

Chapter 2

Service-Oriented Model Engineering and Simulation for System of Systems Engineering

Bernard P. Zeigler and Lin Zhang

2.1 Introduction

The model is the foundation for simulation activities (Ören et al. 1982), especially in regard to system of systems (SoS, a composition of systems which component systems have legacy properties). A valid model and correct simulator are necessary for obtaining simulation results that serve the intended use of the model (Zeigler et al. 2000). Verification, validation, and accreditation (VV&A) is the primary means of establishing the credibility of the simulation results (Pace 2004). VV&A is usually considered after model construction and involves calibration and/or validation of the established model in order to determine whether it is credible. This post-construction determination has important implications to discover model problems and defects, but it cannot solve the problem of how to get a correct model in the first place. Especially for complex systems, due to the complexity and uncertainty of the system, the modeling process can be very complicated, which makes VV&A of a model extremely difficult. Even if defects are found via VV&A, revision of the model will be very difficult and costly.

More importantly, for a system of systems, to construct a valid model is just the first step since a model of a SoS generally experiences a long term of evolution and management. As a result, the key issue for a complex system model is to guarantee the credibility of the full model life cycle with minimum cost.

To meet the challenges in development and management of the SoS model, this chapter introduces the concept of the *model engineering* (Zhang 2011; Zhang

B.P. Zeigler (✉)

RTSync Corp., Arizona Center for Integrative Modeling and Simulation,
Sierra Vista, AZ, USA
e-mail: zeigler@rtsync.com

L. Zhang

School of Automation and Computer Science, Beihang University, Beijing, China

et al. 2014a), which aims at setting up a systematic, normalized, and quantifiable engineering methodology by exploring basic principles in model construction, management, and maintenance to manage the data, processes, and organizations/people involved in the full life cycle of a model to guarantee the credibility of its life cycle. Meanwhile, in recent years, service-oriented technology has been widely used in software intensive systems, as well as model construction and management of complex system simulation. Therefore, we will show how modeling engineering can take advantage of service-oriented technology to provide an efficient way of building and managing the model of a system of systems.

2.2 Some Related History

It helps to recount some history relevant to *service-oriented model engineering* (SOME) as appropriate to a volume dedicated to Tuncer Ören's 80th birthday. As early as 1973, Ören was expressing his normative views for modeling and simulation (M&S) methodologies (Ören 1973) and recently published a treatise on the synergies of simulation, agents, and systems engineering (Ören and Yilmaz 2012). Many of his views on these synergies are covered in the book with Yilmaz on *Agent-directed Simulation and Systems Engineering* (Yilmaz and Ören 2009).

As related by Ören and Zeigler (2012), Ören received his Ph.D. in systems engineering under the supervision of A.W. Wymore. His Ph.D thesis was greatly influenced by Wymore's axiomatic approach for his systems theory (Wymore 1967). Moreover, Ören's mechanical engineering background allowed him to appreciate the vital importance of developing software tools for M&S (Ören 1990). He has always been interested in learning, conceiving, and developing methodologies suitable for complex problems (especially for social problems) which are inherently nonlinear in nature. As part of his Ph.D. requirements, in the late 1960s, he developed a simulation model specification language called GEST (General System Theory implementer) (Ören 1971) based on Wymore's book (Wymore 1967). Part of his aim was to use a translator to generate a simulation program in a language which could be compiled or interpreted. Such translators were implemented later by his students. GEST's model specification language is based on Wymore's concept of systems composed of component systems and couplings that all components to exchange information through input and output ports. Since component systems and coupling recipes were already defined by Wymore, in set-theoretic notation, Ören concentrated on ease of robust specification and readability and avoided any set-theoretic representation.

Over the years, the scope of Ören's concerns has broadened to formulate a body of knowledge for M&S expressed in many publications and presented in detail in Ören (2005, 2014), where he describes a paradigm shift from use of the term M&S to the term simulation systems engineering (SSE): "In the early days, only very few were referring to M&S. Afterwards, to stress modeling process and the associated activities and environments, the term M&S is used by large number of

simulationists. Currently, a very commendable shift of paradigm is being adopted to cover all aspects of simulation studies. This is to conceive M&S –within a larger perspective– as the Simulation Systems Engineering (SSE).” (Slight paraphrase of (Ören 2005))

In this article, we take a similarly broad perspective and probe the nature of model engineering in the context of systems engineering, particularly for systems of systems (SoS) and implemented with service orientation environments. To establish the background needed for this discussion, we briefly introduce the definition and theory of systems of systems as it relates to M&S, in particular to the discrete event system specification (DEVS) formalism for M&S. We then define, and examine in depth, model engineering for SoS which has deals with the full model life cycle. With these concepts as foundation, we analyze the services necessary to support model engineering and the requirements for design of a service-oriented model engineering and simulation environment. Consideration of the results of research in DEVS then enables us to give a more concrete characterization of such an environment. We close with a discussion of how model engineering and DEVS enable new frameworks for application areas and the opportunities for further research.

Some short definitions of terms we employ as initial concepts are drawn from Waite and Ören (2009):

- Body of Knowledge (BoK) – The set of justified true beliefs and competencies – explicit and implicit – that defines a discipline, practice, role, or field of endeavor
- Referent – n. Something referenced or singled out for attention, a designated object, real or imaginary, or any class of such objects
- Model – n. The representation of some referent
- Simulation – n. A mechanization of a model’s evolution through time

Although definitions are still in flux, for our purposes, *service-oriented model engineering* is a form of model engineering that is based on a service approach to computation, and simulation systems engineering is an inclusive term that includes model engineering.

2.3 Theory of Systems of Systems

In systems theory as formulated by Wymore (Ören and Zeigler 2012), systems are defined mathematically and viewed as components to be coupled together to form a higher-level system.

As illustrated in Fig. 2.1, Wymore’s (1967) systems theory mathematically characterizes:

- *Systems* as well-defined mathematical objects characterizing “black boxes” with structure and behavior.

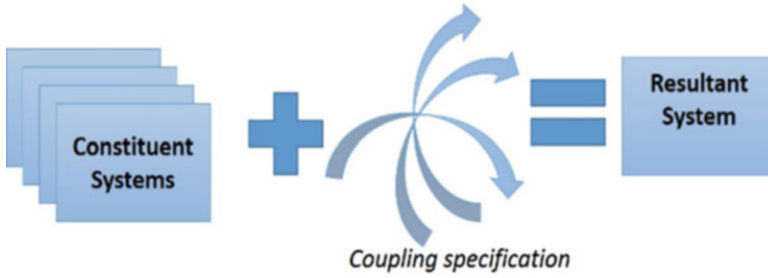


Fig. 2.1 Wymore's system composition

- *Composition of systems* – constituent systems and coupling specification result in a system, called the resultant, with structure and behavior emerging from their interaction.
- *Closure under coupling* – the resultant is a well-defined system just like the original components.

2.3.1 System of Systems

As illustrated in Fig. 2.2, a system of systems (SoS) is a composition of systems, where often component systems have legacy properties, e.g., autonomy, belonging, diversity, and emergence (Boardman and Sauser 2006). In this view, a SoS is a system with the distinction that its parts and relationships are gathered together under the forces of legacy (components bring their preexisting constraints as extant viable systems) and emergence (it is not totally predictable what properties and behavior will emerge). Here in Wymore's terms, *coupling* captures certain properties of relevance to coordination, e.g., connectivity, information flow, etc. *Structural and behavioral properties* provide the means to characterize the resulting SoS, such as fragmented, competitive, collaborative, coordinated, etc.

