A SYSTEMS THEORETICAL PERSPECTIVE ON DATA-DRIVEN MODELING AND SIMULATION

Yilin Huang[(a)], Mamadou D. Seck[(b)], Alexander Verbraeck[(a)]

[(a)] Systems Engineering and Simulation, Delft University of Technology
[(b)] Engineering Management and Systems Engineering, Old Dominion University

[(a)] y.huang@tudelft.nl, a.verbraeck@tudelft.nl [(b)] mseck@odu.edu

ABSTRACT

This paper discusses Data-Driven Modeling and Simulation (DDM&S) from a systems theoretical viewpoint. The focus is on understanding Systems Theory in relation with Modeling and Simulation (M&S) and how data may convey information about a system. Such an understanding could help formalizing the transformation process from data to systems knowledge and in turn offers a possibility to automate the process. In DDM&S, a simulation environment should have the ability to correctly interpret the data provided, extract useful information and automatically incorporate it into the simulation model so that changes in data may change the model. This flexibility could be achieved by representing the system not as a monolithic whole but as a composition of interrelated parts, and by the support of intelligent data analysis and model transformation algorithms. An example of automatic model generation is given to illustrate the concept.

Keywords: data-driven, modeling and simulation, systems theory, model transformation

1. TOWARDS DATA-DRIVEN MODELING AND SIMULATION

The role of data in Modeling and Simulation (M&S) is becoming increasingly important along with the advances in data collection and storage technologies. Despite the discussions about input/output data analysis in literature, little is said about more comprehensive use of data in the M&S process. Data that can be useful for M&S is not limited to measurements or observations from real systems but can also comprise data that is generated from computer-aided tools such as Computer-Aided Design, Engineering and Geographic Information Systems. Data may be used for different purposes such as model construction, configuration and calibration. Fully utilize data in M&S could not only benefit the M&S process but also better integrate M&S with its applications such as design, engineering and training.

Data-Driven is not a new term in M&S but its earlier meaning differs from how it is understood now. Data-driven simulators were once those that allowed users to build models without programming knowledge (Ekere & Hannam, 1989; O'Keefe & Haddock, 1991; Pidd, 1992). Those programs accepted front-end entry data such as flow diagram or GUI input, which have been used since the 1950s. This is not how we conceive of Data-Driven Modeling and Simulation (DDM&S) nowadays. DDM&S discussed in this paper refers to M&S processes where data is used to automatically define, specify and/or modify model properties and logics. The simulation software is designed to correctly interpret the data provided, extract useful information and automatically incorporate it into the model so that changes in data can change the model correspondingly.

In DDM&S, the use of data may range from model construction, configuration and initialization, to model update, estimation and calibration. Simulation models should be developed to allow for such automated data uses. The primary concern here is that we need to create simulation software that provides enough degrees of freedom to represent the set of systems in our domain of interest. This flexibility can be achieved by representing the system of interest not as a monolithic whole but as a composition of interrelated sub-systems (or parts). In this context, we can see the definition of model parts as the "*fixed*" which represents the corresponding domain knowledge, whereas the selection of those parts, their parameterization, initialization, and composition are the "*unfixed*" whose degrees of freedom are constrained by the corresponding domain knowledge. The unfixed affords the flexibility. In such cases, data can be used to specify the unfixed, and changes in data may change the unfixed. The cost of this flexibility is complexity in simulation software. The design choice is then often a compromise between flexibility and complexity.

Although the DDM&S concept is straightforward, developing DDM&S software introduces complexity. The effort can be nonetheless beneficial for large-scale systems that need long-term use of simulation models. These models often take long to develop and modify, and they often need to be changed along with changes in the systems. Many examples can be found in supply chains, transportation and manufacturing among other domains (Kim, 2009; Qiao, Riddick, & McLean, 2003; Tannock, Cao, Farr, & Byrne, 2007; Wang, Chang, Xiao, Wang, & Li, 2011). Some systems may need simulation in (near) real-time so that forecasting can be made responding timely to crucial events. Examples can be found in fire forecasting, water management, critical infrastructure and emergency response systems where sensor data are often used (Darema, 2011; Hu, 2011; Jahn, Rein, & Torero, 2012).

In literature, e.g., Banks, Carson, Nelson, & Nicol (2010); Law (2007), the steps in a simulation study and data analysis within are well discussed. Many discuss data from a statistical perspective. In this paper, as a complement to statistical views, we discuss data from a systems theoretical perspective. We believe this could help formalize transformation processes from data to systems knowledge and could in turn offer a possibility to automate the processes.

