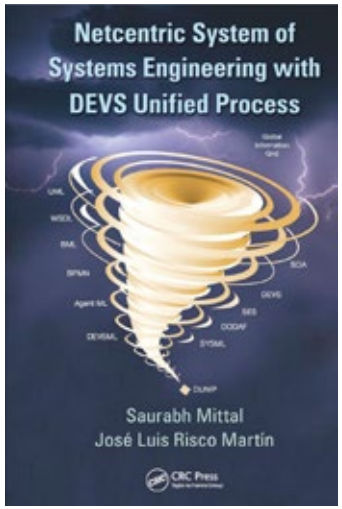
Netcentric System of Systems Engineering with DEVS Unified Process

By Saurabh Mittal and José Luis Risco Martin
Boca Raton, US-FL: CRC Press, 2013 (ISBN 978-1-4398-2706-2)
684 pp., including list of acronyms and index

Reviewed by Sarah Sheard, sarah.sheard@incose.org



A long-standing conflict between the systems engineering and the software engineering definitions of the "system of systems" is eroding because all systems of note today contain some software (at least in development tools, but almost always in the system), and nearly all software today must be considered in terms of its system functions. Nevertheless, there is still a significant difference between systems engineering approaches that take a big-picture, functions-first view, and software engineering approaches that are much more interested in the details of how to make the software work.

This book clearly takes the second approach. A randomly selected 25 pages in the book contained 31% code (or program output), 42% text, 6% equations, and 12% figures (plus references and exercises), which seems consistent with the book as a whole. This translates to over 200 pages of code and about 280 pages of text.

In this book, system development is development of a discrete-event simulation model of the logical behavior of a distributed system of computers and software. INCOSE readers who are not interested in this view of systems of systems would be better off reading other books.

That said, and given the trends in systems engineering suggested by the opening sentence, there is interesting content in this book for systems engineers who are interested in model-based systems engineering and in different kinds of simulation, as well as those interested in how the software for distributed systems of computer systems can be modeled. In particular, by reading sections I, IV, and V, INCOSE systems engineers whose work intersects heavily with software engineering can profit from

- learning terminology used by software simulation experts;
- reading the hundreds of pages of Java programs, in lieu of taking a Java class;
- the list of personnel needed to be brought together to create a good system-of-systems-simulation, including the role of the systems engineer, according to software simulators (p. 185); or
- learning what the software engineers expect a systems engineer to do regarding interviewing system-of-systems personnel to create system simulation scenarios and use cases for the software for a distributed system of systems (section IV, beginning on page 499).

The contents of the book are as follows. Sections II and III will not appeal to "generalist" systems engineers, as they are intended more for graduate students in system-of-system simulations.

Section I (chapters 1–6, 188 pages) describes the "Basics." Chapter 1 defines simulation, object-oriented software systems engineering (the authors' term), and introduces the Java programming language (which constitutes most of those 200 pages of code). Chapter 2 describes systems of systems and complex systems without seeming to understand INCOSE's uses for the terms, even though the references cited are consistent with INCOSE's uses. Instead, after four pages of high-level description, chapter 2 dives right into a hierarchy of system-simulation methods and model formalisms. Chapters 3 and 4 list the formalism associated with discrete event simulations (DEVS) and then the modeling and simulation metamodels used in the book. Chapter 5 discusses the DEVS language in a manner that was incomprehensible to me. (Sample sentences: "The atomic DEVS formalism has deltint, deltext, deltcon, and lamda functions to specify the atomic behavior," and "All the above behavior specifications are code-assisted and validated, as behavior is specified in the editor.") Chapter 6 discusses the DEVS Unified Process. This is not a process like systems engineering processes or CMMI. This process defines stacks and transformations among kinds of models, as well as how to align with a software-development framework called OpenUTF.

Section II (chapters 7–12, 148 pages) is called "Modeling and Simulation-Based Systems Engineering." This is not INCOSE's "systems engineering" except possibly as a working group called "Model-based software systems engineering" would see it. According to the preface, this section is for graduate students and advanced practitioners of DEVS, and for industry professionals who want to "learn the advanced capabilities of DEVS-based systems engineering methodology [sic] and to use [model-driven engineering] in their efforts." Chapter 11 describes, and recommends changes to, the United States Department of Defense Architecture Framework, version 1.0. (Note that this was replaced by version 1.5 in 2007, and the framework is now on version 2.02.). Chapter 12 suggests how that framework and other architecture frameworks can be tested.