The main difference between SoS and general system composition is worth noting. SoS generally refers to systems composed of components that are already in existence and bring certain legacy properties “to the table” when placed into a new composition. This is in contrast to general system composition where components may be built from scratch for the distinct purpose of the new composition. This implies that in the SoS case, a key feature is that the compositions require integration and/or coordination to overcome the features and goals of the existing systems that don't align well with the new system goals. However, we remark also that the term SoS is still in flux and may sometimes mean complex systems whose components themselves are complex systems. Depending on the definition of “complexity” in this context, the two meanings may actually coincide.

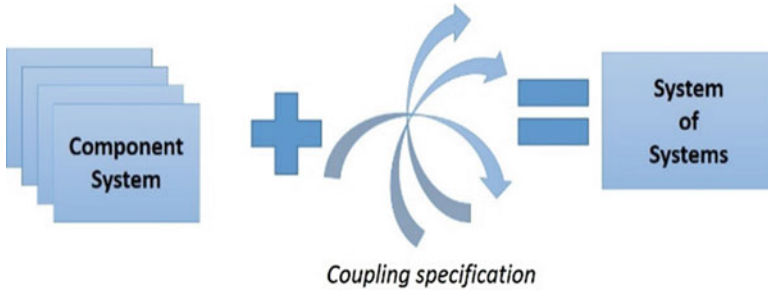


Fig. 2.2 System of systems

2.3.2 *Discrete Event Systems Specification (DEVS)* *Formulation of SoS*

The DEVS formalism (Zeigler et al. 2000), based on systems theory, provides a framework and a set of M&S tools to support systems concepts in application to SoS engineering (Mittal and Martin 2013). A DEVS model is a system-theoretic concept specifying inputs, states, and outputs, similar to a state machine. Critically different, however, is that it includes a time-advance function that enables it to represent discrete event systems, as well as hybrids with continuous components, in a straightforward platform-neutral manner. DEVS provides a robust formalism for designing systems using event-driven, state-based models in which timing information is explicitly and precisely defined. Hierarchy within DEVS is supported through the specification of atomic and coupled models. Atomic models specify behavior of individual components. Coupled models specify the instances and connections between atomic models and consist of ports, atomic model instances, and port connections (ports and connections are not shown here for simplicity). The input and output ports define a model's external interface, through which models (atomic or coupled) can be connected to other models.

As illustrated in Fig. 2.3, based on Wymore's systems theory, the DEVS formalism mathematically characterizes the following:

- DEVS Atomic and Coupled Models specify Wymore systems.
- *Composition of DEVS models* – component DEVS and coupling result in a Wymore system, called the resultant, with structure and behavior emerging from their interaction.
- *Closure under coupling* – the resultant is a well-defined DEVS just like the original components.
- *Hierarchical composition* – closure of coupling enables the resultant-coupled models to become components in larger compositions.

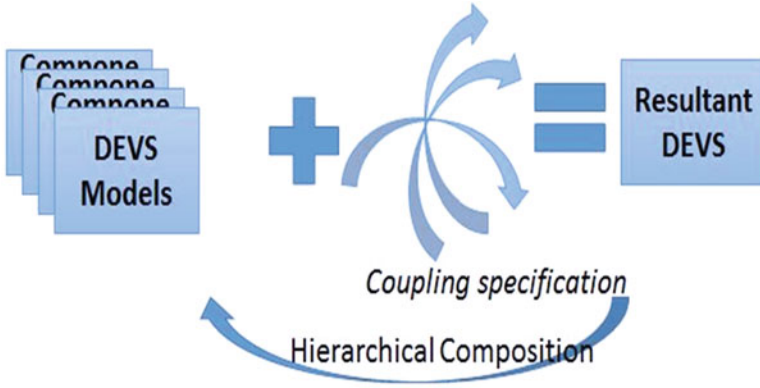


Fig. 2.3 DEVS formulation of systems of systems

2.4 Model Engineering for SoS

A model is an abstract expression of objects to study and embodies high intelligence of human beings in recognition of the world. With continuous development of science and technologies, the model is becoming more and more important. It refers not just to the process of modeling but also to the life cycle of model.

2.4.1 *The Life Cycle of a SoS Model and Related Works*

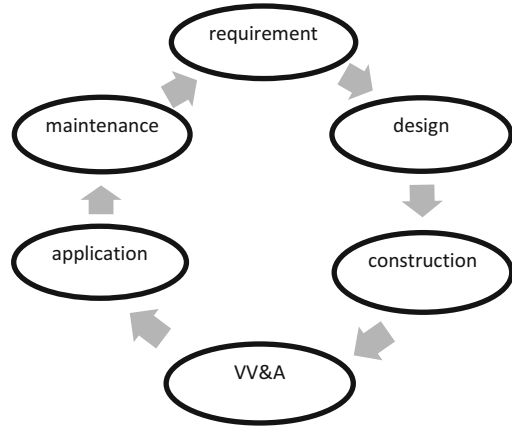
Generally, a model experiences requirement analysis, model design, model construction, model verification and validation (VV&A), model application, and model maintenance (Zhang 2011). These processes compose a complete life cycle of a model as is shown in Fig. 2.4.

How to build a right model is the core issue in simulation. A large number of research achievements on models have been obtained in the past dozens of years. These achievements are related to different phases in a model life cycle, e.g., modeling theory and method, VV&A, and model management.

The life cycle concept has not been emphasized enough in the simulation domain, and related research and applications are not sufficient (Balci 2012). Fishwick (1989) called the simulation model development process as simulation model engineering to emphasize engineering feature of the model development process, but no special explanation on its meaning was given, and no systematic method system was established.

In recent years, the international simulation community has become conscious of the unfavorable influences of missing foundational theory of M&S on the development of simulation curriculum. As a result, research on the M&S life cycle management is gradually attracting attention from academic circles. Radeski and

Fig. 2.4 The life cycle of a SoS model



Parr (2002) proposed the modeling simulation life cycle model framework, which defined organization mode and structure of the modeling simulation process, work products, quality assurance, and project management, and described the features and requirements of the life cycle phases such as development, use, maintenance, and reuse of the modeling simulation system. References (Fishwick 1990; Abdouni Khayari et al. 2010) achieved some valuable results in the model life cycle management and developed a model prototype management system, which provided valuable reference to model development for designers.

2.4.2 *Challenges in Development and Management of SoS Models*

As can be seen from the related work, current research on models generally focused on one phase in the model life cycle and is separate and diffuse. Although importance of the engineering idea is gradually recognized in applications of the full model life cycle, currently no complete theory and technology system and philosophy are available. So there are still lots of challenges in the life cycle of the model of SoS. As pointed in (Zhang et al. 2014a), some reasons for this situation are:

1. High complexity of the referent SoS – Firstly, a SoS is composed of various component systems, and the relationship between them is very complicated. Secondly, generally a complex system is dynamic, variable, and very uncertain. Thirdly, SoS generally performs emergent behaviors. These features make a SoS very complicated. The complexity of SoS leads to the complexity of the model itself.
2. The long life cycle of a SoS – With passage of time, the models should be continuously improved and changed. Different model versions are available. Each version may be applicable to different application phases. Different

versions and application phases of multiple models compose a complicated network. How to keep consistency and credibility of different parts and versions of the model is the key for model maintenance.

