1 Modeling and Simulation of Systems of Systems

This book is about modeling and simulation in support of "virtual build and test" for Systems of Systems (SoS), which include complex information-technology-based business, engineering, military systems, as well as the societal infrastructures they support. Such systems are at the root of this century's global challenges of interacting economic crises, world-wide crop failures, extreme effects of climate change, and out-of-control viral epidemics. We are accustomed to building such systems directly in the real world and letting subsequent use and Mother Nature tell us how good they are. Increasingly, however, it is becoming too dangerous, costly, unethical, or risky to do so. "Build and test within virtual reality" is more often the only workable alternative—where by "virtual" we include a wide range of representations of the eventual fielded reality either wholly within a single computer or as networked distributed simulations, often enhanced with physically analogous and immersive environments.

Following are a few representative examples of problems needing the systems of systems concept. We briefly look at motivating objectives, why current approaches are not adequate to meet these objectives, and how the SoS concept and supporting virtual build and test environments can overcome conventional limitations.

Controlling National Health Care Costs

- *Objective*: coordinate the various health-related systems including hospitals, doctors, pharmacists, and insurers, which are currently largely uncoordinated, so as to improve patient care and greatly reduce its cost.
- *Key Conventional Limitation*: it is not feasible to experiment in reality with alternative systems for coordinating the interactions of the medical, pharmaceutical, and insurance subsystems.
- *System of Systems Approach*: model the national health care system as a system of systems formulating architectures of coordination and measures of quality for health care delivery.
- *Virtual Build and Test*: develop a family of coordinating architectures which employ net-centric information technology and can be evaluated in a simulated environment of providers and clients for their effectiveness in improving care while reducing cost.

Cloud System Architectures

- *Objective*: develop simulation models that can capture software and hardware parts of service-oriented computing systems and their integration.
- *Key Conventional Limitations*: understanding of cloud system architectures using software-centric simulation approaches are inadequate. In particular, simulation-based architectural designs with heavy emphasis on software are not suitable for capturing cloud system dynamics.
- *System of Systems Approach*: introduce co-design concepts commonly used in embedded system simulation for cloud systems.
- *Virtual Build and Test*: formulate domain-neutral software system model abstractions that account for SOA principles (i.e., elevating component simulation to simulation-as-a-service concept). Develop hardware system simulation models that can be integrated with software system simulation models in a systematic fashion.

Failing Agricultural Food Crops

- *Objective*: develop crops (for example, rice is the world's food staple) that are robust enough to survive in the extreme and variable environments increasingly prevalent under climate change.
- *Key Conventional Limitations*: can't experiment with large enough numbers of variations of plant genetics and growing conditions to find the sought for plant varieties.
- *System of Systems Approach*: model plants as systems with dynamic linkages among component traits to understand their combined impact on the whole plant system depending on genotypes and weather and soil conditions.
- *Virtual Build and Test*: develop a family of plant model architectures to simulate plant growth and its regulation in a wide range of genetic and environmental parameters, focusing on sensitivity to resource availability.

Catastrophic Forest Wild Fires

- *Objective*: improve forest fire fighting systems to be capable of dealing with the increasingly challenging wild fires that cause millions of dollars in destruction of natural resources and increasingly homes and possessions.
- *Key Conventional Limitation*: too dangerous and costly to create artificial fires of the magnitude and intensity needed to understand dynamics of spread and develop effective containment methods.
- *System of Systems Approach*: model forest topography, weather, and fuel characteristics with embedded sensor networks, connected prediction and decision making components, and agent models of human and robotic fire fighters.
- *Virtual Build and Test*: develop a family of model architectures to simulate forest fire spread and its suppression by a range of tactics and strategies involving prediction of spread and allocation of human and artificial resources.

Lagging Drug Development

- *Objective*: Develop new effective drugs for wide spread diseases and chronic illnesses that reduce the enormous cost of treatment.

- *Key Conventional Limitations*: high capital investment required to develop new drugs in current laboratory environments and numerous sources of error due to poor understanding of the biology of drug action.
- *System of Systems Approach*: model the biology of drug action at the molecular level with reusable components and their interactions.
- *Virtual Build and Test*: develop a virtual environment that allows simulating the effects of drugs on disease analogs thereby moving the search for drugs from the wet lab to a higher speed and more flexible equivalent.