We first present the theoretical foundation, namely how knowledge is understood in Information Science (IS) and Systems Theory and how model composition is relevant in this context. Then we discuss DDM&S from a systems theoretical perspective where transformations of data-information-knowledge can be performed at different systems epistemological levels. An example of DDM&S used in component-based automatic model generation in light-rail domain is given to illustrate the concept.

2. THEORETICAL FOUNDATIONS

DDM&S brings together the two fields of IS and M&S. To root its concept on solid theoretical grounds, we revisit two hierarchies that concern knowledge: (1) the *Knowledge Hierarchy* widely recognized by the IS community to contextualize data, information and knowledge (Rowley, 2007), and (2) the *Epistemological Hierarchy of Systems* which explains knowledge from a systems theoretical viewpoint (Klir & Elias, 2003).

2.1. Data-Information-Knowledge Hierarchy

In the knowledge hierarchy, data, information and knowledge are arranged from the low to high levels in a pyramid. They are three fundamental and interrelated concepts still with debates about their nature, definition and relation. We present the basic ideas based on (Checkland & Holwell, 1998; Rowley, 2007; Ulrich, 2001; Zins, 2007).

Data comprises sets of symbols recorded in or on a medium. As such, data is unstructured, unprocessed and invariant. Checkland & Holwell (1998) also introduced the term *capta* in between data and information to refer to the subset of data that we select for attention.

Information is data (or capta) that are enriched with contextual "meaning attribution". It is done in a context that may well be shared by some but may also be unique to an individual. What machines cannot do, in a strict sense, is to generate unequivocal information; what they can do is merely to process capta into useful forms that can imply (or match) certain prescribed categories of information (Checkland & Holwell, 1998). Information can be inferred from data. This process is not viable without semiotic clarifications, namely syntactic clarity (comprehensibility), semantic clarity (meaning) and pragmatic clarity (relevance).

Knowledge is propositional. It is a "justified belief substantiated by compelling reasons" (Ulrich, 2001). Knowledge is structured and organized information expected to have greater longevity than a collection of many items of information that are only ephemerally meaningful and relevant. It is the general understanding and awareness garnered from accumulated information tempered by experience, enabling new contexts to be envisaged. Knowledge exists in the human mind but can be given physical representations.

With data we yet know nothing, with information we know what (and/or who, when, where), and with knowledge we know how (Ackoff, 1989; Zeleny, 1987). Understanding supports the transformation process from a lower level to the next (Rowley, 2007). If we consider a simulation model as a representation of the relevant knowledge of a system, then the transformation from data to knowledge for a given context is what one wants to achieve in DDM&S. For the transformation, we have to understand what data is available and what type of systems knowledge it can represent.

2.2. Epistemological Hierarchy of Systems

In M&S, a system being modeled can be called *referent*. Understanding what we know about the referent is essential in modeling. Simon (1962) roughly defined a complex system as one made up of a large number of parts that interact in a non-simple way. Generally, a system can be viewed as a pair of two orthogonal sets that are extremely rich in content: (1) a set of *things* (thinghood) and (2) a set of *relations* among things (systemhood) (Klir, 2001; Klir & Elias, 2003). Modeling (in computer simulation) is the activity to capture the thinghood and systemhood of interest in a computational form and study their evolvement for the purpose of understanding and/or prediction. Based on different levels of knowledge about the systemhood, Klir & Elias (2003) established an epistemological hierarchy of systems, as shown in Figure 1. Each level in the hierarchy implies the profoundness of our system knowledge.

Level 3 Structure System	*Relations between models at Level 2*
Level 2 Generative System	*Models that generate data at Level 1*
Level 1 Data System	*Observations or desirable states of Level 0*
Level 0 Source System	*A source of empirical data*

Figure 1: Hierarchy of epistemological levels of systems (based on Klir & Elias, 2003)

The bottom level of the hierarchy (Level 0) is the "primitive understanding" of a system, known as a *source system*. At this level, we know which variables and states we are interested in. The source system is (at least potentially) a source of empirical data. At the next level is the *data system* (Level 1) where data of the variables (or states) are obtained from observation or measurement, or are defined as desirable states. This is often the level where a system is accessible from the outside. Above the data system is the *generative system* (Level 2) where we acquire the knowledge to define the translation rules that are able to generate the observed

or hypothetical (unobservable and/or internal) data of the system. At Level 3, the *structure system* describes systems knowledge as a set of generative systems that interact with each other in some way. At this level we understand the system as an interactive whole that has generative systems as parts. (Thus generative systems are often referred to as sub-systems.) This level of knowledge is also known as the knowledge of systems or structured sub-systems.