3. Model heterogeneity – A SoS is composed of many heterogeneous component models. Heterogeneity of models generally comes from different development organizations, different platforms and architectures, different development languages and databases, etc. Heterogeneity brings big challenges to integration and maintenance of the system models.
4. Complicated evolution of models – Generally, a SoS is in continuous evolution, so the models will be continuously adjusted and changed. Changes of different relations are very complicated in evolution due to system complexity, so the model elements and its relation should be completely tracked and managed to guarantee correctness of the model evolution.
5. Difficult model reuse – With growth of complexity of the systems to study, the roles and values of model reuse are very remarkable in model development and use of a SoS (Liu et al. 2008). Generally, a SoS includes multiple combined systems. A huge number of system models in past research and development practices have been accumulated. Correct and efficient reuse of models will reduce model development cost, greatly shorten development time, and effectively improve model credibility. Although some research on model reuse has been conducted, no efficient and practicable model reuse method is available now.
6. Massive processing data – Generally, a SoS entails a large amount of data to process, including the required modeling data, data generated in modeling, and data generated in the modeling process. Data processing includes data storage, inquiry, exchange, management, understanding, analysis, and mining, which bring many challenges.
7. The multidisciplinary collaborative model development – Collaborative model development is associated with different steps in the whole model life cycle. Collaboration is required on different phases, e.g., collaborative requirement analysis, collaborative design, and collaborative validation. All this work composes a huge engineering requirement and should be supported by appropriate management tools.
8. Higher requirements for system performance – Compared to a simple system, a SoS requires higher performance, e.g., higher requirements for reliability, security, credibility, cost, and energy saving. To guarantee that these performance requirements are met, special means should be required to analyze and process the models.

2.4.3 Meaning of Model Engineering

2.4.3.1 Concept of Model Engineering

Based on the state-of-the-art researches on the model, a systematic methodology was proposed to cope with challenges in model life cycle management of a SoS (Zhang 2011; Zhang et al. 2014a). The model development and management activities change from a spontaneous and random behavior to conscious, systematic, standardized, and manageable behavior by constructing a model engineering theory and methodology system in order to guarantee credibility of different model phases.

Zhang (2011; Zhang et al. 2014a) gave a definition of model engineering as follows:

Model engineering is defined as a general term for theories, methods, technologies, standards, and tools relevant to a systematic, standardized, quantifiable engineering methodology that guarantees the credibility of the full life cycle of a model with the minimum cost.

Here, *model engineering* involves the following meaning (Zhang 2011; Zhang et al. 2014a):

1. Model engineering regards the full life cycle of a model as its object of study, which studies and establishes a complete technology system at the methodology level in order to guide and support the full model life cycle process such as model construction, model management, and model use of a SoS.
2. Model engineering aims to ensure credibility of the full model life cycle; integrate different theories and methods of models; study and find the basic rules independent of specific fields in the model life cycle; establish systematic theories, methods, and technical systems; and develop corresponding standards and tools.
3. Model engineering manages the data, knowledge, activities, processes, and organizations/people involved in the full life cycle of a model and takes into account time period, cost, and other metrics of development and maintenance of a model.
4. Here, the model credibility is a comprehensive indicator and includes factors such as availability, accuracy, reliability, and quality of service (QoS).

2.4.3.2 Key Technologies of Model Engineering

As described in the above part, current research on the technologies related to the full model life cycle is preliminary and diffuse. For comprehensive and systematic application and implementation of model engineering, many key technologies should be studied (Zhang et al. 2014a). These technologies can be divided into six categories as shown in Fig. 2.5.

1. General technologies

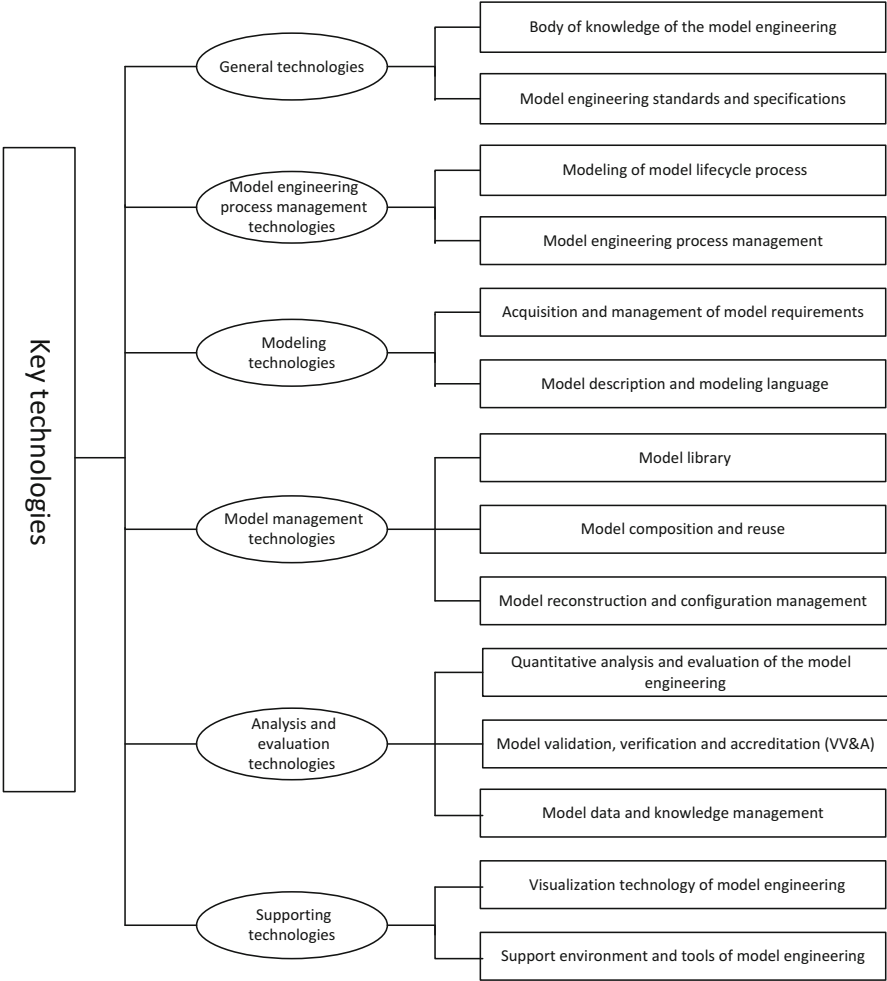


Fig. 2.5 Key technologies of model engineering

- **Body of knowledge of the model engineering:** The body of knowledge system (BoK) includes the concepts and terminologies involved in a specific research field. The model engineering BoK identifies the research scope of the model engineering and its boundary and relationship with other related subjects. Establishment of systematic and complete BoK requires long-term accumulation and extraction.
- **Model engineering standards and specifications:** Standards are the basis for the implementation of the model engineering. During the life cycle process of a model, each activity requires corresponding standards, including model development process, model description, model component interface,

model storage, model data exchange, model interoperation, model service, model maintenance, etc.

