken, NJ, USA: John Wiley & Sons.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

Brooks, F.P. 1995. *The Mythical Man-Month,* Anniversary Edition. New York, NY, USA: Addison Wesley.

< Previous Article | Next Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIHF
END_ENCODED_CONTENT

References

[1] http://www.computer.org/portal/web/swebok/html/ch8#Ref2.3
[2] http://www.computer.org/portal/web/swebok/html/ch9#Ref2.1
[3] http://www.computer.org/portal/web/swebok/html/contentsch5#ch5
[4] http://www.computer.org/portal/web/swebok/html/contentsch9#ch9
[5] http://www.computer.org/portal/web/swebok/html/ch8#Ref6
[6] http://leadinganswers.typepad.com/leading_answers/files/original_waterfall_paper_winston_royce.pdf

Systems Engineering and Project Management

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 Relationships

References

Works Cited

None.

Primary References

Fairley, R.E. 2009. *Managing and Leading Software Projects*. Hoboken, NJ, USA: John Wiley & Sons.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management*, 3rd ed. New York, NY, USA: John Wiley & Sons.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKi8KICAgIHZhciBkaXNxdXNfY29uZmlnID0gZnVuY3Rpb24gKCkgewogICAgICAgIHRoaXMucGFnZS5pZGVudGlmaWVyID0gIlRoZSBOYXR1cmUgb2YgUHJvamVjdCBNYW5hZ2VtZW50IjsKICAgICAgICB0aGlzLnBhZ2UudXJsID0gImh0dHA6Ly9zZWJvay53aWtpc3BhY2VzLmNvbS9UaGUgTmF0dXJlIG9mIFByb2plY3QgTWFuYWdlbWVudCI7CiAgICB9OwogICAgKGZ1bmN0aW9uKCkgewogICAgICAgIHZhciBkID0gZG9jdW1lbnQsIHMgPSBkLmNyZWF0ZUVsZW1lbnQoInNjcmlwdCIpOwogICAgICAgIHMudHlwZSA9ICJ0ZXh0L2phdmFzY3JpcHQiOyBzLmFzeW5jID0gdHJ1ZTsKICAgICAgICBzLnNyYyA9ICIvL2Rpc3F1cy5jb20vZW1iZWQuanMiOwogICAgICAgIChkLmdldEVsZW1lbnRzQnlUYWdOYW1lKCJoZWFkIilbMF0gfHwgZC5nZXRFbGVtZW50c0J5VGFnTmFtZSgiYm9keSIpWzBdKS5hcHBlbmRDaGlsZChzKTsKICAgIH0oKSk7CjwvZ2VuZXJhdGVkIGJ5IGNvbnRlbnQgZnJvbSBodHRwOi8vd3d3LmRpc3F1cy5jb20vPgo=
END_ENCODED_CONTENT

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

Boehm, B., C. Abts., A.W. Brown, S. Chulani, B.K. Clark, E. Horowitz, R. Madachy, D. Reifer, and B. Steece. 2000. *Software Cost Estimation with COCOMO II.* Upper Saddle River, NJ, USA: Prentice Hall.

Chapman, C., and S. Ward. 2003. *Project Risk Management: Processes, Techniques and Insights.* Chichester, West Sussex, England, UK: John Wiley & Sons.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken NJ, USA: John Wiley & Sons.

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management.* Hoboken, NJ, USA: John Wiley & Sons.

Hillson, David. 2003. *Effective Opportunity Management for Projects: Exploiting Positive Risk.* Boca Raton, FL, USA: CRC Press.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities,* version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2.

ISO/IEC/IEEE. 2009. ISO/IEC/IEEE 16326:2009(E). *Systems and Software Engineering - Life Cycle Processes - Project Management*. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC)/Institute of Electrical and Electronics Engineers (IEEE).

NASA. 2007. *Systems Engineering Handbook*. Washington, DC, USA: National Aeronautics and Space Administration.

Olsson, Rolf. 2007. "In search of opportunity management: Is the risk management process enough?" *International Journal of Project Management*, 25 (8), 745–752, 2011.

Stewart, Rodney. 1990. *Cost Estimating.* New York, NY, USA: Wiley.

Valerdi, R. *The Constructive Systems Engineering Cost Model (COSYSMO): Quantifying the Costs of Systems Engineering Effort.* Saarbrucken, Germany: VDM Verlag.

Primary References

Fairley, R.E. 2009. *Managing and Leading Software Projects.* Hoboken, NJ, USA: John Wiley & Sons.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*. 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

Blanchard, B. 2008. *System Engineering Management.* Hoboken, NJ, USA: John Wiley & Sons.

Kerzner, Harold. 2003. *Project Management: A Systems Approach to Planning, Scheduling, and Controlling*, 8th ed. Hoboken, NJ, USA: John Wiley & Sons.

Martin, J. 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products.* London, UK: Taylor and Francis Group CRC-Press, LLC.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKiBUaGV
END_ENCODED_CONTENT

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

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Primary References

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

Blanchard, B. 2008. *System Engineering Management*. Hoboken, NJ, USA: John Wiley & Sons.

Fairley, R.E. 2009. *Managing and Leading Software Projects*. Hoboken, NJ, USA: John Wiley & Sons.

Martin, J. 1997. *Systems Engineering Guidebook: A Process for Developing Systems and Products*. London, UK: Taylor and Francis Group CRC-Press, LLC.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20]**.**

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICog
END_ENCODED_CONTENT

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.

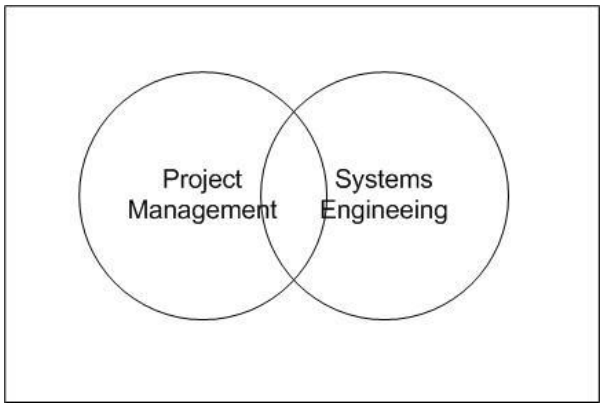


Figure 1. Overlap of PM and SE. (SEBoK Original)

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.

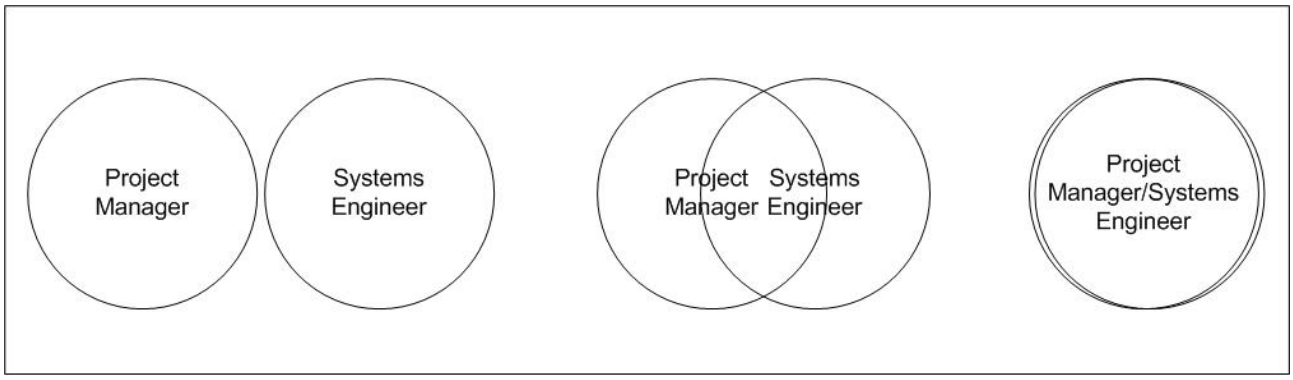


Figure 2. Overlap of Project Roles. (SEBoK Original)

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

CMMI. 2011. *CMMI for Development: Guidelines for Process Integration and Product Improvement*. Old Tappan, NJ, USA: Pearson Education.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Primary References

Chrissis, M.B, M. Konrad, S. Shrum. 2011. *CMMI for Development: Guidelines for Process Integration and Product Improvement*, 3rd ed. Boston, MA, USA: Addison-Wesley Professional.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Additional References

NASA. 2007. *Systems Engineering Handbook*, Revision 1. Washington, DC, USA: National Aeronautics and Space Administration (NASA). NASA/SP-2007-6105.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20]**.**

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKiBSRUQ
END_ENCODED_CONTENT

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.

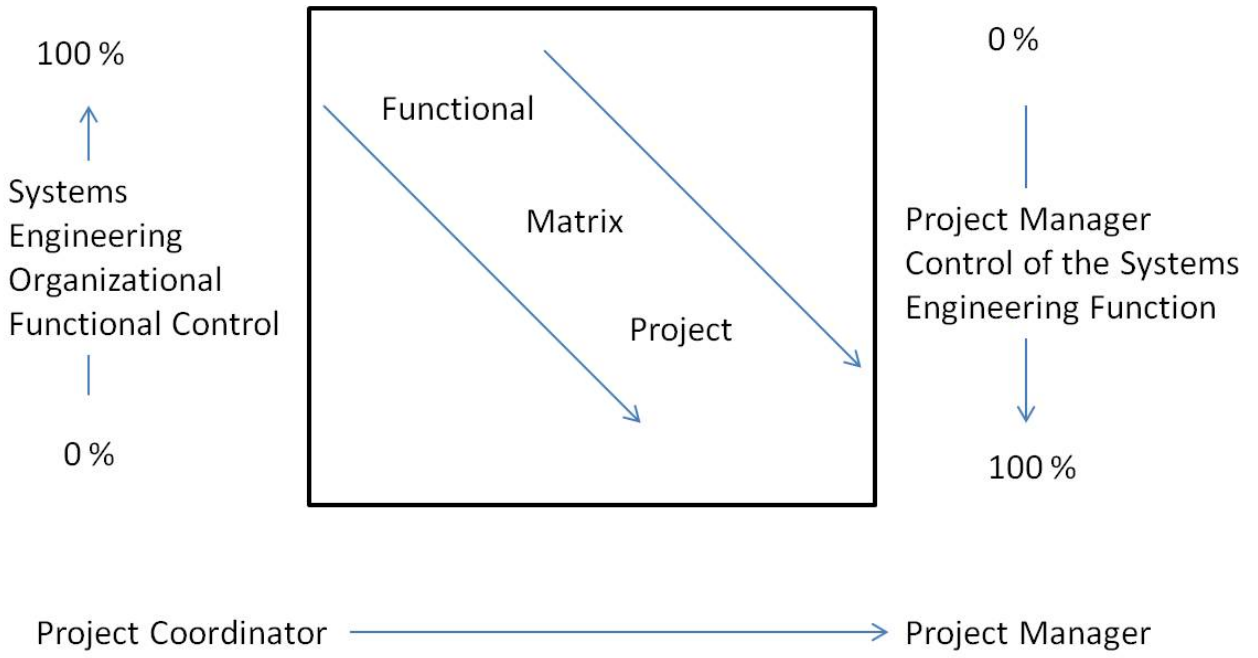


Figure 1. The Organizational Continuum (2). (SEBoK Original and Adapted from Fairley 2009). Reprinted with permission of the IEEE Computer Society. All other rights are reserved by the copyright owner.

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.

References

Works Cited

Blanchard, B.S., and W. J. Fabrycky. 2011. *Systems Engineering and Analysis.* 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Fairley, R.E. 2009. *Managing and Leading Software Projects.* IEEE Computer Society, John Wiley & Sons, Inc. Publication. ISBN: 978-0-470-29455-0.

PMI. 2013. *A Guide to the Project Management Body of Knowledge (PMBOK® Guide)*, 5th ed. Newtown Square, PA, USA: Project Management Institute (PMI).

Primary References

Forsberg, K., H. Mooz, and H. Cotterman. 2005. *Visualizing Project Management.* 3rd ed. New York, NY, USA: John Wiley & Sons.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKiBSRUQ
END_ENCODED_CONTENT

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
- staffing, organizing, directing
- cost, schedule, and performance monitoring
- risk monitoring and control
- operations planning and preparation

- operations management

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

IIE. 1992. *Industrial Engineering Terminology*, revised ddition. Norwood, GA, USA: Institute of Industrial Engineers (IIE). Accessed March 7, 2012. Available: http://www.iienet2.org/Details.aspx?id=645.