A higher epistemological level entails that the knowledge attained at the lower levels is known and it contains additional knowledge that is not available at the lower levels. The process of climbing the epistemological hierarchy transforms the notion of systems knowledge from primitive to more meaningful in order to specify a generative or a structure system that can reproduce the data at Level 1. Problems of this type fall into the category of *Systems Modeling*, and this category can be further divided into *Systems Inference* and *Systems Design* depending on whether the system is in existence (Klir, 1988; Zeigler, Praehofer, & Kim, 2000). *Systems analysis*, on the other hand, is the process of using the generative or structure system to produce data; computer simulation is an example of this type (Zeigler, Praehofer, & Kim, 2000). In DDM&S, we have data about a system that we want to model at the generative and structure levels. This requires knowledge about system decomposition and model composition that we discuss in the following section.

2.3. System Decomposition and Model Composition

Approaching systems complexity, we habitually divide a system into less complex parts, and analyze the parts and their interrelations that are more comprehensible. This approach is well grounded in systems theory and systems thinking literature (Checkland, 1999; Klir & Elias, 2003; Simon, 1996). Systems decomposition also has another advantage. What one knows about a complex system is often partial, i.e., a subset of the parts and interrelations in a system may be known to some experts. Adequate decomposition of a system can provide a systemic structure to aggregate and merge the knowledge about different parts and interrelations. The decomposition can be recursive until an elementary level of sub-systems is reached (Simon, 1996).

The word "system" in Greek means composition in its literary sense. Model components are basic parts (or building blocks) in model construction. If components and their interactions are designed to represent some distinguishable objects and relations in a real system, modelers and domain experts may easily identify and use the components to construct larger models and to validate them. The domain knowledge can be integrated into the model by terminology and definition of the components (Valentin & Verbraeck, 2002). Using components, changes of the model can be limited to some components instead of the whole model. Domain ontology and mapping between the ontology and simulation components are useful in this context.

In M&S, composability is defined as the capability to select and assemble simulation components in various combinations into valid simulation systems to satisfy specific user requirements (Petty & Weisel, 2003). Although component-based modeling is widely promoted and encouraged for its considerable benefits, it is shown difficult to apply (Szabo & Teo, 2007; Tolk, Diallo, King, Turnitsa, & Padilla, 2010; Yilmaz, 2004). Model composability shows many characteristics that are similar to those of software composability and systems design (Baldwin & Clark, 2000; Braude & Bernstein, 2010; Hofmann, 2004; Pidd & Robinson, 2007; Sommerville, 1996) whose design principles can be learnt from. Designing composable models should start at the conceptualization phase.

To design composable and reusable simulation models, the design goal needs to be at least one level higher than the design of any specific application of the model components. How the components may operate with other components and how other designers can make use of the components are the higher-level goals, which make standardization in model component design particularly important. This is not only meant for interface definition (as one does, e.g., for composable software) but also for model definition at a conceptual level (Tolk, Diallo, King, Turnitsa, & Padilla, 2010; Tolk & Muguira, 2003; Yilmaz, 2004). Systems decomposition determines the model composition and interaction. Designers should aim at conceptually decomposing a system into parts and relations such that they capture the essence in a system to serve the intended simulation goal and at the same time provide the flexibility for other composite combinations to represent a set of systems in the domain of interest. In general, domain knowledge can be separated in two classes, the fixed and the unfixed (as mentioned in Section 1). The fixed can be defined by model components at the generative level. The unfixed can be handled by the selection, configuration and composition of the components. In DDM&S, data is used to infer the unfixed. Systems decomposition is thus critical in the sense that it determines possible model structures and configurations in model composition.

3. A SYSTEM THEORETICAL APPROACH OF DDM&S

For a given M&S purpose, a modeler's understanding of the referent follows an epistemological hierarchy (Section 2.2). The transformation of data-information-knowledge (Section 2.1) in DDM&S is supported by modelers' systems understanding associated to the epistemological levels.