2. Model engineering process management technologies

- **Modeling of model life cycle process:** The life cycle model of the model engineering aims to identify the structural framework of activities involved in model construction and management (Zeigler et al. 2000), which is the methodology to guide the model engineering, and ensure improvement of model quality and development efficiency and reduction of full model life cycle cost. Proper process models and corresponding implementation methods can be proposed by referring to the existing achievements in the system engineering and software engineering and other relevant fields and combining the model development features of SoS.
- **Model engineering process management:** The data, knowledge, tools, persons/organizations, and technologies in the full model life cycle should be effectively managed with the model life cycle process model as the guide, with standards and specification as the basis, and with the project management methods and means as reference in order to get the dependable model with the minimum cost. The model maturity definition and control, performance management, flow monitoring and optimization, risk control, and cost control are important in model engineering process management.

3. Modeling technologies

- **Acquisition and management of model requirements:** Accurate requirement acquisition is the key in modeling. Requirement acquisition and management is very challenging due to uncertainty and ambiguity of SoS. Requirement acquisition studies to extract, describe, parse, and validate requirement via automated or half-automated means. Requirement management studies how to reflect the changing requirements in the model construction and maintenance accurately and timely.
- **Model description and modeling language:** Generally, a SoS contains multiple different systems with different properties such as qualitative systems, quantitative systems, continuous systems, discrete event systems, deterministic systems, uncertain systems, etc. One of the core issues in model development of SoS is how to take advantage of effective ways to describe the whole system. Therefore, it is required to study corresponding model description mechanism and structure and develop generic or specific description languages according to the characteristics of the various systems.

4. Model management technologies

- **Model library:** The model library is the foundational platform to carry out model management and perform standardized encapsulation, storage, and query for the models (Ören and Zeigler 1979). The complicated applications such as model reuse, combination, and configuration management can be based on the model library. Traditional database technology, service-oriented

technology, and cloud-computing technology can support construction and management of the model library.

- **Model composition and reuse:** The model composition and reuse is an important technology to improve model construction and maintenance efficiency and improve model credibility of SoS. It mainly studies how to use the existing model components to quickly and correctly compose complicated models according to the system requirements and includes standardized encapsulation of model components, intelligent model matching, model relation management, dynamic model combination, model consistency validation, and model service.
- **Model reconstruction and configuration management:** The requirements for model functions and performances change due to diversified requirements inside and environmental uncertainty, so the models should be quickly reconstructed or configured. The model reconstruction aims to adjust the internal structure without change of the main external functions of models, further optimize the model performance, and ease its understanding, maintenance, and transplant of models. Model configuration can adapt different requirements or change of models in function and performance by adjusting and optimizing internal components and parameters. For SoS model engineering, model reconstruction and configuration management are very important and challenging.

5. Analysis and evaluation technologies

- **Quantitative analysis and evaluation of the model engineering:** The quantitative analysis is one of main features of the model engineering. To ensure credibility of the full model life cycle, many steps should be analyzed, evaluated, and optimized in a quantitative manner, e.g., complexity analysis and evaluation of model development process, cost and benefit analysis and optimization, risk analysis and control, model availability and reliability analysis, and model service quality analysis.
- **Model validation, verification, and accreditation (VV&A):** The model VV&A technology is one important part in the model engineering. Although some rich research achievements have been achieved, they cannot meet the actual requirements of modeling simulation of SoS. Most research focuses on qualitative analysis, and quantitative and formalized analysis methods are lacking, so VV&A technology, especially VV&A quantitative analysis, and formalized analysis technology are still a main research focus in the model engineering.
- **Model data and knowledge management:** Many SoS models contain voluminous data to process. Some models are constructed based on massive data, or even exist in the form of data and their relations. Data management aims to effectively organize and use the data, especially massive data, and plays a key role in quantitative analysis of the model engineering. On the whole, the knowledge is divided into two classes in the model engineering. The class 1 indicates the knowledge in the model, e.g., some qualitative models include

massive knowledge rules. Another class indicates the knowledge on model development and management and generally includes experiences accumulated and extracted by developers and users in practices. Different knowledge should be managed and used in different manners to improve model quality and intelligence and automation of model construction and maintenance.

6. Supporting technologies

- Visualization technology of model engineering: Visualization technology can be used on different phases of the model engineering, can realize transparent model development and management process, facilitate understanding and monitoring, and improve human-machine interaction efficiency. The visualization technology plays an important role in the model engineering.
- Support environment and tools of model engineering: Implementation of the model engineering requires an integrated support environment and corresponding support tool to support different activities of model engineering, e.g., network collaboration, requirement management, process model construction and maintenance, model library management, qualitative and quantitative analysis and evaluation, data integration, knowledge management, model validation, and simulation experiment.

2.4.4 *Body of Knowledge of Model Engineering*

Model engineering is the resultant of fusion of many crossing subjects including the software engineering, system engineering, computer science and engineering, mathematics, system M&S, knowledge engineering, project management, quality management, and related application fields. Based on the body of knowledge in these disciplines, specific BoK of the model engineering is formed according to the requirements and features of the model engineering.

To establish the model engineering BoK, it is necessary to tease out the involved knowledge system in a systematic manner, extract features closely associated with the model-related activities from related fields, and summarize and condense those specific development technologies and management means of the model life cycle process. A preliminary BoK framework of the model engineering was given by Zhang (2011, 2014a). Two aspects are mainly considered in this process.

1. Identify the horizontal crossing relations between the model engineering and other closely associated subjects, properly tailor their overlapping parts, and make these overlapping parts reflect specific features of the model engineering.
2. Identify the modules in the model engineering system and vertical hierarchical relations and horizontal interface relations between modules, make the framework compose an organic whole, and serve for the full life cycle process of the model.

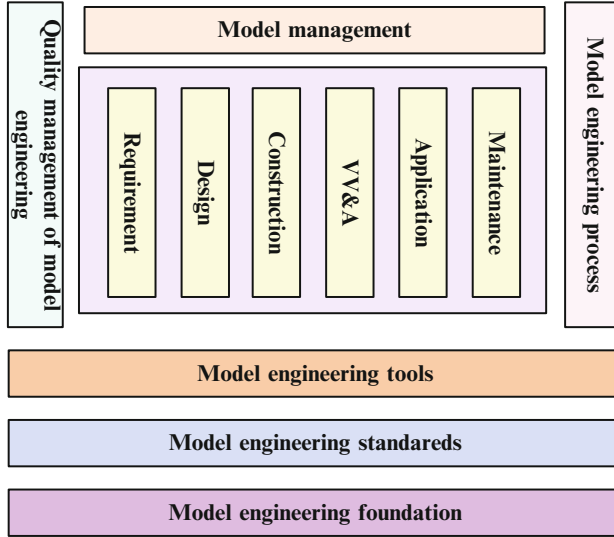


Fig. 2.6 The BoK framework of the model engineering

By introducing the model engineering BoK framework, we hope to reach the following targets:

1. Promote consistent opinion within academic circles on the meaning of model engineering
2. Identify the research scope of model engineering
3. Relate the position of model engineering to other subjects such as software engineering, system engineering, computer science, and mathematics and set their boundaries.