Salvendy, G. (ed.) 2001. *Handbook of Industrial Engineering, Technology and Operations Management,* 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Turner, W.C., J.H. Mize, K.E. Case, and J.W. Nazemtz. 1992. *Introduction To Industrial And Systems Engineering*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall.

Zandin, K.B. (ed.) 2001. *Maynard's Industrial Engineering Handbook*, 5th ed. New York, NY, USA: McGraw-Hill.

Primary References

IIE. 1992. *Industrial Engineering Terminology*, revised ddition. Norwood, GA, USA: Institute of Industrial Engineers (IIE). Accessed March 7, 2012. Available: http://www.iienet2.org/Details.aspx?id=645.

Salvendy, G. (ed.) 2001. *Handbook of Industrial Engineering, Technology and Operations Management,* 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Inc.

Zandin, K.B. (ed.) 2001. *Maynard's Industrial Engineering Handbook,* 5th ed. New York, NY, USA: McGraw-Hill.

Additional References

Operations Engineering

Hopp, W., and M. Spearman. 2001. *Factory Physics,* 3rd ed., New York, NY, USA: McGraw-Hill.

Heizer, J., and B. Render. 2001. *Operations Management,* 6th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Mantel, S., J. Meredith, S. Shafer, and M. Sutton. 2008. *Project Management in Practice.* New York, NY, USA: John Wiley & Sons.

Operations Research

Banks, J., J. Carson, B. Nelson, and D. Nicol. 2005. *Discrete-Event System Simulation,* 4th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Hillier, F., and G. Lieberman. 2010. *Introduction to Operations Research,* 9th ed. New York, NY, USA: McGraw Hill.

Kelton, W. David, R. Sadowski, and D. Sturrock. 2006. *Simulation with Arena,* 4th ed. New York, NY, USA: McGraw-Hill.

Law, A. 2007. *Simulation Modelling and Analysis,* 4th ed. New York, NY, USA: McGraw-Hill.

Winston, W. and J. Goldberg. 2004. *Operations Research Applications & Algorithms,* Independence, KY, USA: Thomson Brooks/Cole.

Production Engineering / Work Design

Freivalds, A. 2009. *Niebel's Methods, Standards, and Work Design,* 12th ed. New York, NY, USA: McGraw-Hill.

Groover, M. 2007 *Work Systems: The Methods, Measurement, and Management of Work,* Upper Saddle River, NJ, USA: Pearson-Prentice Hall.

Grover, M. 2007. *Fundamentals of Modern Manufacturing,* 3rd ed. New York, NY, USA: John Wiley & Sons.

Konz, S., and S. Johnson, 2008. *Work Design: Occupational Ergonomics,* 7th ed. Scottsdale, AZ, USA: Holcomb Hathaway.

Meyers, F., and J. Stewart, 2001 *Motion and Time Study for Lean Manufacturing,* 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall.

Facilities Engineering and Energy Management

Garcia-Diaz, A., and J. MacGregor Smith. 2008. *Facilities Planning and Design,* Upper Saddle River, NJ, USA: Pearson-Prentice Hall.

Tompkins, J., J. White, Y. Bozer, and J. Tanchoco. 2003. *Facilities Planning,* 3rd ed. New York, NY, USA: John Wiley & Sons.

Ergonomics

Chaffin, D., and G. Andersson. 1991. *Occupational Biomechanics.* New York, NY, USA: John Wiley & Sons.

Wickens, C., S. Gordon, and Y. Liu. 2004. *An Introduction to Human factors Engineering.* Upper Saddle River, NJ, USA: Pearson-Prentice Hall.

Engineering Economic Analysis

Blank, L.T., and A.J. Tarquin. 2011. *Engineering Economy,* 7th ed. New York, NY, USA: McGraw-Hill.

Newnan, D., T. Eschenbach, and J. Lavelle. 2011. *Engineering Economic Analysis,* 11th ed. New York, NY, USA: Oxford University Press.

Parl, C. 2007. *Fundamentals of Engineering Economics.* Upper Saddle River, NJ, USA: Prentice Hall.

Thuesen, G., and W. Fabrycky. 2001. *Engineering Economy,* 9th ed. Upper Saddle River, NJ, USA: Prentice Hall.

Quality & Reliability

Ebeling, C.E. 2005. *An Introduction to Reliability and Maintainability Engineering.* Long Grove, IL, USA: Waveland Press, Inc.

Hawkins, D., and D. Olwell. 1998. *Cumulative Sum Chars and Charting for Quality Improvement.* New York, NY, USA: Springer.

Kiemele, M., S. Schmidt, and R. Berdine. 1999. *Basic Statistics: Tools for Continuous Improvement*, 4th ed. Colorado Springs, CO, USA: Air Academy Press.

Montgomery, D., and G. Runger. 2007. *Applied Statistics and Probability for Engineers*, 4th ed. Hoboken, NJ, USA: John Wiley & Sons.

Montgomery, D. 2013. *Design & Analysis of Experiments,* 8th ed. Hoboken, NJ, USA: John Wiley & Sons.

Montgomery, D. 2009. *Introduction to Statistical Quality Control*, 6th ed. Hoboken, NJ, USA: John Wiley & Sons.

Quality Staff. 2006. *Data Quality Assessment: Statistical Methods for Practitioners.* Washington, DC, USA: Environmental Protection Agency (EPA).

Engineering Management

Gido, J., and J. Clements. 2009. *Successful Project Management.* Cincinnati, OH, USA: South Western.

Kersner, H. 2009. *A Systems Approach to Planning, Scheduling, and Controlling,* 10th ed. New York, NY, USA: John Wiley & Sons.

Supply Chain Management

Jacobs, F., and R. Chase. 2010. *Operations and Supply Chain Management.* New York, NY, USA: McGraw-Hill.

Mentzer, J. 2004. *Fundamentals of Supply Chain Management: Twelve Drivers of Competitive Advantage.* Thousand Oaks, CA, USA: Sage.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20]**.**

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKi
END_ENCODED_CONTENT

Systems Engineering and Procurement/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.

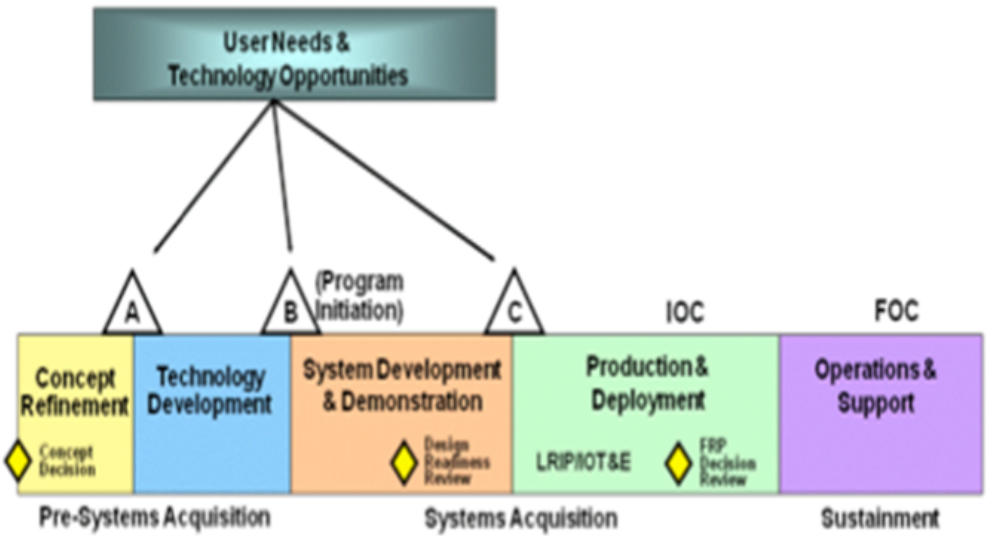


Figure 1. An Acquisition Process Model (DAU 2010). Released by Defense Acquisition University (DAU)/U.S. Department of Defense (DoD).

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.

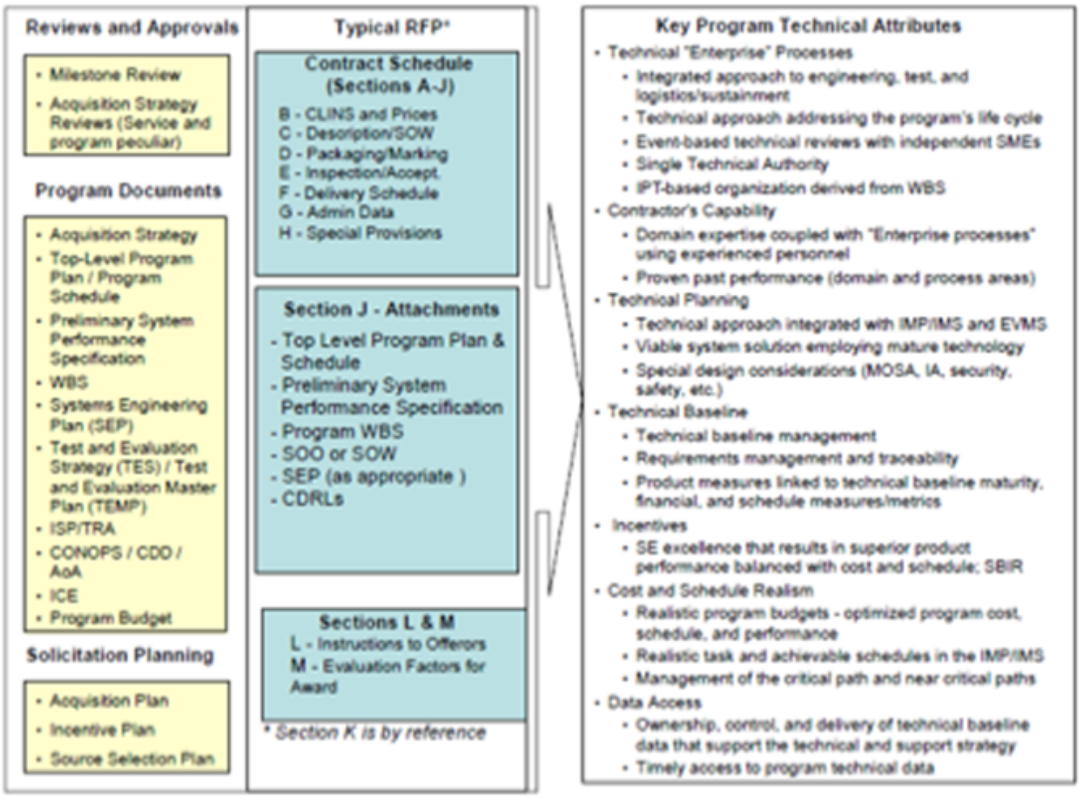


Figure 2. Relating Acquisition to Request for Proposal and Technical Attributes. (DoD 2006a). Released by the U.S. Office of the Secretary of Defense.

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

contract(s).

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.

Typical Contract-Related Activities	System Engineer and Project Manager Roles
1. Identify overall procurement requirements and associated budget. Describe the offer's needs and any constraints on the procurement.	Lead system engineer (LSE) provides program technical requirements. PM provides any programmatic related requirements.
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.	LSE defines the technical strategy/approach and required technical efforts. This should be consistent with the program's Acquisition Strategy.
3. Document market research results and identify potential industry sources.	PM and LSE identify programmatic and technical information needed and assist in evaluating the results.
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.	PM and LSE ensure the specific programmatic and technical needs are defined clearly (e.g., commercial-off-the-shelf (COTS) products).
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.	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.
6. Plan the requirements for the contract Statement of Objectives (SOO) / Statement of Work (SOW) / specification, project technical reviews, acceptance requirements, and schedule.	LSE is responsible for the development of the technical aspects of the SOO/SOW.
7. Plan and conduct Industry Days as appropriate.	PM and LSE supports the CO in planning the meeting agenda to ensure technical needs are discussed.
8. Establish contract cost, schedule, and performance reporting requirements. Determine an incentive strategy and appropriate mechanism (e.g., Award Fee Plan and criteria).	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.
9. Identify data requirements.	LSE identifies all technical Contractor Data Requirements List (CDRL) and technical performance expectations.
10. Establish warranty requirements, if applicable.	LSE works with the CO to determine cost-effective warranty requirements.
11. Prepare a Source Selection Plan (SSP) and RFP (for competitive contracts).	PM and LSE provide input to the SSP per the SOO/SOW.
12. Conduct source selection and award the contract to the successful offeror.	PM and LSE participate on evaluation teams.