1.1 Virtual Build and Test

Addressing challenges such as the ones just enumerated requires taking a System of Systems approach supported by Virtual Build and Test methodology. *Discrete Event Systems Specification* (DEVS) is a simulation modeling formalism that provides the basis for simulating systems of systems in a virtual environment. DEVS has both system theoretic and information theoretical roots. Just as arithmetic underlies addition, multiplication, and other calculation, so DEVS underlies simulation of discrete event models. DEVS Simulation is performed by an engine that implements a technology-agnostic Abstract DEVS Simulator algorithm. In addition, *System Entity Structure* (SES) is a high level ontology framework targeted to modeling, simulation, systems design, and engineering. An SES is a formal structure governed by a small number of axioms that provide clarity and rigor to its models. The structure supports hierarchical and modular compositions allowing large complex structures to be built in stepwise fashion from smaller, simpler ones.

Simulation models typically employ different formalisms for state and time advance, depending on whether discrete or continuous mechanisms are used. DEVS provides a common programming model of the simulation process that allows a composition of heterogeneous models to dynamically evolve on a common time base in a distributed simulation environment. In such a dynamic composition, each component model generates outputs and consumes inputs produced by others in the proper temporal relationship. This book discusses DEVS-based interoperation through standardizing data distribution and time-management functionalities, implemented over data distribution services middleware such as web services.

1.2 Modeling and Simulation Intrinsic to Virtual Build and Test

Before homing in on the details of the modeling and simulation process itself, we will look at the virtual built and test system development process in more detail. As illustrated in Fig. 1.1, the process typically starts with a phase in which the requirements are developed for the system to be built. However, in the SoS context, this system will not be built from scratch but will be a combination of already existing components and new ones to be constructed. Thus, the end goal must be addressed to both implementing new components and integrating them with existing components to meet the desired functional and performance requirements.

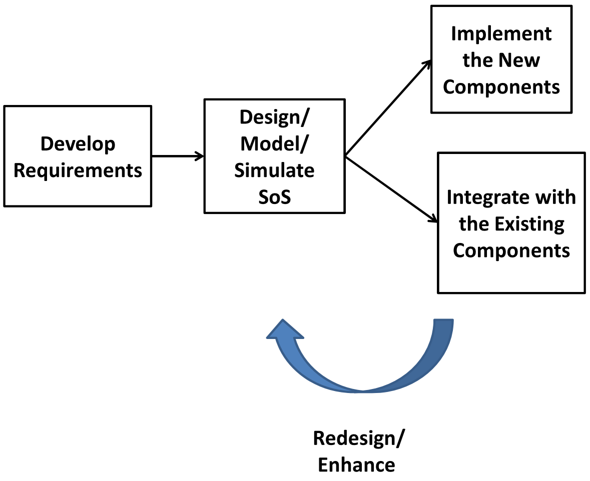
Develop Requirements
Design/ Model/ Simulate SoS
Implement the New Components
Integrate with the Existing Components
Redesign/ Enhance

Fig. 1.1 Centrality of Modeling and Simulation within System of Systems Development

The relative portion of new to existing components offers a way to distinguish different types of SoS problems. For example, designing energy efficient buildings represents a high new-to-existing component ratio where components are manufactured from raw materials and assembled into new structures. On the other hand, designing policies to achieve greater crop yields is characterized by the fact that the new components implementing the policies constitute a relatively small part of the overall agricultural system. Nevertheless, the full range of interventions on reality, from limited management to full scale engineering, are included in the concept of virtual build and test. Most importantly, in all cases there is a need for valid computer representation of the reality in terms of simulation models to support the integration with new components that will enable testing the alternatives before actual implementation and fielding.

In a conventional approach, once we are done designing, modeling and simulating the proposed system we would then construct the new components and integrate them with the existing ones. This follow on phase is like taking a blue print or abstract system specification and implementing it with actual hardware and software. However, this kind of separation between design and implementation does not make full use of the models that have developed in the simulation phase. So there is likely to be a lot of duplicated effort and error that creeps into the implementation. Instead, the modeling and simulation (M&S) environment should help to transfer the models as smoothly as possible from their simulation guises to forms in which they change as little as possible but only the engine in which they are executed changes. Similarly, the integration of new components with existing components should be initially addressed within the simulation phase in a way that does not require starting from scratch later in the implementation phase. This requirement is attainable to the extent that the middleware platform for integrating the components is also the compatible with the engine for executing the new components. The DEVS environment to be discussed in this book offers a solution to meet this requirement by allowing the models developed for simulation to be executed by a real-time engine and interfaced through data distribution middleware to other existing components. In this DEVS-based approach, the new components are designed, developed, and tested as DEVS models within a virtual environment while existing components are represented by stubs or abstractions that provide sufficient fidelity to enable adequate simulation-based testing of the new system. Then the same DEVS models are transferred to the DEVS-based distributed simulation environment with extensions in their message structures to allow them to exchange information in, as well as being time-managed by, the distributed environment. Interfacing to the existing components also occurs by means of the middleware where these real components replace the stubs in the system composition as it was represented in the simulation.