A BoK framework of model engineering is shown in Fig. 2.6 (Zhang 2011; Zhang et al. 2014a). The BoK framework of the model engineering is divided into five parts:

- Part 1: Foundation: including the basic concepts and terms, methodology, technical system, etc. It provides the basic guidance for the implementation of the model engineering and also is the foundation and guarantee of the model engineering independent of other subjects.
- Part 2: Model life cycle: it describes different phases of the full model life cycle at the technical level. This part modeling requirement, model design, model construction, model VV&A, model application, model maintenance.
- Part 3: Implementation and management of the model engineering: it includes the process management quality management of model engineering and model configuration management. All activities in the full model life cycle are managed and controlled implementation, process, and quality.

- Part 4: Model engineering tools: it provides the necessary software tools for the implementation and application of the full life cycle of the model engineering.
- Part 5: Related standards of model engineering: it includes rules, protocols, or specifications, which are necessary for implementation of model engineering and development of related tools.

The detailed contents of each part can be found in Zhang et al. (2014a).

2.5 Service-Oriented Model Engineering and Simulation Environment

The construction and management based on model engineering are shown in Fig. 2.7 (Zhang et al. 2014b). Taking modeling as an example, a simulation scenario is given, then multiple subtasks of the system are formed automatically according to the scenario, and automatic matching between tasks and processes is completed in the model engineering platform, so a new model is built. This just-built model can be added into the model library as a case; therefore, the model library is enriched. The use, management, and maintenance are completed by the full life cycle of model engineering.

2.5.1 *Architecture of Service-Oriented Model Engineering and Simulation Environment*

Service-oriented technology is one of the most powerful and popular technologies to the development, management, and integration of software intensive systems and has been widely applied to lots of different domains.

A service-oriented model engineering and simulation environment is a kind of software to support the implementation of model engineering (Zhang et al. 2014b) and simulation (Fig. 2.8). There are five layers including model component layer, model service layer, model management layer, simulation layer, and application layer. The functions of each layer are as follows:

1. *Model component layer*: there are various models, such as qualitative model, quantitative model, linear model, and nonlinear model. Meanwhile, these models are provided by different organizations and developers, which can lead to model heterogeneity, so model component layer is needed to classify and organize models.
2. *Model service layer*: this layer is a process of model normalization. It provides interface specifications among models and conducts unified service encapsulation and transformation (e.g., service-agent modeling (Si et al. 2009; Liu

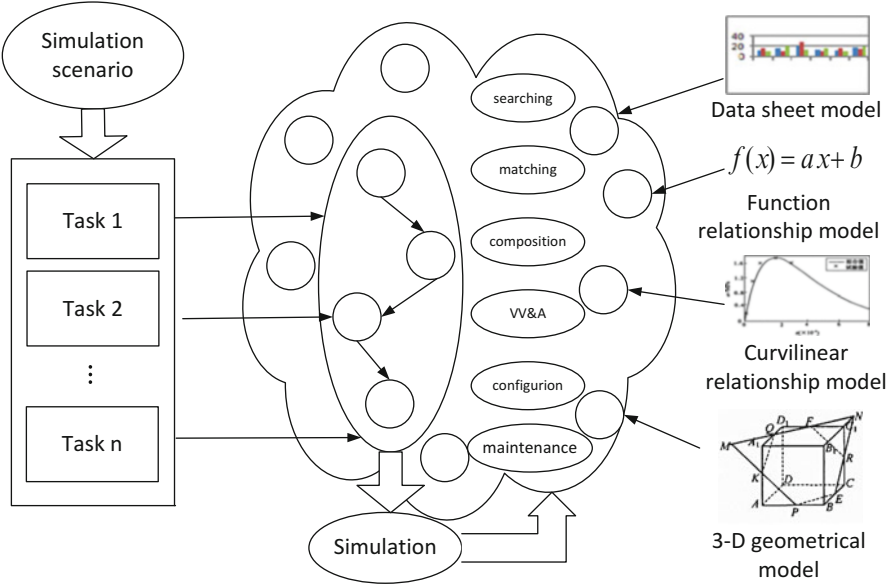


Fig. 2.7 Construction and management based on model engineering

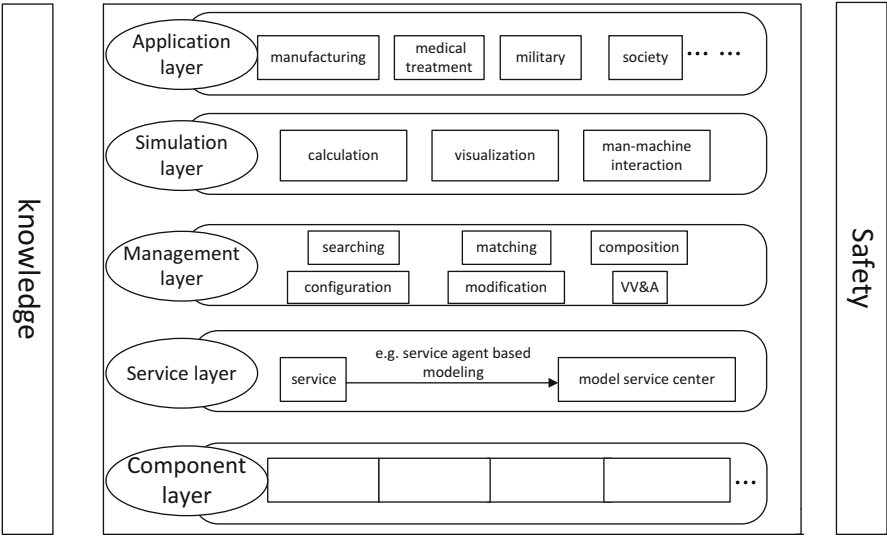


Fig. 2.8 Architecture of service-oriented model engineering and simulation environment

et al. 2014) is a kind of method), and then the standardized model is put into the model service center.

3. *Model management layer*: model management layer executes management and operation of models in the model service center, such as searching, matching,

composition, configuration, modification, VV&A, etc. Fast and accurately matching among models is guaranteed by management, so requirements of M&S can be satisfied.

4. *Simulation layer*: simulation layer mainly consists of simulating calculation, visualization, man-machine interaction. Simulating calculation means getting simulation results under the help of software and hardware. Visualization technology can realize transparent model development and management, as well as facilitate the process of understanding and monitoring; man-machine interaction can support different types of interactions.
5. *Application layer*: different kinds of applications can be carried out with the help of model engineering. These applications can include manufacturing, medical treatment, military, environment, society, etc. This reflects the value of model engineering itself and its contributions to society development.

2.5.2 Implementation of a Service-Agent-Based Model Engineering Supporting Environment

According to the idea of model engineering, the elements in a model library should have the characteristics of service oriented, intelligence, standardization, etc.

2.5.3 Encapsulation of Model Components

To achieve this purpose, we use service-agent (SA) that was proposed in Si et al. (2009) to encapsulate component models in the model library. Service-agent is a combination of service and agent, which can make service with agent characteristics (Si et al. 2009; Liu et al. 2014) and will be well suited to features of SoS. The structure of a SA is shown in Fig. 2.9.