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 outsource 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

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense (DoD).

DoD. 2011. *Systems Engineering Plan (SEP) Outline.* Washington, DC, USA: Office of the Undersecretary of Defense for Acquisition, Transportation, and Logistics (AT&L), US Department of Defense (DoD).

DoD. 2006. *Guide for Integrating Systems Engineering into DoD Acquisition Contracts.* Washington, DC, USA: Office of the Undersecretary of Defense for Acquisition, Transportation, and Logistics (AT&L), US Department of Defense (DoD).

DoD. 2001. *Systems Engineering Fundamentals.* Washington, DC, USA: Defense Acquisition University Press/US Department of Defense.

Primary References

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/US Department of Defense (DoD).

DoD. 2011. *Systems Engineering Plan (SEP) Outline.* Washington, DC, USA: Office of the Undersecretary of Defense for Acquisition, Transportation, and Logistics (AT&L), US Department of Defense (DoD).

DoD. 2006. *Guide for Integrating Systems Engineering into DoD Acquisition Contracts.* Washington, DC, USA: Office of the Undersecretary of Defense for Acquisition, Transportation, and Logistics (AT&L), US Department of Defense (DoD).

Additional References

MITRE. 2011. "Acquisition Systems Engineering." *Systems Engineering Guide.* Accessed March 9, 2012. Available: http://www.mitre.org/work/systems_engineering/guide/acquisition_systems_engineering/.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKg
END_ENCODED_CONTENT

Systems Engineering and Specialty Engineering

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.

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:

- Integration of Specialty Engineering
- Reliability, Availability, and Maintainability
- Human Systems Integration
- Safety Engineering
- Security Engineering
- System Assurance
- Electromagnetic Interference/Electromagnetic Compatibility
- Resilience Engineering
- Manufacturability and Producibility
- Affordability
- Environmental 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.

References

Works Cited

None.

Primary References

None.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogK
END_ENCODED_CONTENT

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.

Integration Process for Specialty Engineering

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.

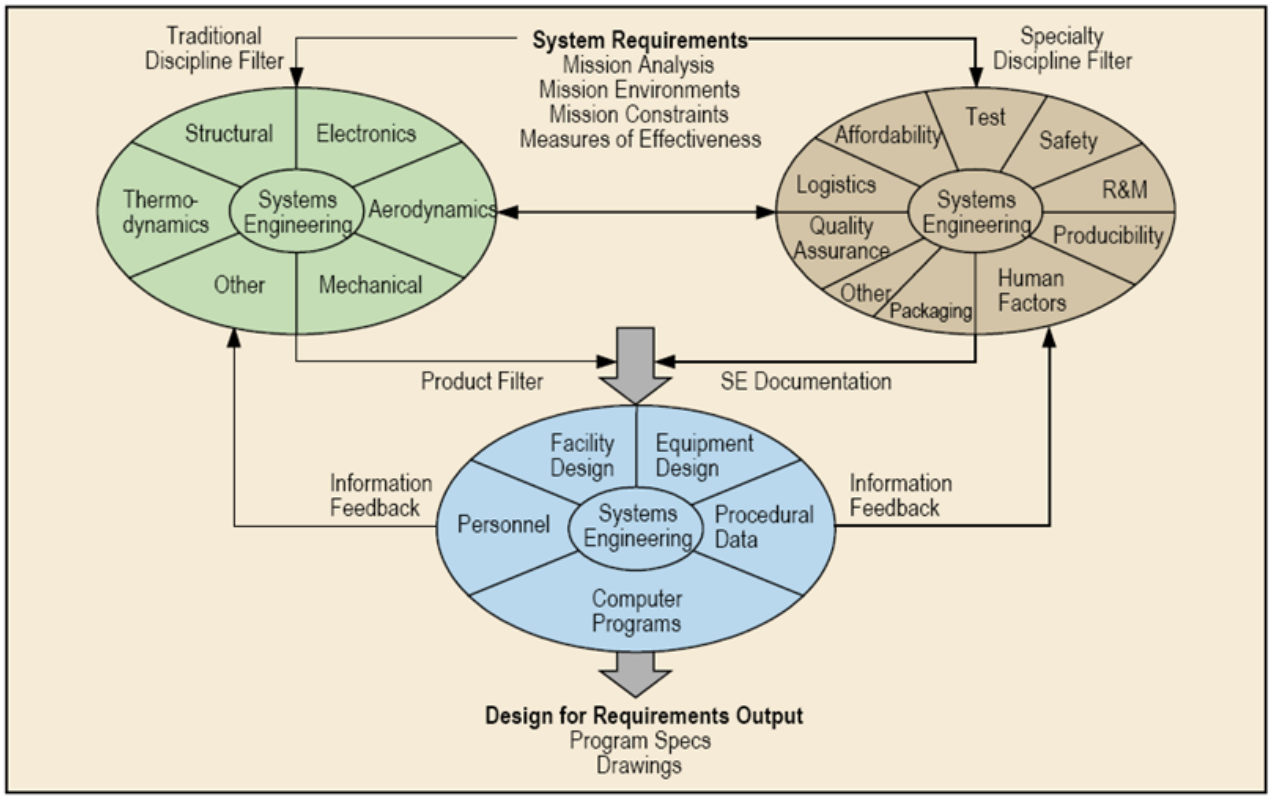


Figure 1. Integration Process for Specialty Engineering (USAF 2000). Released by the U.S. Air Force.

References

Works Cited

USAF. 2000. *Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems Command and Control Systems Management Information Systems*, version 3.0. Hill AFB: Department of the Air Force Software Technology Support Center. May 2000. Accessed on September 11, 2011. Available at http://www.stsc.hill.af.mil/resources/tech%5Fdocs/.

Primary References

USAF. 2000. *Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems Command and Control Systems Management Information Systems*, version 3.0. Hill AFB: Department of the Air Force Software Technology Support Center. May 2000. Accessed on September 11, 2011. Available at http://www.stsc.hill.af.mil/resources/tech%5Fdocs/.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogK
END_ENCODED_CONTENT

Reliability, Availability, and Maintainability

<html> <meta name="citation_title" content="Reliability, Availability, and Maintainability"> <meta name="citation_author" content="Pyster, Art"> <meta name="citation_author" content="Olwell, David H."> <meta name="citation_author" content="Hutchison, Nicole"> <meta name="citation_author" content="Enck, Stephanie"> <meta name="citation_author" content="Anthony, James F., Jr."> <meta name="citation_author" content="Henry, Devanandham"> <meta name="citation_author" content="Squires, Alice (eds)"> <meta name="citation_publication_date" content="2012/11/30"> <meta name="citation_journal_title" content="Guide to the Systems Engineering Body of Knowledge (SEBoK)"> <meta name="citation_volume" content="version 1.0.1"> <meta name="citation_pdf_url" content="http://www.sebokwiki.org/1.0.1/index.php?title=Main_Page"></html>

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.

This article focuses primarily on the reliability of physical system elements. Software reliability is a separate discipline. Readers interested in software reliability should refer to the IEEE Std 1633 (IEEE 2008).

Probability Models for Populations

Reliability is 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. Different failure mechanisms are referred to as failure modes and can be modeled separately or aggregated into a single failure model.

Let T be a random time to failure. Reliability can be thought of as the complement of the cumulative distribution function (CDF) for T for a given set of environmental conditions e:

$$R(t \mid e) = 1 - F(t \mid e) = \mathrm{P}(T > t \mid e)$$

Maintainability is 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 is the 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).

Each of these probability models is usually specified by a continuous, non-negative distribution. Typical distributions used in practice include exponential (possibly with a threshold parameter), Weibull (possibly with a threshold parameter), log-normal, and generalized gamma.

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.

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, biased, 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.

Design Issues

System requirements should include specifications for reliability, maintainability, and availability, and each should be conditioned on the projected operating environments.

A proposed design should be analyzed prior to development to estimate whether or not it will meet those specifications. This is usually done by assuming historical data on actual or similar components represents the future performance of the components for the proposed system. If no data is available, conservative engineering judgment is often applied. The system dependency on the reliability of its components can be captured in several ways, including reliability block diagrams, fault trees, and failure mode effects and criticality analyses (FMECA) (Kececioglu 1991).

If a proposed design does not meet the preliminary RAM specifications, it can be adjusted. Critical failures are mitigated so that the overall risk is reduced to acceptable levels. This can be done in several ways:

1. **Fault tolerance** is a strategy that seeks to make the system robust against the failure of a component. This can be done by introducing redundancy. Redundant units can operate in a *stand-by* mode. A second tolerance strategy is to have the redundant components share the load, so that even if one or more of them fail the system continues to operate. There are modeling issues associated with redundancy, including switching between components, warm-up, and increased failure rates for surviving units under increased load when another load-sharing unit fails. Redundancy can be an expensive strategy as there are cost, weight, volume, and power penalties associated with stand-by components.
2. **Fault avoidance** seeks to improve individual components so that they are more reliable. This can also be an expensive strategy, but it avoids the switching issues, power, weight, and volume penalties associated with using redundant components.
3. A third strategy is to repair or replace a component following a **preventive maintenance** schedule. This requires the assumption that the repair returns the component to "good as new" status, or possibly to an earlier age-equivalent. These assumptions can cause difficulties; for example, an oil change on a vehicle does not return the engine to 'good as new' status. Scheduled replacement can return a unit to good as new, but at the cost of wasting potential life for the replaced unit. As a result, the selection of a replacement period is a non-linear optimization problem that minimizes total expected life-cycle costs. These costs are the sum of the expected costs of planned and unplanned maintenance actions.
4. A fourth strategy is to **control the environment** so that a system is not operated under conditions that accelerate the aging of its components.

Any or all of the above strategies (fault tolerance, fault avoidance, preventive maintenance, and environmental control) may be applied to improve the designed reliability of a system.

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.

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.

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.

System Metrics

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.

System Models

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.

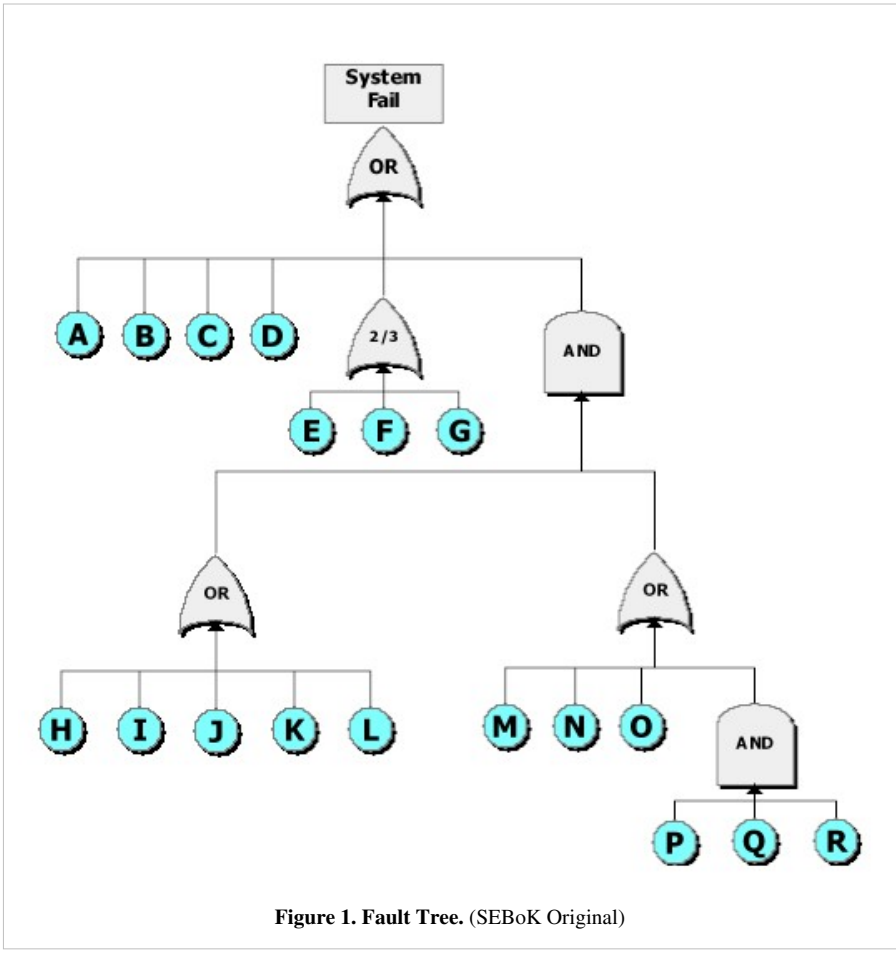


Figure 1. Fault Tree. (SEBoK Original)

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.