As a consequence of this DEVS-based approach, there is an added benefit to the redesign and enhancement that inevitably occurs within the system's life cycle. Whenever such a modification of the SoS is needed, it can first be expressed and tested within the original simulation environment and then transferred to actual operation in the same way that original implementation was performed. There would be no need to start from scratch to develop a simulation model of the existing system to support the design, development, and testing of the modifications before fielding them.

1.3 Multidisciplinary Collaboration Using Multi-formalism Modeling

A characteristic property of SoS is that the component systems are typically associated with different disciplines and are constructed by expert developers trained in these disciplines. As a consequence, model components reflect diverse world views and heterogeneous formalisms that must be integrated together in the composite model of the SoS. As a universal computational basis for systems theory, DEVS offers a single common framework for accommodating a limitless variety of domain-specific modeling formalisms. A DEVS-based environment offers a practical means of supporting interdisciplinary team or community-developed libraries of heterogeneous model components that can be integrated together to construct the virtual models. Moreover, the System Entity Structure offers a structured means of organizing such model repositories to help compose the right combinations of alternatives to satisfy the particular objectives of the moment.

In a truthful disclaimer, the vision of a DEVS-based environment with all the functionality just depicted is still on the way toward full realization. Nevertheless, progress has been made and environments exist that show the feasibility of the approach and suggest where further research and development is needed in more specifics.

We discuss three DEVS modeling and simulation environments for SoS virtual build and test in this book.

MS4 Me™ is a modeling and simulation environment developed as the first in a commercial line of DEVS products (ms4systems.com). MS4 Me is aimed at a variety of users such as students, managers, modelers, developers, and programmers enabling them to work at the level for which they are most comfortable and productive. MS4 Me employs a DEVS simplification, Finite Deterministic DEVS (FD-DEVS), as the basis for a simplified syntax/content assisted language that beginners and non-programmers can employ to quickly and easily construct basic DEVS models. These models of component systems can then be coupled together to create a simulation with a few natural language statements for the System Entity Structure. Once the basic outlines of the model have been established, the environment features a structured approach for DEVS experts and Java programmers to enhance the FDDEVS descriptions with the depth needed to actually simulate real systems of systems. A notable feature intended to accelerate such a development process is the Sequence Designer tool which automatically creates both FDDEVS models and an SES to couple them together from a simple sequential diagram input. Chapters 2 through 12 of the book are devoted to DEVS concepts as implemented in MS4 Me with the remaining chapters devoted to expositions of how these concepts apply to SoS. The following DEVS modeling environments are discussed in depth to illustrate their powerful features in this regard.

Component-Based System Modeler and Simulator (CoSMoS) offers an integrated platform for developing component-based, modular, hierarchical families of models. It is grounded in a unified logical, visual, and persistent perspective on models. It supports specifying families of parallel DEVS, Cellular Automata, and XML-Schema models. Its unique features are storing models in relational databases, model complexity metrics, and separating simulatable and non-simulatable models from one another. Systems of systems may be modeled as separate software and hardware systems which can be then used together to model alternative system architectures. The CoSMoS lifecycle process affords basic capabilities starting from model conceptualization and ending with simulation execution. It integrates *DEVS-Suite* simulator, which supports developing and execution of hierarchical parallel DEVS models. The simulator is built using DEVSJAVA and DEVS Tracking Environment with configurable synchronized control for simulation execution and viewing. It supports automating design of experiments in combination with animation of simulation execution, viewing time-based data trajectories at run-time, and storing simulation data for post processing. Model libraries for Semiconductor Supply-Chain Manufacturing, Service-Oriented Computing, MIPS32 Processors, Network-on-a-chip, Swarm Net, UML design patterns, and Household Agents have been developed. The simulator is in use in academic institutions in many countries and several research centers in government agencies and commercial entities. CoSMoS and DEVS-Suite are featured in Chaps. 14 through 16.