A SA has several features (Liu et al. 2014): (1) A SA is an autonomous entity which observes and acts upon an environment and directs its activity toward achieving goals. (2) A SA pertains to SOA standard with XML-based protocols such as WSDL, SOAP, etc. (3) A SA has states, e.g., working state, prepared state, waiting state, and searching state.

After component models are encapsulated into SAs, the process of modeling of a complex system can be transferred into the process of composition of SAs in the model library. Theoretically, the composition of component models with SAs can be automatic and have the ability to be self-adapted to the uncertainty of SoS.

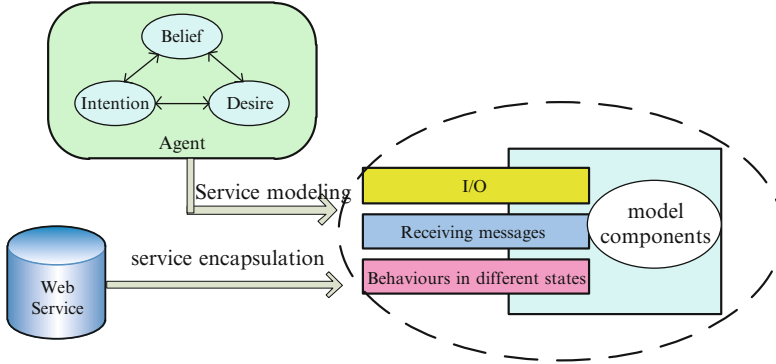


Fig. 2.9 Service-agent (SA) structure

2.5.3.1 Specifications of Service Agents

Specifications of SA are basis of communication, interaction, and composition among SAs. A set of SA specifications were given by Zhang et al. (2014b). The SA specifications are described as three parts: interface specifications, architectural specifications, and implementation specifications.

1. *Interface specifications*: SA external interface is either java component interface or web service, which can support integration of local area network (LAN) and wide area network (WAN). The purpose of user-oriented interface layer specification is to unify the descriptions of components and build standard calling interfaces for components. Component interface follows the principle of service orientation, which let users pay attention to functions provided by components, rather than the internal structure and state of components.
2. *Architectural specifications*: SA has the characteristics of environment awareness and special communication interface, which general simulation components do not have. These reflect the intelligence of SA. Architectural layer is designed to support the underlying services for interface layer, while conducts constraint and guidance as a component for realizing infrastructure by specialized technological standard for implementation layer.
3. *Implementation specifications*: One implementation for the SA is an agent based on JADE/JAVA platform. Compared with general agents, this endows SA with function operations in the form of web service and can support web integration. We discuss DEVS-based approaches to service-oriented model engineering in Section 2.6.

2.5.3.2 Cooperation Mechanism of SA

Cooperation mechanism of SA can be different according to different SA specifications and applications. A mechanism is given in Liu et al. (2014). In the

mechanism, a SA is designed to have four states including the working state, prepared state, waiting state, and searching state. In working state, service agent provides service and receives output from its previous SA. In prepared state, all behaviors are blocked for a new message to come. Waiting state is for the workers. In this state, the worker receives other organization members' ID information and input/output matching information from the organizer. Searching state is for the organizer. The organizer adopts "first come first serve" strategy to choose workers.

A composition algorithm is given based on the above cooperation mechanism, and a software platform prototype for an abstracted SoS simulation problem is also developed (Liu et al. 2014). This prototype used JADE4.1 (Java Agent Development Framework) platform to perform model composition with the SA-based method. JADE is a MAS (multi-agent system) software development platform in JAVA language. Web services are published in a Tomcat7.0 container and are packaged by WSIG (JADE Web Service Integration Gateway) plug-in. The purpose of WSIG is to achieve the integration of MAS and WS (web service) architecture.

2.6 DEVS-Based Service-Oriented Model Engineering and Simulation Environment

Mittal (2014) describes model engineering for cyber complex adaptive systems, a very challenging class of SoS, by extending Model-Based Systems Engineering (MBSE) paradigms (Zeigler 1976; Zeigler et al. 2000; Mittal and Martin 2013). Applied to complex adaptive systems, model engineering must address distinct challenges posed in the M&S domain such as model composability and executability. These problems can be overcome with formalisms that distinguish models (which represent the essence of a SoS) from simulators (which are the platforms for executing the models to generate their behavior). To do so, we can employ the theoretical and conceptual frameworks such as the systems-based DEVS concepts presented earlier. The DEVS formalism provides a sound and practical foundation for the architecture of model engineering and simulation environment presented in Fig. 2.7. Some of the main reasons for basing the architecture on DEVS are the following:

- DEVS formalizes what a model is, what it must contain, and what it doesn't contain (e.g., experimentation and simulation control parameters are not contained in the model).
- DEVS represents a system of interest (SoI) using well-defined input and output interfaces. This is critical because composing models requires respecting such boundaries for the constituent referent SoIs.
- DEVS is *universal* and *unique* for discrete event system models in the sense that any system that accepts events as inputs over time and generates events as outputs over time is equivalent to a DEVS in a strong sense: i.e., its behavior and structure can be described by such a DEVS.

- A DEVS model is a system-theoretic concept specifying inputs, states, outputs, similar to a state machine. Critically different, however, is that it includes a time-advance function that enables it to represent discrete event systems, as well as hybrid systems with continuous components in a straightforward platform-neutral manner.
- DEVS-compliant simulators execute DEVS-compliant models correctly and efficiently. DEVS defines what's necessary to compose modularized models that will be executable by a compliant simulator.
- DEVS models can be executed on multiple different simulators, including those on desktops (for development) and those on high-performance platforms, such as multi-core processors.
- DEVS supports model continuity which allows simulation models to be executed in real time as software by replacing the underlying simulator engine.

Mittal (2014) stresses the fundamental difference between *software-based* discrete event simulation and *systems-based* discrete event simulation. While the former is strictly based on object-oriented software engineering paradigm (e.g., Schmidt 2006; Volter et al. 2006), the latter enforces Wymore's System Theory on the object-oriented discrete event simulation engine as shown in Section 2. Since cyber complex adaptive systems are multi-agent adaptive systems at the fundamental level, there are many agent-based modeling (ABM) tools available to represent them. Unfortunately, due to their software-based object orientation, the large majority of these tools do not conform to Wymore systems theory's closure under composition principle. In contrast, a DEVS-based agent has the notion of a system attached to it and is built on formal semantics that adheres to Wymore's systems theory. Such an approach makes it possible to develop a simulator, a simulation protocol, and a distributed high-performance engine for agent/system model's execution that ensures that closure of coupling is not violated. Moreover, DEVS formal specification allows it to interface with model-checking tools based on unified modeling language (UML) tools to supplement simulation with formal verification and validation, a critical feature of model engineering (Zeigler and Nutaro 2014).

Several M&S environments exist that support the DEVS-based methodology just described, including DEVS-Suite, CD++, DEVSim++, JAMES II, Python DEVS, and VLE (see the list at DEVS Standardization Group (2014) for descriptions). Mittal and Martin (2013) describe packaging all these functionalities in a netcentric DEVS Virtual Machine (VM) that provides an agent-execution environment to apply to cyber complex adaptive systems. The M&S environment MS4Me was developed as the first in a commercial line of DEVS products (ms4systems.com). It employs Xtext, an EBNF grammar, within the Eclipse Modeling Framework on the Rich Client Platform and the Graphical Modeling Project to provide a full-blown IDE specifically tailored to a DEVS development environment (Zeigler and Sarjoughian 2012).