3.1. Data and the Epistemological Hierarchy

Data at Level 1 (L1) is the *empirical data* obtained by observations and/or measurements performed on the referent. It is sets of samples on some observables including input, output and state variables. Having the data as a starting point, the effort of modeling is to specify a generative or a structure system (L2/L3) that

can produce some data that agree with the empirical data. Can the L2/L3 knowledge be obtained solely through transformation of L1 data? This can be achieved, e.g., by using a regression model. But if we can obtain domain knowledge about the dynamics and the structure of the referent (L2/L3), then the L2 and L3 knowledge of the model is not necessarily derived directly from the L1 data. In practice, the elicitation of L2/L3 knowledge in M&S relies heavily on (human) domain expertise. The L2/L3 knowledge of the referent often can be obtained from other systems.

Take a public transportation system (say, System T) as an example (the referent). Its L3 knowledge is, e.g., the design of the infrastructure network, and the design is the outcome of some design process, which can be conceived of as another independent system (say, System D). The design process produces design documents (L1 of System D) according to which the infrastructure network (L3 of System T) is built. These documents, therefore, correspond to some knowledge at L3 of System T. When modeling System T, we may use the design documents to determine some L3 knowledge about the model of System T. In other words, we need to find a function that transforms L1 of System D into some L3 of System T.

The knowledge of a referent at a certain level may be generated from more than one system, and similarity, one system may generate knowledge about the referent at different epistemological levels. The knowledge hierarchy (and transformation) in IS can be applied to each of the epistemological levels. A certain type of data after correct analysis and interpretation may deliver certain information of the referent. In DDM&S, designers should explore different possibilities of using various types of data and try to transform them into information and eventually into knowledge at different epistemological levels. The transformation is meant in a computational sense where machines (and software) can only interpret data in a predefined way. When carefully designed, they can efficiently process data into certain structure that can match some prescribed categories of information and knowledge. As modelers we also need to carefully design these prescribed categories in order to satisfy the flexibility needs of the simulation software.

3.2. Using Data in Modeling

Data is used in every simulation study. They can be gathered, e.g., from systems design (e.g., CAD data), planning and operation (e.g., scheduling and ERP data). Simulated data from other models may also be a data source if the results are validated. Some data sources typically contain information that can be mapped into different levels of model knowledge. Making these relations explicit to the modelers help them design the simulation software.

Modeling concerns two worlds, the world of the "real" where our interests are situated, and the world of the "virtual" where the models of the "real" are built. The tough questions in modeling are about constructing viable passages from the real to the virtual. In a strict sense, what is obtainable from a "real system" is only data. When one observes a system of interest (as a referent), one always chooses a perspective and tries to get or give some rational explanations of the system. A modeler tries to express these explanations (many of which are domain knowledge) by means of a model. When the modeler uses data for this purpose, he or she often needs the domain expertise to know how to correctly interpret the data. When manual analysis and interpretation of the data become too cumbersome and time-consuming, we may consider formalizing the existing know-how and encoding this into automated processes.

4. DATA-DRIVEN MODEL GENERATION: AN EXAMPLE

In this section, we discuss a data-driven approach for automatic simulation model generation where prebuilt and validated model components are used as building blocks. We view this approach as an automated reuse of model components. Automation is the execution by a machine agent (usually a computer) of a function that was previously carried out by a human (Parasuraman & Riley, 1997). When we, as human modelers, want to construct a model from model components, we need to know what components to use, and how to configure and structure them together. For an automated process, the same types of information are required which are derived from the provided data.

4.1. Case Description

The modeling case is in the domain of urban public light-rail transportation. It is a long-term project in cooperation with a public transportation company in the Netherlands. At the beginning of the project, the scales of the models developed were relatively small, e.g., modeling an intersection or a specific area in a city to assess the control and operation strategies (Kanacilo & Verbraeck, 2006; Kanacilo & Verbraeck, 2006; Kanacilo & Verbraeck, 2007). The models (i.e., the components and structures) were defined by humans in XML files that were then converted into models. After a number of simulation studies, the organization decided to use the simulation models more extensively. Larger models were needed, e.g., a complete light-rail service line or the network of a whole city. This was when problems arose because manual definition of the model became unmanageable. In a later study (Huang, Seck, & Verbraeck, 2010; Huang, Verbraeck, Oort, & Veldhoen, 2010), the XML definition contained thousands of nodes and dozens of levels and attributes per node on average. The problem lies not only in the amount of effort and time but also in the fact that the manual procedure turned out to be increasingly error-prone which caused difficulties in the debugging process.