Section III (chapters 13–17, 160 pp.), is intended for people "who are interested in building DEVS virtual machines and netcentric SoS." This section has 50% code, 27% figures and tables, and only 15% text (again sampling 25 pages). I can't

Netcentric System of Systems Engineering *continued*

imagine many INCOSE systems engineers making it all the way through this section, much less using it. I can only see software engineers using it.

Section IV (chapters 18–22, 140 pages) is about "Case Studies," which sounds like where INCOSE systems engineers would find value. Chapters 18–20 indeed discuss designing a software simulation from informal scenarios. These chapters contain primarily figures and code. They would certainly help teach a senior INCOSE systems engineer what simulation software engineers on a distributed software system might be looking for, or better, serve as a conversation starter between the systems and software engineers along the lines of, what do you need and what should I be trying to create for you? Chapter 21 describes executable UML. Much of this content has been made available by the INCOSE Model-Based System Engineering Working Group in a way that is more digestible by INCOSE systems engineers; however, the advantage of this chapter is that it appears in a book with the other content, with similar language and formalisms. INCOSE systems engineers who have survived chapters 7–20 will find this chapter easy to read and relatively familiar. Chapter 22 addresses using these techniques to model an enterprise's business processes.

Section V (chapter 23) purports to address Netcentric (sic) Complex Adaptive Systems. The background sections are in agreement with the definitions used by INCOSE's Complex System Working Group. Particularly valuable is the table of questions to ask about the complex adaptive system to be able to model it correctly (p. 650 and following). However, the chapter falls short of providing easily understood advice about modeling such a system.

Weaknesses include a tendency to jump right into Java coding detail after minimal context and explanation, failure to explain the acronyms before using them (and not all are in the acronyms list at the back), and the use of confusing and unidiomatic English such as the following: "To the contrary of the structured programming, centered on functions, object-oriented programming is centered on data," and "When we define a class that does not extend other class, Java uses the Object class, which is the root of the class hierarchy in Java."

INCOSE Author News

China Aviation Publishing & Media Co. Ltd of Beijing, China, has announced that it will be publishing a Chinese translation of INCOSE Fellow Scott Jackson's book *Systems Engineering for Commerical Aircraft*, first published in 1997 (Aldershot, GB: Ashgate).

Just One More Thing

Andrew Cashner, cashner@uchicago.edu**, assistant editor,** ***INSIGHT***

This is my final issue as assistant editor of ***INSIGHT***. My heartfelt thanks go to Chief Editor Bob Kenley for a productive and supportive collaboration over the last seven years. I thank Bob, Chuck Eng, Cecilia Haskins, and all those throughout the Council, for their efforts to make this periodical an open forum for international discussion among systems engineering practitioners across subfields and specialties. I have enjoyed the opportunity to work with the regular contributors and theme editors. With apologies to the many others not listed, these include Rick Dove, Denise Howard, Samantha Robitaille, John Thomas, Dave Walden, and Holly Witte. I especially wish to thank Jack Ring for his original and thought-provoking articles. I am a musicologist, not a systems engineer, but I have been inspired by Jack Ring's concept of knowledge as dynamic exchange within a community, to be measured not in quantity but in velocity—and I have been privileged to see that exchange happening in these pages. There are many other cases in which my encounter with systems engineering has influenced my work as a musician and musicologist: from inspiring me to think in terms of systems (my dissertation in part traces networks of musical influence and exchange across seventeenth-century Mexico and Spain), to motivating me to approach the research process itself as a systems engineering challenge.

Editing ***INSIGHT*** has given me a special vantage point into the INCOSE community, and I have admired the warmth and collegiality within the discipline. In the smallest of ways, I have sought to help this publication represent your vibrant, highly dynamic community in all its breadth. Going forward, I hope ***INSIGHT***, and INCOSE in general, continues to seek a truly global perspective. I hope the INCOSE members will continue to foster a culture of straightforward communication that will make discoveries and ideas in specialized domains widely accessible. Very best wishes to my successor Lisa Hoverman and to all of INCOSE.