2.6.1 *System Entity Structure (SES)*

The System Entity Structure (SES) formalism is an ontology representation of compositions, components, and coupling patterns that can be pruned to select a particular hierarchical model tree information structure. Automatic synthesis can then generate an executable simulation model through selection of model components in the model base (Zeigler 1984; Kim et al. 1990; Kim and Zeigler 1989). The SES is implementation agnostic and can be represented in various knowledge representation frameworks including standard relational data formalisms (Park et al. 1997; Kim and Kim 2006; Zeigler et al. 2013).

The SES/MB framework has been studied in various computational environments and applied to numerous industrial problems (Mittal et al. 2006; Cheon et al. 2008). Recently, the SES/MB framework has seen increasing application to M&S of system of systems (SoS). Commercial environments have been developed to enable more flexible representation of alternatives (including composition patterns and hierarchical components) and rule-based constraints for pruning enabling the development of suites of families of models (Zeigler and Sarjoughian 2012). Distributed simulations of complex federations in HLA can now be generated in addition to the original stand-alone simulations (Kim et al. 2013; Seo and Zeigler 2009).

The methodology has led to the Hierarchical Encapsulation and Abstraction Principle which allows combining the top-down paradigm for the constructive simulation with the bottom-up paradigm for the emergent simulation. Applications have been to simulation study of agent-based tactical and operational effectiveness of warfare and network vulnerability analysis (Chi et al. 2009; You et al. 2013; You and Chi 2009).

Table 2.1 summarizes the above review by examining the layers of service-oriented model engineering and simulation environment presented in Fig. 2.8 from the point of view of DEVS support.

DEVS enables new frameworks for application domains, especially those that feature continuous/and discrete systems that interact (sometimes called hybrid systems). Some examples such as production flows in the food industry, building energy design, quantum key distribution (QKD) systems, and agent-based transportation evacuation are presented in Table 2.2 in terms of their novel features and unique capability offered when compared to existing approaches. DEVS also supports tools for simulating such models. For example, a compiler that employs DEVS to execute models expressed in the well-known simulation language, Modelica.

Table 2.1 Layers of service-oriented model engineering and simulation environment

Layer	Description	DEVS support
Model component layer	Classifies and organizes models	Implementation agnostic, flexible representation of alternatives (including composition patterns and hierarchical components), and rule-based constraints for pruning enabling the development of suites of families of models
Model service layer	Provides model standardization with interface specifications, unified service encapsulation, and transformation	The DEVS formalism provides a formal basis for semantic and pragmatic interoperability among DEVS models using Service-Oriented Architecture and DEVS Namespace
Model management layer	Executes searching, matching, composition, configuration, modification, and VV&A	Pruning of the SES enables automatic synthesis of an executable simulation model through extraction and coupling of model components in the model center
Simulation layer	Provides simulation, visualization, man-machine interaction	The DEVS Abstract Simulator provides a standard distributed DEVS protocol for interoperation of DEVS simulators
Application layer	Enables model engineering applications to specific domains	Numerous applications have been done with DEVS-based M&S. Examples of the development of frameworks are given below

2.7 Conclusion

Inspired by Prof. Tuncer Ören’s broad perspective on the M&S enterprise, we probed the nature of model engineering as spanning the life cycle of a model in the context of systems of systems engineering, particularly implemented with service orientation. We presented an architecture for service-oriented model engineering and simulation environments whose layers support the various activities of model engineering. Based on the results of DEVS research, we gave a more concrete characterization of such an environment and how model engineering and DEVS enable new frameworks for application areas. Unifying the various activities needed to produce credible models via the concept model engineering is only in its infancy.

Further research is needed to organize and deepen the body and knowledge and to probe each of the architectural layers we have identified, establish their cross connections, and add new ones as needed. The application context of service orientation is a good one to focus attention on the implementation of support for model engineering but not necessarily the only context in which support might be conceived. Similarly, DEVS theory and research have given much solid substance to the body of knowledge and practice of model engineering but more general

Table 2.2 DEVS-enabled frameworks

Application area	Novel feature	Unique capability
Components: processing units, conveyor belts	New framework for carrying out simulations of continuous-time stochastic processes	Keeping track of parameters related to the process and the flowing material (temperature, concentration of pollutant) is also considered. Since these parameters can change over time in a continuous manner, the possibility to transmit those laws as functions is introduced in the model
Development of DEVS models for building energy design	Allow different professions involved in the building design process to work independently to create an integrated model	Results indicate that the DEVS formalism is a promising way to improve poor interoperability between models of different domains involved in building performance simulations
Components: occupants, thermal network points, windows, HVACs, etc.		
Quantum key distribution (QKD) system with its components using DEVS	DEVS assures that the developed component models are composable and exhibit temporal behavior independent of the simulation environment	Enable users to assemble and simulate any collection of compatible components to represent complete QKD system architectures
Components: classical pulse generator, polarization modulator, electronically variable optical attenuator, etc.		
DEVS framework for transportation evacuation integrating event scheduling into an agent-based method	This framework has a unique hybrid simulation space that includes a flexible-structured network and eliminates time-step scheduling used in classic agent-based models	Hybrid space overcomes the cellular space limitation and provides flexibilities in simulating evacuation scenarios
Components: vehicles, agents		Model is significantly more efficient than popular multi-agent simulators. Keeps high model fidelity and the same agent cognitive capability, collision avoidance, and low agent-to-agent communication cost

theory of M&S should inform and unify such a body. For example, ultimately the elements, such as experimental frame, simulator, etc., and relations (modeling, simulation, applicability, etc.) of the theory of M&S must be brought in to fully consider the best practices for M&S and offer normative views on how to formulate the knowledge needed to make them better.