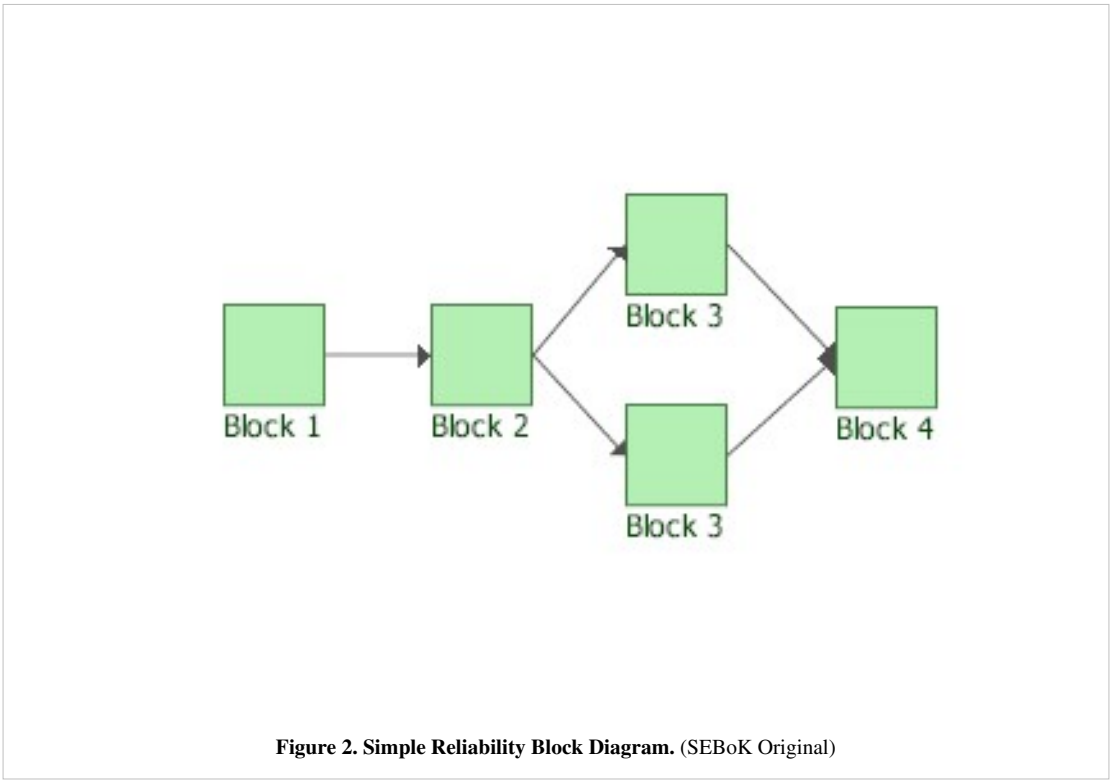


Figure 2. Simple Reliability Block Diagram. (SEBoK Original)

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.

Software 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.

References

Works Cited

American Society for Quality (ASQ). 2011. *Glossary: Reliability.* Accessed on September 11, 2011. Available at http://asq.org/glossary/r.html.

IEEE. 2008. IEEE Recommended Practice on Software Reliability. New York, NY, USA: Institute of Electrical and Electronic Engineers (IEEE). IEEE Std 1633-2008.

Kececioglu, D. 1991. *Reliability Engineering Handbook,* Volume 2. Upper Saddle River, NJ, USA: Prentice Hall.

Meeker, W.Q. and L.A. Escobar. 1998. Statistical Methods for Reliability Data. New York, NY, USA: Wiley and Sons.

Nelson, Wayne. 1990. Accelerated Testing: Statistical Models, Test Plans, and Data Analysis. New York, NY, USA: Wiley and Sons.

ReliaSoft. 2007. *Failure Modes and Effects Analysis (FMEA) and Failure Modes, Effects and Criticality Analysis (FMECA).* Accessed on September 11, 2011. Available at http://www.weibull.com/basics/fmea.htm.

Primary References

Blischke, W.R. and D.N. Prabhakar Murthy. 2000. *Reliability Modeling, Prediction, and Optimization.* New York, NY, USA: Wiley and Sons.

DoD. 2005. *DOD Guide for Achieving Reliability, Availability, and Maintainability.* Arlington, VA, USA: U.S. Department of Defense (DoD). Accessed on September 11, 2011. Available at: http://www.acq.osd.mil/se/docs/RAM_Guide_080305.pdf [1]

Kececioglu, D. 1991. *Reliability Engineering Handbook,* Volume 2. Upper Saddle River, NJ, USA: Prentice Hall.

Lawless, J.F. 1982. *Statistical Models and Methods for Lifetime Data.* New York, NY, USA: Wiley and Sons.

Martz, H.F. and R.A. Waller. 1991. *Bayesian Reliability Analysis.* Malabar, FL, USA: Kreiger.

Meeker, W.Q. and L.A. Escobar. 1998. *Statistical Methods for Reliability Data.* New York, NY, USA: Wiley and Sons.

Additional References

Olwell, D.H. 2011. "Reliability Leadership." *Proceedings of the 2001 Reliability and Maintainability M Symposium.* Philadelphia, PA, USA: IEEE. Accessed 7 March 2012 at [IEEE web site. [2]]

ReliaSoft. 2007. "Availability." Accessed on September 11, 2011. Available at: http:/ / www. weibull. com/ SystemRelWeb/availability.htm.

SAE. 2000a. *Aerospace Recommended Practice ARP5580: Recommended Failure Modes and Effects Analysis (FMEA) Practices for Non-Automobile Applications.* Warrendale, PA, USA: Society of Automotive Engineers (SAE) International.

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.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogK
END_ENCODED_CONTENT

References

[1] http://www.acq.osd.mil/se/docs/RAM_Guide_080305.pdf
[2] http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=902456

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.

HSI Domains

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).
- 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.

- 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).
- 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 ladders, handholds, maintenance provisions, etc.) (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).
- 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.
- 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).
- 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).

References

Works Cited

Booher, H.R. (ed.). 2003. *Handbook of Human Systems Integration.* Hoboken, NJ, USA: Wiley.

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

US Air Force. 2009. *Air Force Human Systems Integration Handbook.* Brooks City-Base, TX, USA: Directorate of Human Performance Integration. Available at http:/ / www. wpafb. af. mil/ shared/ media/ document/ AFD-090121-054.pdf. [1]

Primary References

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Additional References

Blanchard, B. S., and W. J. Fabrycky. 2011. *Systems Engineering and Analysis*. 5th ed. Prentice-Hall International series in Industrial and Systems Engineering. Englewood Cliffs, NJ, USA: Prentice-Hall.

Helander, Martin, Landauer, T.K, and Prabhu, P.V. 1997. *Handbook of Human-Computer Interaction.* Amsterdam, Netherlands: Elsevier.

Pew, R.W. and A.S. Mavor. 2007. *Human-System Integration in the System Development Process: A New Look.* Washington, DC, USA: National Academies Press.

Wickens, C.D., Lee, J. D, Liu, Y., and Becker, S.E. Gordon. 2004. *An Introduction to Human Factors Engineering.* Englewood Cliffs, NJ, USA: Prentice-Hall.

Woodson, W.E, Tillman, B. and Tillman, P. 1992. "Human Factors Design Handbook: Information and Guidelines for the Design of Systems, Facilities, Equipment, and Products for Human Use." 2nd Ed. New York, NY, USA: McGraw Hill.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICog
END_ENCODED_CONTENT

References

[1] http://www.wpafb.af.mil/shared/media/document/AFD-090121-054.pdf

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."

Hazards

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 Safety Personnel

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.*

(USAF 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 1. Safety Ontology. (SEBoK Original)

Ontology Element Name	Ontology Element Attributes	Relationships to Safety
Failure modes	Manner of failure	Required attribute
Severity	Consequences of failure	Required attribute
Criticality	Impact of failure	Required attribute
Hazard Identification	Identification of potential failure modes	Required to determine failure modes
Risk	Probability of a failure occurring	Required attribute
Mitigation	Measure to take corrective action	Necessary to determine criticality and severity

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.

References

Works Cited

DoD. 2012. *Standard practice for System Safety.* Arlington, VA, USA: Department of Defense (DoD). MIL-STD 882E. Accessed 4 November 2014 at http://assistdoc1.dla.mil/qsDocDetails.aspx?ident_number=36027

Roland, H.E. and B. Moriarty. 1990. *System Safety Engineering and Management.* Hoboken, NJ, USA: Wiley-IEEE.

USAF. 1998. *The US Air Force Mishap Prevention Program.* Washington, DC, USA: US Air Force, Air Force Instruction (AFI).

Primary References

None.

Additional References

Bahr, N. J. 2001. "System Safety Engineering and Risk Assessment." In *International Encyclopedia of Ergonomics and Human Factors.* Vol. 3. Ed. Karwowski, Waldemar. New York, NY, USA: Taylor and Francis.

ISSS. "System Safety Hazard Analysis Report." The International System Safety Society (ISSS). DI-SAFT-80101B. http://www.system-safety.org/Documents/DI-SAFT-80101B_SSHAR.DOC.

ISSS. "Safety Assessment Report." The International System Safety Society (ISSS). DI-SAFT-80102B. http://www.system-safety.org/Documents/DI-SAFT-80102B_SAR.DOC.

ISSS. "Engineering Change Proposal System Safety Report." The International System Safety Society (ISSS). DI-SAFT-80103B. http://www.system-safety.org/Documents/DI-SAFT-80103B_ECPSSR.DOC.

ISSS. "Waiver or Deviation System Safety Report." The International System Safety Society (ISSS). DI-SAFT-80104B. http://www.system-safety.org/Documents/DI-SAFT-80104B_WDSSR.DOC.

ISSS. "System Safety Program Progress Report." The International System Safety Society (ISSS). DI-SAFT-80105B. http://www.system-safety.org/Documents/DI-SAFT-80105B_SSPPR.DOC.

ISSS. "Health Hazard Assessment Report." The International System Safety Society (ISSS). DI-SAFT-80106B. http://www.system-safety.org/Documents/DI-SAFT-80106B_HHAR.DOC.

ISSS. "Explosive Ordnance Disposal Data." The International System Safety Society (ISSS). DI-SAFT-80931B. http://www.system-safety.org/Documents/DI-SAFT-80931B_EODD.pdf.

ISSS. "Explosive Hazard Classification Data." The International System Safety Society (ISSS). DI-SAFT-81299B. http://www.system-safety.org/Documents/DI-SAFT-81299B_EHCD.pdf.

ISSS. "System Safety Program Plan (SSPP)." The International System Safety Society (ISSS). DI-SAFT-81626. http://www.system-safety.org/Documents/DI-SAFT-81626_SSPP.pdf.

ISSS. "Mishap Risk Assessment Report." The International System Safety Society (ISSS). DI-SAFT-81300A. http://www.system-safety.org/Documents/DI-SAFT-81300A_MRAR.DOC.

Joint Software System Safety Committee. 1999. *Software System Safety Handbook.* Accessed 7 March 2012 at http://www.system-safety.org/Documents/Software_System_Safety_Handbook.pdf.

Leveson, N. 2011. Engineering a safer world: systems thinking applied to safety. Cambridge, Mass: MIT Press.

Leveson, N. G. 2012. “Complexity and Safety.” In *Complex Systems Design & Management,* ed. Omar Hammami, Daniel Krob, and Jean-Luc Voirin, 27–39. Springer Berlin Heidelberg. http://dx.doi.org/10.1007/978-3-642-25203-7_2.

NASA. 2004. *NASA Software Safety Guidebook.* Accessed 7 March 2012 at [[1]].

Roland, H. E., and Moriarty, B. 1985. *System Safety Engineering and Management.* New York, NY, USA: John Wiley.

SAE. 1996. *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.* ARP 4761. Warrendale, PA, USA: Society of Automotive Engineers. Accessed 28 August 2012 at [http://standards.sae.org/arp4761/ [2]].

SAE. 1996. *Certification Considerations for Highly-Integrated Or Complex Aircraft Systems.* ARP 4754. Warrendale, PA, USA: Society of Automotive Engineers. Accessed 28 August 2012 at [http://standards.sae.org/arp4754/ [3]].