Automation seemed to be a solution but it was unclear by then how it could be done. Our approach was threefold. First, the existing models (and model components) were investigated with experiences in how

they were constructed. A computational procedure with the data structure and algorithms were designed and the necessary information requirement for model generation was specified. Second, the available data sources that could deliver the required information were identified, and the plan of how to obtain the missing data was made. Third, the model components were adapted and completed so that they can be easily used for automated composition and configuration. These three major tasks were carried out in parallel and the outcome of one influenced another.

4.2. The Model Component Library

The model component library has been gradually developed for the light-rail transportation simulation project. Some components have been adapted and added for the purpose of model generation. The library is called LIBROS (Library for Rail Operations Simulation). It follows the DEVS formalism (Zeigler, Praehofer, & Kim, 2000) for model component specifications. Railway operational elements such as vehicles, tracks, and sensors are specified as atomic models, each of which represents one functional aspect of the rail infrastructure or as required by the simulation model. They can be used to create more complex rail components such as stations and block sections, which in turn can be further composed until a complete representation of the modeled system is formed. The DEVS simulator underlying LIBROS is ESDEVS (Seck & Verbraeck, 2009). It implements the parallel DEVS and dynamic structure DEVS (Barros, 1995) on top of DSOL (Distributed Simulation Object Library) (Jacobs, 2005; Jacobs, Lang, & Verbraeck, 2002) which is a general-purpose event-scheduling based simulator. To enable model identification, configuration and coupling, different types of coupled models are defined in forms of meta-models in the library.

4.3. Model Generator: The Concept

Some model generators used formal model definitions for model generation (Balci, Nance, Derrick, Page, & Bishop, 1990; Foeken & Voskuijl, 2010; Kang, 1997; Son, Jones, & Wysk, 2000; Son, Wysk, & Jones, 2003). We prefer the data-driven approach for the reasons stated in Section 4.1. More recent works are inclined towards this approach (Bergmann & Strassburger, 2010; Jeong & Allan, 2004; Lucko, Benjamin, Swaminathan, & Madden, 2010; Shephard, Beall, O'Bara, & Webster, 2004; Tannock, Cao, Farr, & Byrne, 2007; Wang, Chang, Xiao, Wang, & Li, 2011). Some of those works discussed the concept or built a prototype. Some used data (or data models) that contained logical relations represented the model structure, or the model structure is generated in a parameterized way. In our work, the model structure is generated from data that do not directly contain logical relations. The generator can create the model structure from the data sources. The algorithm constructs models from selecting, structuring and configuring model components. Model selection heuristics that represent the domain knowledge of which components are relevant to the modeling goal are used to guide the component identification and the composite (Lee & Zobel, 1996), i.e., they are used to define the data inferential rules.

In some cases, if the modeled system has a simple structure then the model may be directly generated from the data. However, in many cases, the system is complex and the data that describe the system do not contain the relational logic that can be directly applied to the desired model structure. In such cases, several steps are necessary in the model generation procedure and intermediate data structures are employed to incrementally construct a relational representation of the system structure that in turn can be transformed into the corresponding model composite structure.

As mentioned earlier, different data sources may provide systems knowledge at different levels. The data may come from design, operation scheduling, resource allocation, etc. Starting from these data sources, the data analysis results shall eventually reflect the structural and behavioral preconditions that are the basis of constructing the (initial) model structure and initializing the model state. The transformation from the data source to the model structure is often too complex to be accomplished in one go. It can be generally divided into three steps. Assume that after pre-processing the data can be correctly interpreted describing the system in a primitive format without logical relations, e.g., a list of numerical or textual descriptions.

At the first step, a relational graph is created from the data based on its descriptive content. The data inference may involve some common sense rules or some basic domain knowledge. In both cases, the information needed to create the relational graph is self-contained by the data sources, i.e., no extra supporting information is required. The relational graph may be created incrementally for the convenience of structuring. For example, we may first identify that entities a, b, c are related to form A, and entities x, y, z are related to form B; at a later round of structuring, A and B may be grouped together to form C, and so forth until the desired level of structure is reached. The relational graph represents the structure of the data content. At the second step, we can discover the systems structure it represents with the help of a domain ontology map. By searching the ontology space, a match in patterns of entity attributes or relations ascertains what that part of the data structure represents within the overall systems structure. At the third step, the systems structure is transformed into corresponding model structure according to a model counterpart table. A model counterpart table specifies a mapping relation between a systems entity and its counterpart in the model. The mapping relation maybe one to one or one to more, i.e., a systems entity may have more than one version of models. For instance, a rail vehicle can be modeled with one physical entity or several segmented physical entities. Depending on the available data detail and the desired model detail, one model counterpart can be chosen for model generation. Additionally, the