References

- Abdouni Khayari RE, Musovic A, Lehmann A et al (2010) A model validation Study of Hierarchical and Distributed Web Caching Model. In: Proceedings of the 2010 spring simulation multi-conference, Orlando, 11–15 Apr, p 98
- Balci O (2012) A life cycle for modeling and simulation. *Simulation* 88(7):870–883
- Boardman J, Sauser B (2006) System of systems – the meaning of “of”. IEEE/SMC international conference on system of systems engineering. IEEE, Los Angeles
- Cheon S, Kim DH, Zeigler BP (2008) System entity structure for XML meta data modeling: application to the US climate normals. 17th international conference on software engineering and data engineering, Los Angeles
- Chi SD, You YJ, Jung CH, Lee HS, Kim JI (2009) FAMES: fully agent-based modeling & emergent simulation SpringSim 2009. Proceedings of the 2009 spring simulation multiconference
- DEVS Standardization Group (2014) <http://cell-devs.sce.carleton.ca/devsgroup/?q=node/8>
- Fishwick PA (1989) Qualitative methodology in simulation model engineering. *Simulation* 52(3):95–101
- Fishwick PA (1990) Toward an integrated approach to simulation model engineering. *Int J Gen Syst* 17(1):1–19
- Kim JK, Kim TG (2006) A plan-generation-evaluation framework for design space exploration of digital systems design. *IEICE Trans Fund Electron Comm Comput Sci* E89-A(3):772–781
- Kim TG, Zeigler BP (1989) A knowledge-based environment for investigating multicompuser architectures. *J Inform Soft Technol* 31(10):512–520
- Kim BS, Choi CB, Kim TG (2013) Multifaceted modeling and simulation framework for system of systems using HLA/RTI CNS 2013. Proceedings of the 16th communications & networking symposium, society for computer simulation international, San Diego
- Kim TG, Lee C, Christensen CER, Zeigler BP (1990) System entity structuring and model base management. *IEEE Trans Syst Man Cybern* 20(5):1013–1024
- Liu X, Tang Y, Zheng L (2008) Survey of complex system and complex system simulation. *J Syst Simul* 20(23):6303–6312 (in Chinese)
- Liu J, Zhang L, Tao F (2014) Service-oriented model composition. In: Proceedings of SCS 2014 summer simulation multi-conference (SummerSim’14), Monterey, 6–10 July
- Mittal S (2014) Model engineering for cyber complex adaptive systems. EMSS, Bordeaux
- Mittal S, Martin JLR (2013) Netcentric system of systems engineering with DEVS unified process. CRC Press, Boca Raton, 712 p
- Mittal S, Mak E, Ntutro JJ (2006) DEVS-based dynamic model reconfiguration and simulation control in the enhanced DoDAF design process. *J Defense Model Simul* 3(4):239–267
- Ören TI (1971) GEST: a combined digital simulation language for large-scale systems. In: Proceedings of the Tokyo 1971 AICA (Association Internationale pour le Calcul Analogique) symposium on simulation of complex systems, Tokyo, 3–7 Sept, pp B-1/1–B-1/4
- Ören TI (1973) Application of system theoretic concepts to the simulation of large scale adaptive systems. In: Proceedings of the sixth Hawaii international conference on system sciences, Honolulu, 9–11 Jan
- Ören TI (1990) Simulation methodology: top down approach. In: Sage AP (ed) Concise encyclopedia of information processing in systems and organizations. Pergamon Press, Oxford, pp 421–425
- Ören TI (2005) Toward the body of knowledge of modeling and simulation (M&SBoK). In: Proceeding of I/ITSEC (Interservice/Industry Training, Simulation Conference), Orlando, 28 Nov–1 Dec, paper 2025, pp 1–19

- Ören TI (2014) Publications, presentations and other activities of Dr. Tuncer Ören on modeling and simulation: normative views for advancement and advanced methodologies <http://www.site.uottawa.ca/~oren/pubsList/MS-advanced.pdf>
- Ören TI, Yilmaz L (2012) Synergies of simulation, agents, and systems engineering. *Expert Syst Appl* 39(1):81–88
- Ören TI, Zeigler BP (1979) Concepts for advanced simulation methodologies. *Simulation* 32(3):69–82
- Ören TI, Zeigler BP (2012) System theoretic foundations of modeling and simulation: a historic perspective and the legacy of A Wayne Wymore. *Simulation* 88(9):1033–1046
- Ören TI, Zeigler BP, Elzas MS (eds) (1982) *Simulation and model-based methodologies: an integrative view*. Series: NATO ASI Subseries F, vol 10. NATO Advanced Institute Ottawa, Ontario, 26 July–6 Aug
- Pace DK (2004) Modeling and simulation verification and validation challenges. *Johns Hopkins APL Tech Dig* 25(2):2004
- Park HC, Lee WB, Kim TG (1997) RASES: a database supported framework for structured model base management. *Simul Pract Theory* 5(4):289–313
- Radeski A, Parr S (2002) Towards a simulation component model for HLA. In: *Proceedings of the 2002 fall simulation interoperability workshop (SISO Fall 2002)*. Paper ID 02F-SIW-079, Nov
- Schmidt DC (2006) Model-driven engineering. *IEEE Comput*, Orlando, FL, 39(2):25–31. doi:10.1109/MC.2006.58
- Seo C, Zeigler BP (2009) Interoperability between DEVS simulators using service oriented architecture and DEVS namespace. In: *A joint symposium DEVS integrative M&S (DEVS) and high performance computing (HPC), Proceedings of the Spring Simulation Conference, San Diego, CA*.
- Si N, Zhang L, Tao F, Guo H (2009) Research on multi-agent system based service composition methodology in semantic SOA (in Chinese). In: *Proceedings of the 5th conference on multi-agent system and control, Chongqing*, 19–20 Sept
- Volter M, Stahl T, Bettin J, Haase A, Helsen S (2006) *Model-driven software development: technology, engineering, management*. John, Chichester/Hoboken
- Waite W, Oren TI (2009) Modeling & simulation body-of-knowledge index(M&S BOKIndex). <http://sim-summit.org/WebinarBrief/BOK%20Webinar%20Brief%20Feb%202009%20v5%20Waite.pdf>
- Wymore AW (1967) *A mathematical theory of systems engineering: the elements*. John, New York
- Yilmaz L, Ören TI (2009) *Agent-directed simulation and systems engineering*, 1st edn. Wiley Series in Systems Engineering and Management, –VCH, Weinheim
- You YJ, Chi SD (2009) SIMVA: simulation-based network vulnerability analysis system SpringSim '09. *Proceedings of the 2009 spring simulation multicongference*
- You YJ, Chi SD, Kim JI (2013) HEAP-based defense modeling and simulation methodology. *IEICE Trans* 96-D(3):655–662
- Zeigler BP (1976) *Theory of modeling and simulation*, 1st edn. Wiley, New York
- Zeigler BP (1984) *Multifaceted modelling and discrete event simulation*. Academic, London
- Zeigler BP, Nutaro JJ (2014) Combining DEVS and model-checking: using systems morphisms for integrating simulation and analysis in model engineering. EMSS, Bordeaux
- Zeigler BP, Sarjoughian HS (2012) *Guide to modeling and simulation of systems of systems*. Springer, Dordrecht, p 393
- Zeigler BP, Kim TG, Praehofer H (2000) *Theory of modeling and simulation: integrating discrete event and continuous complex dynamic systems*, 2nd Revised edn. Academic Press Inc., Waltham, pp.11
- Zeigler BP, Seo C, Kim D (2013) System entity structures for suites of simulation models. *Int J Model Simul Sci Comput* 4(3). doi:10.1142/S1793962313400060
- Zhang L (2011) Model engineering for complex system simulation, The 58th CAST forum on new viewpoints and new doctrines, Li, 14–16 Oct

- Zhang L, Shen YW, Zhang XS, Song X, Tao F, Liu Y (2014a) The model engineering for complex system simulation. In: The 26th European modeling & simulation symposium (Simulation in Industry), Bordeaux, 10–12 Sept
- Zhang L, Li F, Song X, Liu YK (2014b) A supporting environment of model engineering for complex systems. In: The 11th Chinese intelligent system conference (CISC'14), Beijing, 18–19 Oct

Concepts and Methodologies for Modeling and
Simulation

A Tribute to Tuncer Ören

Yilmaz, L. (Ed.)

2015, XV, 352 p. 116 illus., Hardcover

ISBN: 978-3-319-15095-6