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKiBSRU1PVkUgVEhJUyBCTE9DSyBXSEVOIFVTSU5HIFRISVMgVEVNUExBVEUgSU4gQ09NUE9TRVIgKiBMZWFybiBtb3JlIGFib3V0IGNvbmZpZ3VyaW5nIERpc3F1cyBmb3IgeW91ciBzaXRlOiBodHRwczovL2RvY3MuZGlzcXVzLmNvbS9kZXZlbG9wZXJzLmh0bWwgKi8KICAgIHZhciBkaXNxdXNfY29uZmlnID0gZnVuY3Rpb24gKCkgewogICAgICAgIHRoaXMucGFnZS51cmwgPSAnaHR0cDovL3d3dy5zZWJva3dpa2kub3JnL3dpa2kvU2VjdXJpdHlfRW5naW5lZXJpbmcnOyAgLy8gUmVwbGFjZSBQQUdFX1VSTCB3aXRoIHlvdXIgcGFnZSdzIGNhbm9uaWNhbCBVUkwgdmFyaWFibGUKICAgICAgICB0aGlzLnBhZ2UuaWRlbnRpZmllciA9ICdTZWN1cml0eV9FbmdpbmVlcmluZyc7IC8vIFJlcGxhY2UgUEFHRV9JREVOVElGSUVSIHdpdGggeW91ciBwYWdlJ3MgdW5pcXVlIGlkZW50aWZpZXIgdmFyaWFibGUKICAgIH07CiAgICAoZnVuY3Rpb24oKSB7ICAvLyBET05UIEVESVQgQkVMT1cgVEhJUyBMSU5FCiAgICAgICAgdmFyIGQgPSBkb2N1bWVudCwgcyA9IGQuY3JlYXRlRWxlbWVudCgnc2NyaXB0Jyk7CiAgICAgICAgcy5zcmMgPSAnaHR0cHM6Ly9zZWJva3dpa2kuZGlzcXVzLmNvbS9lbWJlZC5qcyc7CiAgICAgICAgcy5zZXRBdHRyaWJ1dGUoJ2RhdGEtdGltZXN0YW1wJywgK25ldyBEYXRlKCkpOwogICAgICAgIChkLmhlYWQgfHwgZC5ib2R5KS5hcHBlbmRDaGlsZChzKTsKICAgIH0pKCk7CjwvcmVhbHNjcmlwdD4KPG5vc2NyaXB0PlBsZWFzZSBlbmFibGUgSmF2YVNjcmlwdCB0byB2aWV3IHRoZSA8YSBocmVmPSJodHRwczovL2Rpc3F1cy5jb20vP3JlZj1zZWJva3dpa2kiPmNvbW1lbnRzIHBvd2VyZWQgYnkgRGlzcXVzLjwvYT48L25vc2NyaXB0Pgo=
END_ENCODED_CONTENT

References

[1] http://www.hq.nasa.gov/office/codeq/doctree/871913.pdf
[2] http://standards.sae.org/arp4761/
[3] http://standards.sae.org/arp4754/

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).

Multidisciplinary Reach

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.

Robust Security Design

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.

Defense Application

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.

References

Works Cited

Baldwin, K., J. Miller, P. Popick, and J. Goodnight. 2012. *The United States Department of Defense Revitalization of System Security Engineering Through Program Protection.* Proceedings of the 2012 IEEE Systems Conference, 19-22 March 2012, Vancouver, BC, Canada. Accessed 28 August 2012 at http://www.acq.osd.mil/se/docs/IEEE-SSE-Paper-02152012-Bkmarks.pdf [1].

DODI5200.44, United States Department of Defense, *Protection of Mission Critical Functions to Achieve Trusted Systems and Networks*, Department of Defense Instruction Number 5200.44, November 2012, Accessed 3 November 2014 at Defense Technical Information Center http://www.dtic.mil/whs/directives/corres/pdf/520044p.pdf [2].

Tipton, H.F. (ed.). 2006. *Official (ISC)2 guide to the CISSP CBK,* 1st ed. Boston, MA, USA: Auerbach Publications.

Primary References

Anderson, R.J. 2008. *Security Engineering: A Guide to Building Dependable Distributed Systems,* 2nd Ed. New York, NY, USA: John Wiley & Sons. Accessed October 24, 2014 at http://www.cl.cam.ac.uk/~rja14/book.html

DAU. 2012. "Defense Acquisition Guidebook (DAG): Chapter 13 -- Program Protection." Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). November 8, 2012. Accessed October 24, 2014 at https://dag.dau.mil/ [3]

ISO. 2008. "Information technology -- Security techniques -- Systems Security Engineering -- Capability Maturity Model® (SSE-CMM®)," Second Edition. Geneva, Switzerland: International Organization for Standardization (ISO), ISO/IEC 21827:2008.

ISO/IEC. 2013. "Information technology — Security techniques — Information security management systems — Requirements," Second Edition. Geneva, Switzerland: International Organization for Standardization (ISO)/International Electrotechnical Commission (IEC), ISO/IEC 27001:2013.

Kissel, R., K. Stine, M. Scholl, H. Rossman, J. Fahlsing, J. Gulick. 2008. "Security Considerations in the System Development Life Cycle," Revision 2. Gaithersburg, MD. National Institute of Standard and Technology (NIST), NIST 800-64 Revision 2:2008. Accessed October 24, 2014 at the Computer Security Resource Center [4]

Ross, R., J.C. Oren, M. McEvilley. 2014. "System Security Engineering: An Integrated Approach to Building Trustworthy Resilient Systems." Gaithersburg, MD. National Institute of Standard and Technology (NIST) Special Publication (SP), NIST SP 800-160:2014 (Initial Public Draft). Accessed October 24, 2014 at the Computer Security Resource Center http://csrc.nist.gov/publications/drafts/800-160/sp800_160_draft.pdf [5]

Additional References

Allen, Julia; Barnum, Sean; Ellison, Robert; McGraw, Gary; and Mead, Nancy. 2008. *Software security engineering: a guide for project managers.* New York, NY, USA: Addison Wesley Professional.

ISO. 2005. *Information technology -- Security techniques -- Code of practice for information security management.* Geneva, Switzerland: International Organization for Standardization (ISO). ISO/IEC 27002:2005.

Jurjens, J. 2005. "Sound Methods and effective tools for model-based security engineering with UML." *Proceedings of the 2005 International Conference on Software Engineering.* Munich, GE: ICSE, 15-21 May.

MITRE. 2012. "Systems Engineering for Mission Assurance." In *Systems Engineering Guide.* Accessed 19 June 2012 at MITRE http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/ se_for_mission_assurance/ [6].

NIST SP 800-160. *Systems Security Engineering - An Integrated Approach to Building Trustworthy Resilient Systems*. National Institute of Standards and Technology, U.S. Department of Commerce, Special Publication 800-160. Accessed October 24, 2014 at the Computer Security Resource Center http://csrc.nist.gov/publications/ drafts/800-160/sp800_160_draft.pdf [5].

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIC
END_ENCODED_CONTENT

References

[1] http://www.acq.osd.mil/se/docs/IEEE-SSE-Paper-02152012-Bkmarks.pdf
[2] http://www.dtic.mil/whs/directives/corres/pdf/520044p.pdf
[3] https://dag.dau.mil/
[4] http://csrc.nist.gov/publications/nistpubs/800-64-Rev2/SP800-64-Revision2.pdf
[5] http://csrc.nist.gov/publications/drafts/800-160/sp800_160_draft.pdf
[6] http://www.mitre.org/work/systems_engineering/guide/enterprise_engineering/se_for_mission_assurance/

System Assurance

Systems are subject to attacks for a multitude of reasons. System Assurance is the discipline that identifies and mitigates or removes exploitable vulnerabilities. This is increasingly important for both commercial and governmental activities.

System Assurance

NATO AEP-67 (Edition 1), Engineering for System Assurance in NATO Programs, defines system assurance (glossary) 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 (glossary) 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."

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.

References

Works Cited

CNSS. 2010. *National Information Assurance Glossary", Committee on National Security Systems Instruction (CNSSI) no. 4009". Fort Meade, MD, USA: The Committee on National Security Systems.*

DHS. 2010. *Build Security In.* Washington, DC, USA: US Department of Homeland Security (DHS). Accessed September 11, 2011. Available: https://buildsecurityin.us-cert.gov.

Goertzel, K., et al. 2008. *Enhancing the Development Life Cycle to Produce Secure Software: A Reference Guidebook on Software Assurance.* Washington, DC, USA: Data and Analysis Center for Software (DACS)/US Department of Homeland Security (DHS).

NATO. 2010. *Engineering for System Assurance in NATO programs.* Washington, DC, USA: NATO Standardization Agency. DoD 5220.22M-NISPOM-NATO-AEP-67.

Primary References

Anderson, Ross J. 2008. *Security Engineering: A Guide to Building Dependable Distributed Systems*, 2nd ed. New York, NY, USA: John Wiley & Sons.

Additional References

MITRE. 2011. "Systems Engineering for Mission Assurance." System Engineering Guide. Accessed March 7, 2012. Available: http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/ se_for_mission_assurance/.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIHF
END_ENCODED_CONTENT

References

[1] https://buildsecurityin.us-cert.gov/

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).

Electromagnetic Interference

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 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.

Electromagnetic Compatibility

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.

Figure 1. International & Regional Spectrum Management Forums. (SEBoK Original)

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 polices 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 reviews 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.

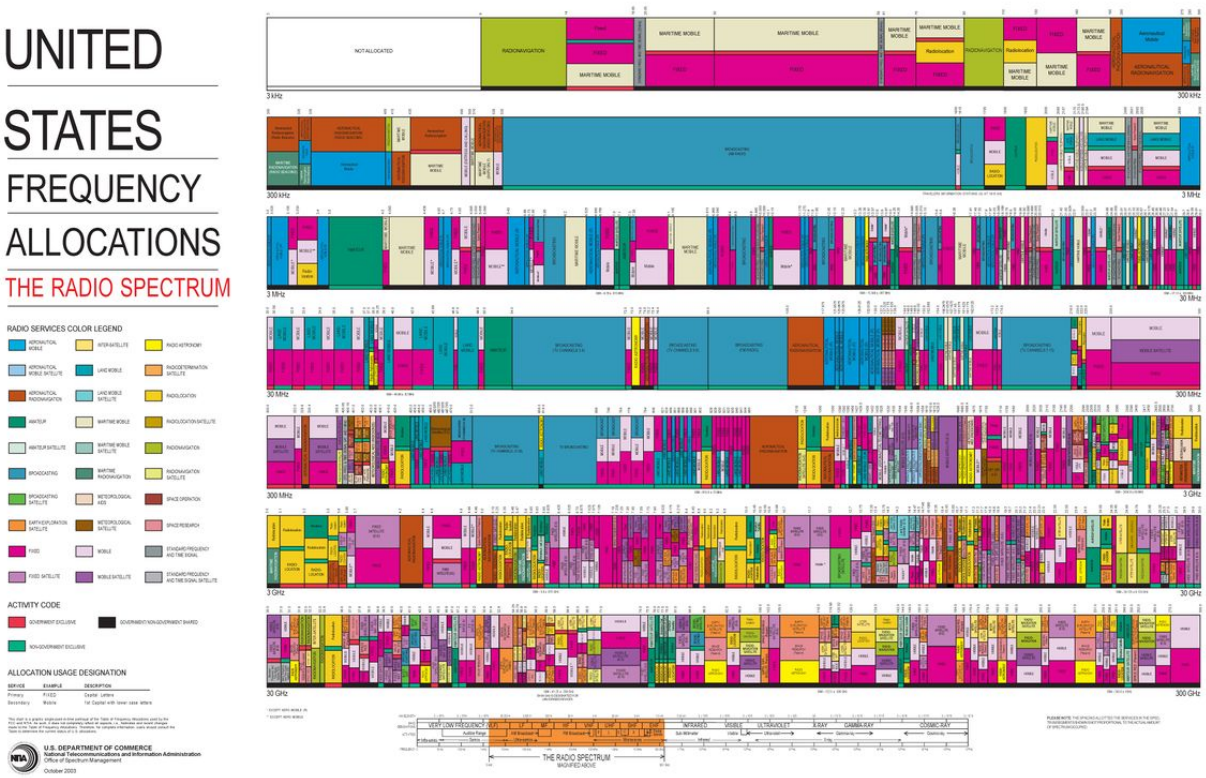


Figure 2. The Radio Spectrum (Department of Commerce 2003). Released by the U.S. Department of Commerce. Source is available at http://www.ntia.doc.gov/files/ntia/publications/2003-allochrt.pdf (Retrieved September 15, 2011)

Radiation Hardness

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.