model counterpart table also specifies if any other model parts shall be added to a given systems entity. In railway simulation, e.g., different types of intersections may have different control rules. These rules are not a part of the data but defined as a configurable sub-component in the intersection model. For an identified type of intersection, the model counterpart table specifies which control unit shall be added. With the identified component counterparts and the added components, the entire model structure is fully fledged. The simulation model can then be constructed, configured and initialized accordingly using the available model components in the library.

Limited by the length of this paper, we cannot give detailed examples of the model generation method and algorithms. Readers of interest may refer to Huang, Seck, & Verbraeck (2011) for a short example, or to Huang (2013) for a more complete example.

4.4. Some Remarks

Component-based modeling or component-based engineering in general is founded on a paradigm common to many engineering disciplines: complex systems can be obtained by assembling components (Gössler & Sifakis, 2005). Recursively constructing more complex components from simpler ones is a useful concept because it tackles incomprehensible problems from tangible bases. The simplicity of the concept makes it powerful. Component-based model generation automates the model composition for a given modeling objective. It is useful or may be the only solution if the model scale drastically increases and using simpler models is not an alternative. Once developed, the software has appealing long-term benefits.

The completeness of data used for model generation is important because the model generator cannot deal with unanticipated incompleteness. Such problems need to be solved prior to model generation. A straightforward solution is to complete the missing data when possible. In the light-rail model generation case, some contents that were not indicated in the original data were added manually for model transformation. If it is impossible to complete missing data, implementing rules in the generator, the model counterpart table or in the model components can be alternatives. In the case discussed, some data was not available but according to domain experts, the information may be inferred from the data that was available. Therefore, additional rules were added into the model counterpart table to generate the part of information that was missing.

5. CONCLUSIONS

This paper presented a broad view of DDM&S that is not limited to the use of observed data from the system of interest but encompasses various categories of data such as design and engineering documents. This view is motivated by the fact that more and more data becomes available along with the advances in data collection and storage technologies. Model-based approach in systems development provides different data sources that may be useful for inferring information about systems structure and behavior. To use these data in M&S, a comprehensive approach with understanding the data in relation with Systems Theory is necessary in addition to statistical procedures.

To explain the concept, we reviewed the data-information-knowledge hierarchy in IS and the epistemological hierarchy in Systems Theory. Based on these theories, we showed how multiplicity of data sources can be associated to different epistemological levels of systems, and how data may be transformed to the related systems knowledge. Using data in M&S has heavily relied on human intervention. However, when the manual process is well understood (especially at a systems theoretical level), the relevant domain knowledge and modeling knowledge may be embodied into an automated process. These processes can be beneficial for many systems such as those that have long-term or real-time needs for simulation models. The automation may also better integrate M&S with other technologies or applications such as optimization, design and engineering. Because of the variation in systems knowledge and the diversity in simulation goals, we may decompose the system such that the "fixed" parts (parts that are unlikely to change) are represented by pre-developed model components and the "unfixed" parts to be represented by model configuration and composition. The decomposition of a system into parts and relations should capture the essence in a system to serve the intended simulation goal, and at the same time provide the flexibility for other composite combinations to represent a set of systems in our domain of interest. This flexibility is supported by data analysis and transformation algorithms that infer data with certain structures that can match some prescribed categories of systems knowledge. The example provided in the last section explained an automated process of model generation using different data sources. The research in DDM&S is rich in content. It often requires knowledge in many disciplines. This interdisciplinary nature predestines the application of systems engineering approaches in developing such simulation software.

REFERENCES

Ackoff, R. L. (1989). From data to wisdom. Journal of Applied Systems Analysis, 16, 3-9.

Balci, O., Nance, R. E., Derrick, E. J., Page, E. H., & Bishop, J. L. (1990). Model generation issues in a simulation support environment. In Proceedings of the 1990 Winter Simulation Conference (pp. 257-263).

Baldwin, C. Y. & Clark, K. B. (2000). The Power of Modularity. MIT Press.