Virtual Laboratory Environment (VLE) is aimed at collaborative development of simulation models for Living Systems of Systems. The environment employs DEVS to combine traditional continuous models with discrete event and other modern formalisms to encourage collaborative development, validation, replication, and experimentation within a virtual laboratory context. The goal is to support decision making in SoS applications such as control of epidemic outbreaks among animals (remember the avian flu?) and reengineering of rice varieties to withstand loss of viability. The French National Institute for Agricultural Research (INRA), one of the world's elite agricultural institutes, chose VLE to integrate its stock of existing agriculture models and to develop new simulation capabilities using DEVS. As discussed in Chap. 17, the RECORD project documented why INRA chose DEVS and VLE to support model reuse and collaboration among its model developers. The DEVS/VLE combination was preferred among many commercial and academic contenders based on VLE's implementation of DEVS's formal support for coupling and integration of models in the diverse formalisms being employed in the agricultural domain.

1.4 Background in the Literature

Since the appearance of one of the author's (Zeigler) "Theory of Modeling and Simulation" in 1976, the DEVS approach to modeling and simulation it introduced has taken root in academia and is emerging into common research and industrial use. Since it is aimed at more of a guide to the use of DEVS concepts and tools, this book cannot replicate the background theory in full within the space available. Thus, we refer you, the reader, to consider acquiring the background from some of the following books:

- Systems of Systems—Innovations for the 21st Century, Edited by Mo Jamshidi, Wiley, 2008.
- Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Agents, by Bernard P. Zeigler, Academic Press, Orlando, 1990.
- Multifaceted Modeling and Discrete Event Simulation, by Bernard P. Zeigler, Academic Press, London, 1984.
- Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems, by Bernard P. Zeigler, Herbert Praehofer, and Tag Gon Kim, 2nd Edition, By, Academic Press, NY, 2000.
- Modeling and Simulation-Based Data Engineering: Introducing Pragmatics into Ontologies for Net-Centric Information Exchange, by Bernard P. Zeigler and Phillip E. Hammonds, Academic Press, NY, 2007.
- Discrete Event Modeling and Simulation Technologies: A Tapestry of Systems and AI-Based Theories and Methodologies, Editors: Hessam S. Sarjoughian and François E. Cellier Springer-Verlag Publishers, 2001.
- Discrete-Event Modeling and Simulation: A Practitioner's Approach (Computational Analysis, Synthesis, and Design of Dynamic Systems) by Gabriel A. Wainer, CRC Press, 2009.
- Discrete-Event Modeling and Simulation: Theory and Applications (Computational Analysis, Synthesis, and Design of Dynamic Systems), Editors: Gabriel A. Wainer, Pieter J. Mosterman, CRC Press, 2010.
- Building Simulation Software: Theory, Algorithms, and Applications, by James Nutaro, Wiley Publishers, NY, 2010.

- DEVS Net-Centric System of Systems Engineering with DEVS Unified Process. CRC-Taylor & Francis Series on System of Systems Engineering, by Saurabh Mittal and José L. Risco-Martín (to appear).
- Agent-Directed Simulation and Systems Engineering Editors: Levent Yilmaz and Tuncer Ören, Wiley, 2009.

1.5 Guide to Modeling and Simulation of Systems of Systems

The present book builds upon the material in the earlier books but goes beyond them in several critical ways. It centers on the unifying theme of "virtual build and test" as a means of integrating and providing context to the various technical concepts and tools discussed. In doing so, it provides an inclusive exposition of the many aspects of DEVS-based concepts and tools, all relating back to the "virtual build and test" theme. In particular, Chaps. 2–8 offer a step-by-step introduction to DEVS concepts and corresponding MS4 Me features that enable you to gain hands-on experience with the concepts to build sophisticated SoS models. A User Reference to the features of MS4 Me accompanies this book and is also sold by the publisher. The software itself is available from MS4 Systems (http://ms4systems.com). Chapters 9–12 develop more advanced concepts for modeling and simulation of SoS and illustrate them with MS4 Me features developed earlier. Chapters 13–18 discuss applications of the concepts to virtual build and test for a variety of SoS application domains using the capabilities of CoSMoS/DEVS-Suite and VLE as well as MS4 Me. The software packages are available at http://acims.asu.edu (CoSMoS/DEVS-Suite) and http://vle-project.org (VLE). Exercises to reinforce the concepts, to encourage using the three sets of tools, and to compare their capabilities are provided throughout the book.

The primary target for this book is both practitioners and academics (professors and students). Practitioners include simulation software developers supporting system development, systems engineers designing and architecting systems, and managers of such projects. The technical level targeted at first or second year graduate students is augmented with introductory and summary sections written at a more general level so that managers and others can skim over technical parts.

Hopefully, the advances described here will inspire continued research and development of the DEVS framework that builds upon the concepts and tools we present.