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.

References

Works Cited

Bagad, V.S. 2009. *Electronic Product Design,* 4th ed. Pune, India: Technical Publications Pune.

DAU. 2010. *Defense Acquisition Guidebook (DAG).* Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

NSTISS. 1993. *Tempest Countermeasures for Facilities.* Ft. Meade, MD, USA: National Security Telecommunications and Information Systems Security (NSTISSI). 29 November, 1993. NSTISSI No. 7000.

NTIA. 2011. *Manual of Regulations and Procedures for Federal Radio Frequency Management,* May 2011 Revision of the 2008 Edition. Washington, DC, USA: National Telecommunications and Information Administration, U.S. Department of Commerce.

Stine, J. and D. Portigal. 2004. *An Introduction to Spectrum Management.* Bedford, MA, USA: MITRE Technical Report Spectrum. March 2004.

Van Lint, V.A.J. and A.G. Holmes Siedle. 2000. *Radiation Effects in Electronics: Encyclopedia of Physical Science and Technology.* New York, NY, USA: Academic Press.

Primary References

DAU. 2010. *Defense Acquisition Guidebook (DAG)*. Ft. Belvoir, VA, USA: Defense Acquisition University (DAU)/U.S. Department of Defense (DoD). February 19, 2010.

Additional References

None.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKiBSRUFEIFRISVMgQkVGT1JFIFVTSU5HIFRISVMgSkFWQVNDUklQVCAqIC
END_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).

Overview

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).

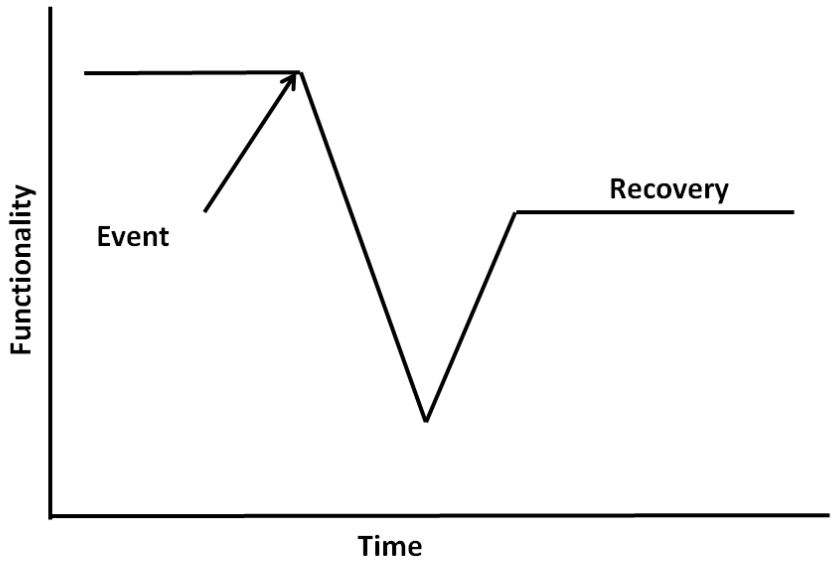


Figure 1. Disruption Diagram. (SEBoK Original)

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 capacity 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.

References

Works Cited

Billings, C. 1991. *Aviation Automation: A Concept and Guidelines.* Moffett Field, CA, USA: National Aeronautics and Space Administration (NASA).

CIPP. February 2007. *Critical Thinking: Moving from Infrastructure Protection to Infrastructure Resilience,* CIP Program Discussion Paper Series. Fairfax, VA, USA: Critical Infrastructure Protection (CIP) Program/School of Law/George Mason University (GMU).

DHS. 2010. *Department of Homeland Security Risk Lexicon*. Washington, DC, USA: US Department of Homeland Security, Risk Steering Committee. Available: http://www.dhs.gov/xlibrary/assets/dhs-risk-lexicon-2010.pdf.

Hitchins, D. 2009. "What Are The General Principles Applicable to Systems?" *INCOSE Insight* 12 (4): 59-63.

Hollnagel, E., D. Woods, and N. Leveson (eds). 2006. *Resilience Engineering: Concepts and Precepts.* Aldershot, UK: Ashgate Publishing Limited.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions.* Hoboken, NJ, USA: John Wiley & Sons.

Mendoca, D., and W. Wallace. 2006. "Adaptive Capacity: Electric Power Restoration in New York City Following the 11 September 2001 Attacks." Presented at 2nd Resilience Engineering Symposium, November 8-10, 2006, Juan-les-Pins, France.

C. T. Onions (ed.). 1973. *Oxford English Dictionary on Historical Principles*, 3rd ed., s.v. "Resilience". Oxford, UK: Oxford Univeristy Press.

Perrow, C. 1999. *Normal Accidents.* Princeton, NJ, USA: Princeton University Press.

Primary References

DHS. 2010. *Department of Homeland Security Risk Lexicon*. Washington, DC, USA: US Department of Homeland Security, Risk Steering Committee. Available: http://www.dhs.gov/xlibrary/assets/dhs-risk-lexicon-2010.pdf.

Jackson, S. 2010. *Architecting Resilient Systems: Accident Avoidance and Survival and Recovery from Disruptions.* Hoboken, NJ, USA: John Wiley & Sons.

Additional References

Jackson, S. 2007. "A Multidisciplinary Framework for Resilience to Disasters and Disruptions." *Journal of Design and Process Science.* 11: 91-108.

Madni, A., and S. Jackson. 2009. "Towards A Conceptual Framework for Resilience Engineering." *IEEE Systems Journal.* 3 (2): 181-191.

MITRE. 2011. "Systems Engineering for Mission Assurance." *System Engineering Guide.* Accessed March 7, 2012. Available: http:/ / www. mitre. org/ work/ systems_engineering/ guide/ enterprise_engineering/ se_for_mission_assurance/.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIHF
END_ENCODED_CONTENT

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.

Multiple Systems

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

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).

References

Works Cited

Maier, M., and E. Rechtin. 2002. *The Art of Systems Architecting,* 2nd ed. Boca Raton, FL, USA: CRC Press.

Blanchard, B.S., and W.J. Fabrycky. 2005. *Systems Engineering and Analysis,* 4th ed. Prentice-Hall International Series in Industrial and Systems Engineering. Englwood Cliffs, NJ, USA: Prentice-Hall.

Primary References

None.

Additional References

Anderson, D. 2010. *Design for Manufacturability & Concurrent Engineering; How to Design for Low Cost, Design in High Quality, Design for Lean Manufacture, and Design Quickly for Fast Production.* Cambria, CA, USA: CIM Press.

Boothroyd, G., P. Dewhurst, and W. Knight. 2010. *Product Design for Manufacture and Assembly.* 3rd Ed. Boca Raton, FL, USA: CRC Press.

Bralla, J. 1998. *Design for Manufacturability Handbook.* New York, NY, USA: McGraw Hill Professional.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKiBSRUQgQkVGT1JFIFVTSU5HIFRISVMgU0hPVVRCT1ghIFJFQUQgVEhFU0UgSU5TVFJVQ1RJT05TIEZJUlNUISAqICogKgogICAgICogVGhpcyBpcyB3aGVyZSB5b3UgY2FuIHNldCB1cCBjb25maWd1cmF0aW9uIHZhcmlhYmxlcyBmb3IgRGlzcXVzIGZvciB5b3VyIHNpdGUuIFlvdSBjYW4gZmluZCB0aGUgZnVsbCBsaXN0IG9mIHZhcmlhYmxlcwogICAgICogb24gb3VyIGNvbmZpZ3VyYXRpb24gZG9jdW1lbnRhdGlvbjogaHR0cHM6Ly9kaXNxdXMuY29tL2FwaS9kb2NzL2NvbmZpZ3VyYXRpb24vCiAgICAgKiBUaGVzZSB2YXJpYWJsZXMgYXJlIG9wdGlvbmFsOiBpZiBhIHZhcmlhYmxlIGlzIG5vdCBzZXQsIGl0IHNpbXBseSB3aWxsIG5vdCBiZSB1c2VkLgogICAgICovCiAgICAvKgogICAgdmFyIGRpc3F1c19jb25maWcgPSBmdW5jdGlvbiAoKSB7CiAgICAgICAgdGhpcy5wYWdlLnJlbW90ZV9hdXRoX3MzID0gIjxwcmUgc3R5bGU9XCJmb250LXNpemU6IDEwcHg7IGxpbmUtaGVpZ2h0OiAxMnB4O1wiPjxjb2RlPjwvY29kZT48L3ByZT4iOwogICAgICAgIHRoaXMucGFnZS5hcGlfa2V5ID0gIjxwcmUgc3R5bGU9XCJmb250LXNpemU6IDEwcHg7IGxpbmUtaGVpZ2h0OiAxMnB4O1wiPjxjb2RlPjwvY29kZT48L3ByZT4iOwogICAgICAgIHRoaXMucGFnZS5hcGlfYmFzZSA9ICI8cHJlIHN0eWxlPVwiZm9udC1zaXplOiAxMHB4OyBsaW5lLWhlaWdodDogMTJweDtcIj48Y29kZT48L2NvZGU+PC9wcmU+IjsKICAgIH07CiAgICAqLwogICAgKGZ1bmN0aW9uKCkgewogICAgICAgIHZhciBkID0gZG9jdW1lbnQsIHMgPSBkLmNyZWF0ZUVsZW1lbnQoJ3NjcmlwdCcpOwogICAgICAgIHMuc3JjID0gJy8vc2Vib2stZGlnaXRhbC5kaXNxdXMuY29tL2VtYmVkLmpzJzsKICAgICAgICBzLnNldEF0dHJpYnV0ZSgnZGF0YS10aW1lc3RhbXAnLCArbmV3IERhdGUoKSk7CiAgICAgICAgKGQuaGVhZCB8fCBkLmJvZHkpLmFwcGVuZENoaWxkKHMpOwogICAgfSkoKTsKPC9zY3JpcHQ+Cjxub3NjcmlwdD5QbGVhc2UgZW5hYmxlIEphdmFTY3JpcHQgdG8gdmlldyB0aGUgPGEgaHJlZj0iaHR0cHM6Ly9kaXNxdXMuY29tLz9yZWY9bm9zY3JpcHQiPmNvbW1lbnRzIHBvd2VyZWQgYnkgRGlzcXVzLjwvYT48L25vc2NyaXB0Pgo8ZGl2IGNsYXNzPSJjbGVhciI+PC9kaXY+
END_ENCODED_CONTENT

Affordability

A system is affordable to the degree that system performance, cost, and schedule constraints are balanced over the system life, while mission needs are satisfied in concert with strategic investment and organizational needs (INCOSE 2011). Design for affordability is the practice of considering affordability as a design characteristic or constraint.

Increasing competitive pressures and the scarcity of resources demand that systems engineering (SE) improve affordability. Several recent initiatives have made affordability their top technical priority. They also call for a high priority to be placed on research into techniques — namely, improved systems autonomy and human performance augmentation — that promise to reduce labor costs, provide more efficient equipment to reduce supply costs, and create adaptable systems whose useful lifetime is extended cost-effectively.

Yet methods for cost and schedule estimation have not changed significantly to address these new challenges and opportunities. There is a clear need for

- new methods to analyze tradeoffs between cost, schedule, effectiveness, and resilience;
- new methods to adjust priorities and deliverables to meet budgets and schedules; and
- more affordable systems development processes.

All of this must be accomplished the context of the rapid changes underway in technology, competition, operational concepts, and workforce characteristics.

Background

Historically, cost and schedule estimation has been decoupled from technical SE tradeoff analyses and decision reviews. Most models and tools focus on evaluating either cost-schedule performance or technical performance, but not the tradeoffs between the two. Meanwhile, organizations and their systems engineers often focus on affordability to minimize acquisition costs. They are then drawn into the easiest-first approaches that yield early successes, at the price of being stuck with brittle, expensive-to-change architectures that increase technical debt and life cycle costs.

Two indications that the need for change is being recognized in systems engineering are that the *INCOSE SE Handbook* now includes affordability as one of the criteria for evaluating requirements (INCOSE 2011); and, there is

a trend in SE towards stronger focus on maintainability, flexibility, and evolution (Blanchard, Verma, and Peterson 1995).