Banks, J., Carson, II, J. S., Nelson, B. L., & Nicol, D. M. (2010). Discrete-Event System Simulation (5th). Pearson Education.

Barros, F. J. (1995). Dynamic Structure Discrete Event System Specification: A new Formalism for Dynamic Structure Modeling and Simulation. In Proceedings of The 1995 Winter Simulation Conference (pp. 781-785).

Bergmann, S. & Strassburger, S. (2010). Challenges for the Automatic Generation of Simulation Models for Production Systems. In Proceedings of the 2010 Summer Simulation Multiconference, Ottawa, Canada, 2010 (pp. 545-549).

Braude, E. J. & Bernstein, M. E. (2010). Software Engineering: Modern Approaches (2nd). John Wiley & Sons.

Checkland, P. (1999). Systems Thinking, Systems Practice: Includes a 30-Year Retrospective. John Wiley & Sons.

Checkland, P. & Holwell, S. (1998). Information, Systems and Information Systems - making sense of the field. John Wiley & Sons.

Darema, F. (2011). {DDDAS} computational model and environments. Journal of Algorithms and Computational Technology, 5(4), 545-560.

Ekere, N. N. & Hannam, R. G. (1989). Evaluation of approaches to modelling and simulating manufacturing systems. International Journal of Production Research, 27(4), 599-611.

Foeken, M. J. & Voskuijl, M. (2010). Knowledge-based simulation model generation for control law design applied to a quadrotor UAV. Mathematical and Computer Modelling of Dynamical Systems, 16(3), 241-256.

Gössler, G. & Sifakis, J. (2005). Composition for component-based modeling. Science of Computer Programming, 55(1-3), 161-183.

Hofmann, M. (2004). Criteria for decomposing systems into components in modeling and simulation: Lessons learned with military simulations. Simulation, 80(7-8), 357-365.

Hu, X. (2011). Dynamic Data Driven Simulation. SCS M&S Magazine, 5, 16-22.

Huang, Y. (2013). Automated Simulation Model Generation. Ph.D. dissertation, Delft Univeristy of Technology.

Huang, Y., Seck, M. D., & Verbraeck, A. (2010). The Architecture and Components of LIBROS: Strengths, Limitations, and Plans. In Proceedings of The 2010 European Simulation and Modelling Conference, Hasselt, Belgium, 2010 (pp. 80-87). Eurosis-ETI.

Huang, Y., Seck, M. D., & Verbraeck, A. (2011). From Data to Simulation Models: Component-based Model Generation with a Data-driven Approach. In Proceedings of the 2011 Winter Simulation Conference, Phoenix, AZ, 2011 (pp. 3724-3734).

Huang, Y., Verbraeck, A., van Oort, N., & Veldhoen, H. (2010). Rail Transit Network Design Supported by an Open Source Simulation Library: Towards Reliability Improvement. In Transportation Research Board 89th Annual Meeting Compendium of Papers, Washington, DC, USA, 2010. TRB.

Jacobs, P. H. M. (2005). The {DSOL} simulation suite - Enabling multi-formalism simulation in a distributed context. Ph.D. dissertation, Delft University of Technology.

Jacobs, P. H. M., Lang, N. A., & Verbraeck, A. (2002). D-SOL: A Distributed Java Based discrete event simulation architecture. In Proceedings of the 2002 Winter Simulation Conference (pp. 793-800). IEEE.

Jahn, W., Rein, G., & Torero, J. L. (2012). Forecasting fire dynamics using inverse computational fluid dynamics and tangent linearisation. Advances in Engineering Software, 47(1), 114-126.

Jeong, K.-Y. & Allan, D. (2004). Integrated system design, analysis and database-driven simulation model generation. In Proceedings of the IEEE Annual Simulation Symposium (pp. 80-85).

Kanacilo, E. M. & Verbraeck, A. (2006). Decision Support for Control Design of Rail Infrastructures. In SIMTECT 2006 - Simulation: Challenges and Opportunities for a Complex and Networked World.

Kanacilo, E. M. & Verbraeck, A. (2006). Simulation services to support the control design of rail infrastructures. In Proceedings of the 2006 Winter Simulation Conference (pp. 1372-1379). IEEE.

Kanacilo, E. M. & Verbraeck, A. (2007). Assessing tram schedules using a library of simulation components. In Proceedings of the 2007 Winter Simulation Conference (pp. 1878-1886). IEEE.

Kang, S. (1997). Knowledge based automatic simulation model generation system. IEE Proceedings: Circuits, Devices and Systems, 144(2), 88-96.