Modularization

Modularization of the system's architecture around its most frequent sources of change (Parnas 1979) is a key SE principle for affordability. This is because when changes are needed, their side effects are contained in a single systems element, rather than rippling across the entire system.

This approach creates the need for three further improvements:

- refocusing the system requirements, not only on a snapshot of current needs, but also on the most likely sources of requirements change, or evolution requirements;
- monitoring and acquiring knowledge about the most frequent sources of change to better identify requirements for evolution; and
- evaluating the system's proposed architecture to access how well it supports the evolution requirements, as well as the initial snapshot requirements.

This approach can be extended to produce several new practices. Systems engineers can

- identify the commonalities and variability across the families of products or product lines, and develop architectures for creating (and evolving) the common elements *once* with plug-compatible interfaces for inserting the variable elements (Boehm, Lane, and Madachy 2010);
- extrapolate principles for service-oriented system elements that are characterized by their inputs, outputs, and assumptions, and that can easily be composed into systems in which the sources of change were not anticipated; and
- develop classes of smart or autonomous systems whose many sensors identify needed changes, and whose autonomous agents determine and effect those changes in microseconds, or much more rapidly than humans can, reducing not only reaction time, but also the amount of human labor needed to operate the systems, thus improving affordability.

Pitfalls

There are pitfalls for the unwary. Autonomous systems experience several hazardous failure modes, including

- **system instability due to positive feedback —** where an agent senses a parameter reaching a control limit and gives the system a strong push in the other direction, causing the system to rapidly approach the other control limit, causing the agent (or another) to give it an even stronger push in the original direction, and so on
- **self-modifying autonomous agents which fail** after several self-modifications — the failures are difficult to debug because the agent's state has been changing
- **autonomous agents performing weakly at commonsense reasoning** about system control decisions by human operators, and so tend to reach wrong conclusions and make wrong decisions about controlling the system
- **multiple agents making contradictory decisions** about controlling the system, and lacking the ability to understand the contradiction or to negotiate a solution to resolve it

Practical Considerations

Autonomous systems need human supervision, and the humans involved require better methods for trend analysis and visualization of trends (especially, undesired ones).

There is also the need, with autonomous systems, to extend the focus from life cycle costs to total ownership costs, which encompass the costs of failures, including losses in sales, profits, mission effectiveness, or human quality of life. This creates a further need to evaluate affordability in light of the value added by the system under consideration. In principle, this involves evaluating the system's total cost of ownership with respect to its mission effectiveness and resilience across a number of operational scenarios. However, determining the appropriate scenarios and their relative importance is not easy, particularly for multi-mission systems of systems. Often, the best that can be done involves a mix of scenario evaluation and evaluation of general system attributes, such as cost, schedule, performance, and so on.

As for these system attributes, different success-critical stakeholders will have different preferences, or utility functions, for a given attribute. This makes converging on a mutually satisfactory choice among the candidate system solutions a difficult challenge involving the resolution of the multi-criteria decision analysis (MCDA) problem among the stakeholders (Boehm and Jain 2006). This is a well-known problem with several paradoxes, such as Arrow's impossibility theorem that describes the inability to guarantee a mutually optimal solution among several stakeholders, and several paradoxes in stakeholder preference aggregation in which different voting procedures produce different winning solutions. Still, groups of stakeholders need to make decisions, and various negotiation support systems enable people to better understand each other's utility functions and to arrive at mutually satisfactory decisions, in which no one gets everything that they want, but everyone is at least as well off as they are with the current system.

Also see System Analysis for considerations of cost and affordability in the technical design space.

Primary References

Works Cited

Blanchard, B., D. Verma, and E. Peterson. 1995. *Maintainability: A Key to Effective Serviceability and Maintenance Management.* New York, NY, USA: Wiley and Sons.

Boehm, B., J. Lane, and R. Madachy. 2010. "Valuing System Flexibility via Total Ownership Cost Analysis." Proceedings of the NDIA SE Conference, October 2010, San Diego, CA, USA.

Boehm, B., and A. Jain. 2006. "A Value-Based Theory of Systems Engineering." Proceedings of the International Council on Systems Engineering (INCOSE) International Symposium (IS), July 9-13, 2006, Orlando, FL, USA.

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2. p. 79.

Parnas, D.L. 1979. "Designing Software for Ease of Extension and Contraction." *IEEE Transactions on Software Engineering*. 5 (2): 128-138.

Primary References

INCOSE. 2012. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 3.2.2. San Diego, CA, USA: International Council on Systems Engineering (INCOSE), INCOSE-TP-2003-002-03.2.2. p. 79.

Blanchard, B., D. Verma, and E. Peterson. 1995. *Maintainability: A Key to Effective Serviceability and Maintenance Management.* New York, NY, USA: Wiley and Sons.

Parnas, D.L. 1979. "Designing Software for Ease of Extension and Contraction." *IEEE Transactions on Software Engineering*. 5 (2): 128-138.

Additional References

Kobren, Bill. 2011. "Supportability as an Affordability Enabler: A Critical Fourth Element of Acquisition Success Across the System Life Cycle." *Defense AT&L: Better Buying Power.* Accessed August 28, 2012. Available: http://www.dau.mil/pubscats/ATL%20Docs/Sep-Oct11/Kobren.pdf.

Myers, S.E., P.P. Pandolfini, J.F. Keane, O. Younossi, J.K. Roth, M.J. Clark, D.A. Lehman, and J.A. Dechoretz. 2000. "Evaluating affordability initiatives." *Johns Hopkins APL Tech. Dig.* 21 (3): 426–437.

< Previous Article | Parent Article | Next Article >

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qIEk
END_ENCODED_CONTENT

Environmental Engineering

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.

Operating Environment

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.

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).

Environmental Impact

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.

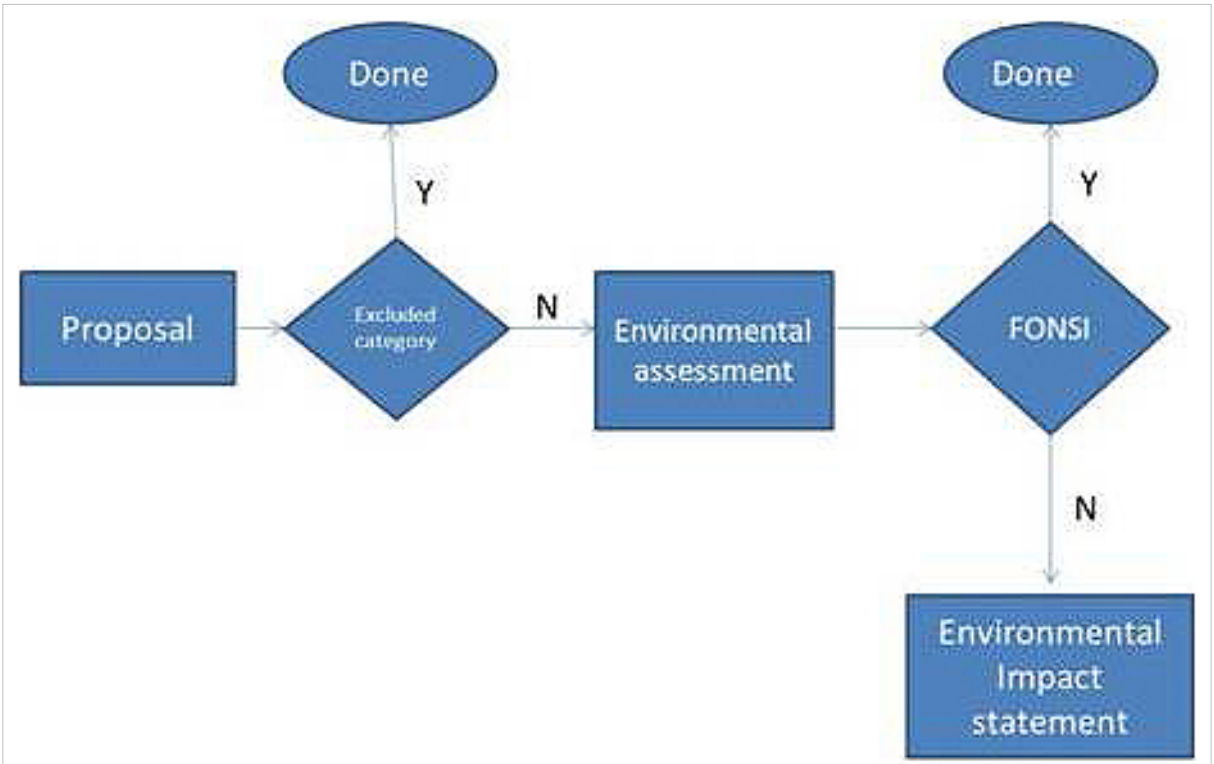


Figure 1. Flowchart to Decide if an EIS is Necessary. (SEBoK Original)

Preparation of an EIS is a resource significant task. Bregman (2000) and Kreske (1996) provide accessible overviews of the process. Lee and Lin (2000) provide a handbook of environmental engineering calculations to aid in the technical submission. Numerous firms offer consulting services.

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.

Best Practices

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

Green Design

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.

Energy Efficiency

There is a large amount of literature that has been published about design for energy efficiency. Lovins (2010) offer 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).

Compliance and the Enterprise

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.

ISO14001 sets the standards for organization to comply with environmental regulations. Kwon and Seo (2002) discuss this in a Korean context, and Whitelaw (2004) presents a handbook on implementing ISO14001.

References

Works Cited

Berry, MA. 1998. "Proactive corporate environmental management: a new industrial revolution." *The Academy of Management Executive*, 12 (2): 38-50.

Bregman, J.I. 2000. *Environmental Impact Statements*, 2nd ed. Boca Raton, FL, USA: CRC Press.

Brown, C. 2011 "The Green Fleet Price Tag." Business Fleet. Available: http://www.businessfleet.com/Article/Story/2011/07/The-Green-Fleet-Price-Tag.aspx.

BSI. 2011. "Specification for the assessment of the life cycle greenhouse gas emissions of goods and service, PAS 2050:2011." London, UK: British Standards Institution (BSI). Available: http://shop.bsigroup.com/en/forms/PASs/PAS-2050.

Burby, R.J., and R.G. Paterson. 1993. "Improving compliance with state environmental regulations." *Journal of Policy Analysis and Management*, 12(4): 753–772.

Dechant, K., B. Altman, R.M. Downing, and T. Keeney. 1994. "Environmental Leadership: From Compliance to Competitive Advantage." *Academy of Management Executive*, 8(3): 7.

DoD. 2014. *Department of Defense Test Method Standard: Environmental Engineering Considerations and Laboratory Tests*, MIL-STD-810G Change Notice 1. Washington, DC, USA: US Army Test and Evaluation Command, US Department of Defense (DoD). Accessed November 4, 2014. Available: http://www.atec.army.mil/publications/Mil-Std-810G/MIL-STD-810G%20CN1.pdf.

Eccleston, C. 2000. *Environmental Impact Statements: A Comprehensive Guide to Project and Strategic Planning*. New York, NY, USA: Wiley.

EPA. 2011. "Green Engineering. Environmental Protection Agency (EPA)." Available: http://www.epa.gov/oppt/greenengineering/.

EC. 1985. "Council Directive of 27 June 1985 on the assessment of the effects of certain public and private projects on the environment (85/337/EEC)." European Commission (EC). Available: http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CONSLEG:1985L0337:20090625:EN:PDF.

FAA. 2002. "Best Practices for Environmental Impact Statement (EIS) Management." Federal Aviation Administration (FAA). Available: http://www.faa.gov/airports/environmental/eis_best_practices/?sect=intro.

Graedel, T.E., and B.R. Allenby. 2009. *Industrial Ecology and Sustainable Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Gupta, M.C. 1995. "Environmental management and its impact on the operations function." *International Journal of Operations and Production Management*, 15 (8): 34-51.

Hill, T. 2011. "Honolulu Rail's Next Stop?" *Honolulu Magazine*. July 2011.

Intel. 2011. "Energy Efficiency." Intel Corporation. Accessed: August 29, 2012. Available: http://www.intel.com/intel/other/ehs/product_ecology/energy.htm.

Kreske, D.L. 1996. *Environmental impact statements: a practical guide for agencies, citizens, and consultants*. New York, NY: Wiley.

Kwon, D.M., and M.S. Seo. 2002. "A study of compliance with environmental regulations of ISO 14001 certified companies in Korea." *Journal of Environmental Management*. 65 (4): 347-353.

Lee, C.C., and S.D. Lin. 2000. *Handbook of Environmental Engineering Calculations*. New York, NY, USA: McGraw Hill Professional.

Lockton, D., and D. Harrison. 2008. "Making the user more efficient: Design for sustainable behaviour." *International Journal of Sustainable Engineering*. 1 (1): 3-8.

Lovins, A. 2010. "Factor Ten Engineering Design Principles," version 1.0. Available: http:// www. rmi. org/ Knowledge-Center/Library/2010-10_10xEPrinciples.

Lovins, A., et al. 2011. "Case Studies." Available: http://move.rmi.org/markets-in-motion/case-studies/.

Maydl, Peter. 2004. "Sustainable Engineering: State-of-the-Art and Prospects." *Structural Engineering International*. 14 (3): 176-180.

Meryman, H. 2004. "Sustainable Engineering Using Specifications to Make it Happen." *Structural Engineering International*. 14 (3).

MoD. 2006. *Standard 00-35, Environmental Handbook for Defence Materiel (Part 3) Environmental Test Methods*. London, England, UK: UK Ministry of Defence (MoD). Available: http:// www. everyspec. com/ DEF-STAN/ download.php?spec=DEFSTAN00-35_I4.029214.pdf.

Nash, J. 2001. *Regulating from the inside: can environmental management systems achieve policy goals?* Washington, DC, USA: Resources for the Future Press.

NEPA. 1969. *42 USC 4321-4347. National Environmental Policy Act (NEPA)*. Accessed January 15, 2012. Available: http://ceq.hss.doe.gov/nepa/regs/nepa/nepaeqia.htm.

Stasinopoulos, P. 2009. *Whole system design: an integrated approach to sustainable engineering*. London, UK: Routledge.

UN. 1992. "Rio Declaration on Environment and Development." United Nations (UN). Available: http:// www. unep.org/Documents.Multilingual/Default.asp?documentid=78&articleid=1163.

Whitelaw, K. 2004. *ISO 14001: Environmental Systems Handbook*, 2nd ed. Oxford, UK: Elsevier.

WCED. 1987. "Our Common Future. World Commission on Economic Development (WCED)." Available: http:// www.un-documents.net/wced-ocf.htm.

Primary References

Bregman, J.I. 2000. *Environmental Impact Statements*, 2nd ed. Boca Raton, FL, USA: CRC Press.

Graedel, T.E., and B.R. Allenby. 2009. *Industrial Ecology and Sustainable Engineering*. Upper Saddle River, NJ, USA: Prentice Hall.

Lee, C.C., and S.D. Lin. 2000. *Handbook of Environmental Engineering Calculations*. New York, NY, USA: McGraw Hill Professional.

Whitelaw, K. 2004. *ISO 14001: Environmental Systems Handbook*, 2nd ed. Oxford, UK: Elsevier.

Additional References

None.

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

SEBoK v. 1.3.1 released 5 December 2014

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 [20].

ENCODED_CONTENT
PGRpdiBpZD0iZGlzcXVzX3RocmVhZCI+PC9kaXY+CjxzY3JpcHQgdHlwZT0idGV4dC9qYXZhc2NyaXB0Ij4KICAgIC8qICogKiBS
END_ENCODED_CONTENT

Article Sources and Contributors

Letter from the Editor *Source*: http://www.sebokwiki.org/d/index.php?oldid=50309 *Contributors*: Apyster, Bkcase, Cnielsen, Dholwell, Eleach, Kguillemette, Radcock, Smenck2, Wikiexpert

BKCASE Governance and Editorial Board *Source*: http://www.sebokwiki.org/d/index.php?oldid=50356 *Contributors*: Apyster, Bkcase, Cnielsen, Dhenry, Kguillemette, Radcock, Smenck2

Acknowledgements and Release History *Source*: http://www.sebokwiki.org/d/index.php?oldid=50357 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Ddori, Dhenry, Dholwell, Eleach, Janthony, Jgercken, Kguillemette, Radcock, Smenck2, Smurawski, Wikiexpert

Related Disciplines *Source*: http://www.sebokwiki.org/d/index.php?oldid=50204 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Kguillemette, Rmalove, SYSIND11, Smenck2, Wikiexpert, Zamoses

Systems Engineering and Software Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=49748 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Rmadachy, Rmalove, Skmackin, Smurawski, Wikiexpert, Zamoses

The Nature of Software *Source*: http://www.sebokwiki.org/d/index.php?oldid=49114 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Rmalove, Wikiexpert, Zamoses

An Overview of the SWEBOK Guide *Source*: http://www.sebokwiki.org/d/index.php?oldid=49205 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Hdavidz, Jgercken, Kguillemette, Rmalove, Smenck2, Wikiexpert, Zamoses

Key Points a Systems Engineer Needs to Know about Software Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=49717 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Eleach, Jgercken, Kguillemette, Rmalove, Smenck2, Smurawski, Wikiexpert, Zamoses

Key Points a Systems Engineer Needs to Know about Managing a Software Team *Source*: http://www.sebokwiki.org/d/index.php?oldid=49715 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Eleach, Kguillemette, Rmalove, Smenck2, Wikiexpert

Systems Engineering and Project Management *Source*: http://www.sebokwiki.org/d/index.php?oldid=50201 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Kguillemette, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses

The Nature of Project Management *Source*: http://www.sebokwiki.org/d/index.php?oldid=50205 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Hdavidz, Jgercken, Kguillemette, Rmalove, Rturner, Smenck2, Wikiexpert, Zamoses

An Overview of the PMBOK® Guide *Source*: http://www.sebokwiki.org/d/index.php?oldid=50206 *Contributors*: Asquires, Bkcase, Dhenry, Dholwell, Kguillemette, Rmalove, Smenck2, Wikiexpert

Relationships between Systems Engineering and Project Management *Source*: http://www.sebokwiki.org/d/index.php?oldid=49461 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Eleach, Jgercken, Kguillemette, Rturner, Smenck2, Smurawski, Wikiexpert, Zamoses

The Influence of Project Structure and Governance on Systems Engineering and Project Management Relationships *Source*: http://www.sebokwiki.org/d/index.php?oldid=49712 *Contributors*: Asquires, Bkcase, Dhenry, Dholwell, Eleach, Kguillemette, Smenck2, Smurawski, Wikiexpert

Systems Engineering and Industrial Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=50208 *Contributors*: Apyster, Bkcase, Dhenry, Dholwell, Eleach, Hsillitto, Kguillemette, SYSIND11, Smenck2, Wikiexpert

Systems Engineering and Procurement/Acquisition *Source*: http://www.sebokwiki.org/d/index.php?oldid=50209 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Jgercken, Kguillemette, Rmalove, Skmackin, Smenck2, Smurawski, Wikiexpert, Zamoses

Systems Engineering and Specialty Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=48451 *Contributors*: Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Rmalove, Rturner, Skmackin, Smenck2, Wikiexpert, Zamoses

Integration of Specialty Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=46168 *Contributors*: Apyster, Asquires, Bkcase, Dcarey, Dfairley, Dhenry, Jgercken, Kforsberg, Rmalove, Skmackin, Smenck2, Smurawski, Wikiexpert, Zamoses

Reliability, Availability, and Maintainability *Source*: http://www.sebokwiki.org/d/index.php?oldid=48317 *Contributors*: Apyster, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Rmalove, Rturner, Skmackin, Smenck2, Smurawski, Wikiexpert, Zamoses

Human Systems Integration *Source*: http://www.sebokwiki.org/d/index.php?oldid=48802 *Contributors*: Apyster, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Rmalove, Sbooth, Skmackin, Smenck2, Smurawski, Wikiexpert, Zamoses

Safety Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=49997 *Contributors*: Apyster, Asquires, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Jgercken, Rmalove, Skmackin, Smenck2, Smurawski, Wikiexpert, Zamoses

Security Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=49987 *Contributors*: Apyster, Araher, Asquires, Bkcase, Cnielsen, Dfairley, Dhenry, Dholwell, Jgercken, Skmackin, Smenck2, Wikiexpert, Zamoses

System Assurance *Source*: http://www.sebokwiki.org/d/index.php?oldid=50210 *Contributors*: Apyster, Araher, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Kguillemette, Wikiexpert, Zamoses

Electromagnetic Interference/Electromagnetic Compatibility *Source*: http://www.sebokwiki.org/d/index.php?oldid=46957 *Contributors*: Apyster, Araher, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Jsnoderly, Rturner, Skmackin, Smenck2, Smurawski, Wikiexpert, Zamoses

Resilience Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=50211 *Contributors*: Apyster, Araher, Asquires, Bkcase, Dhenry, Dholwell, Jgercken, Kguillemette, Rturner, Sjackson, Skmackin, Smenck2, Smurawski, Wikiexpert, Zamoses

Manufacturability and Producibility *Source*: http://www.sebokwiki.org/d/index.php?oldid=49117 *Contributors*: Apyster, Araher, Asquires, Bkcase, Dfairley, Dhenry, Dholwell, Jgercken, Kforsberg, Rturner, Skmackin, Wikiexpert, Zamoses

Affordability *Source*: http://www.sebokwiki.org/d/index.php?oldid=50213 *Contributors*: Araher, Bkcase, Dhenry, Dholwell, Kguillemette, Rmadachy, Smenck2, Wikiexpert

Environmental Engineering *Source*: http://www.sebokwiki.org/d/index.php?oldid=50214 *Contributors*: Bkcase, Cnielsen, Dhenry, Dholwell, Eleach, Kguillemette, Smenck2, Smurawski, Wikiexpert

Image Sources, Licenses and Contributors

File:RickSignature_Full.png *Source*: http://www.sebokwiki.org/d/index.php?title=File:RickSignature_Full.png *License*: unknown *Contributors*: Bkcase

File:RickAdcock.png *Source*: http://www.sebokwiki.org/d/index.php?title=File:RickAdcock.png *License*: unknown *Contributors*: Bkcase

File:PM-SE1.jpg *Source*: http://www.sebokwiki.org/d/index.php?title=File:PM-SE1.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

File:PM-SE2.jpg *Source*: http://www.sebokwiki.org/d/index.php?title=File:PM-SE2.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

File:P6_Fig1_The_Organizational_Continuum_KN.jpg *Source*: http://www.sebokwiki.org/d/index.php?title=File:P6_Fig1_The_Organizational_Continuum_KN.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

File:ACQProcessModel_NoWhiteS.png *Source*: http://www.sebokwiki.org/d/index.php?title=File:ACQProcessModel_NoWhiteS.png *License*: unknown *Contributors*: Dhenry, Smenck2, Smurawski

File:RelatingACQtoRFP_NoWhiteS.png *Source*: http://www.sebokwiki.org/d/index.php?title=File:RelatingACQtoRFP_NoWhiteS.png *License*: unknown *Contributors*: Smenck2, Smurawski

File:Fig._1_Integration_Process_for_Specialty_Engineering.png *Source*: http://www.sebokwiki.org/d/index.php?title=File:Fig._1_Integration_Process_for_Specialty_Engineering.png *License*: unknown *Contributors*: Smenck2, Smurawski

File:SEBoKv05_KA-SpecialtyEng_RAM_Eqation1.png *Source*: http://www.sebokwiki.org/d/index.php?title=File:SEBoKv05_KA-SpecialtyEng_RAM_Eqation1.png *License*: unknown *Contributors*: Bkcase

File:Fault_tree.jpg *Source*: http://www.sebokwiki.org/d/index.php?title=File:Fault_tree.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

File:Simple_RBD.jpg *Source*: http://www.sebokwiki.org/d/index.php?title=File:Simple_RBD.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

File:Figure 1.png *Source*: http://www.sebokwiki.org/d/index.php?title=File:Figure_1.png *License*: unknown *Contributors*: Smenck2, Smurawski

File:Figure 2.jpg *Source*: http://www.sebokwiki.org/d/index.php?title=File:Figure_2.jpg *License*: unknown *Contributors*: Smenck2, Smurawski

File:Disruption_Diagram.PNG *Source*: http://www.sebokwiki.org/d/index.php?title=File:Disruption_Diagram.PNG *License*: unknown *Contributors*: Bkcase, Smurawski

File:Environmental_Engineering_HighRes.jpg *Source*: http://www.sebokwiki.org/d/index.php?title=File:Environmental_Engineering_HighRes.jpg *License*: unknown *Contributors*: Smenck2, Smurawski