Kim, B.-I., Kim, Jeong, S., Shin, J., Koo, J., Chae, J., and Lee, S. (2009). A layout- and data-driven generic simulation model for semiconductor fabs. IEEE Transactions on Semiconductor Manufacturing, 22(2), 225-231.

Klir, G. J. (2001). Facets of Systems Science (2nd). Kluwer Academic/Plenum Publishers.

Klir, G. J. (1988). The Role of Uncertainty Principles in Inductive Systems Modelling. Kybernetes, 17(2), 24-34. Klir, G. J. & Elias, D. (2003). Architecture of Systems Problem Solving (2nd). Kluwer Academic/Plenum Publishers.

Law, A. M. (2007). Simulation Modeling and Analysis (4th). McGraw-Hill.

Lee, C. H. & Zobel, R. N. (1996). Representation of simulation model components for model generation and a model library. In Proceedings of the IEEE Annual Simulation Symposium (pp. 193-201).

Lucko, G., Benjamin, P. C., Swaminathan, K., & Madden, M. G. (2010). Comparison of manual and automated simulation generation approaches and their use for construction applications. In

Proceedings of the 2010 Winter Simulation Conference (pp. 3132-3144).
O'Keefe, R. M. & Haddock, J. (1991). Data-driven generic simulators for flexible manufacturing systems. International Journal of Production Research, 29(9), 1795-1810.
Parasuraman, R. & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. Human Factors, 39(2), 230-253. Petty, M. D. & Weisel, E. W. (2003). A composability lexico. In Proceedings of the Spring 2003 Simulation Interoperability Workshop (pp. 181-187).
Pidd, M. (1992). Guidelines for the design of data driven generic simulators for specific domains. Simulation, 59(4), 237-243.
Pidd, M. & Robinson, S. (2007). Organising insights into simulation practice. In Proceedings of the 2007 Winter Simulation Conference, Piscataway, NJ, USA, 2007 (pp. 771-775). IEEE Press.
Qiao, G., Riddick, F., & McLean, C. (2003). Data driven design and simulation system based on XML. In Proceedings of the 2003 Winter Simulation Conference (pp. 1143 - 1148 vol.2).
Rowley, J. E. (2007). The wisdom hierarchy: representations of the {DIKW} hierarchy. Journal of Information Science, 33(2), 163-180.
Seck, M. D. & Verbraeck, A. (2009). {DEVS} in {DSOL}: Adding {DEVS} operational semantics to a generic Event-Scheduling Simulation Environment. In Proceedings of the 2009 Summer Computer Simulation Conference.
Shephard, M. S., Beall, M. W., O'Bara, R. M., & Webster, B. E. (2004). Toward simulation-based design. Finite Elements in Analysis and Design, 40(12), 1575-1598.
Simon, H. A. (1962). The Architecture of Complexity. Proceedings of the American Philosophical Society, 106(6), 467-482.
Simon, H. A. (1996). The Sciences of the Artificial (3rd). MIT Press.
Sommerville, I. (1996). Software Engineering (5th). Addison-Wesley.
Son, Y. J., Jones, A. T., & Wysk, R. A. (2000). Automatic generation of simulation models from neutral libraries: An example. In Proceedings of the 2000 Winter Simulation Conference (pp. 1558-1567).
Son, Y. J., Wysk, R. A., & Jones, A. T. (2003). Simulation-based shop floor control: Formal model, model generation and control interface. IIE Transactions, 35(1), 29-48.
Szabo, C. & Teo, Y. M. (2007). On Syntactic Composability and Model Reuse. In Proceedings of the International Conference on Modeling and Simulation (pp. 230-237). IEEE Computer Society Press.
Tannock, J., Cao, B., Farr, R., & Byrne, M. (2007). Data-driven simulation of the supply-chain -- Insights from the aerospace sector. International Journal of Production Economics, 110, 70-84.
Tolk, A. & Muguira, J. A. (2003). The Levels of Conceptual Interoperability Model. In Proceedings of the 2003 Fall Simulation Interoperability Workshop. IEEE CS Press.
Ulrich, W. (2001). A Philosophical Staircase for Information Systems Definition, Design and Development: A Discursive Approach to Reflective Practice in ISD (Part 1). Journal of Information Technology Theory and Application (JITTA), 3(3), 55-84.
Valentin, E. & Verbraeck, A. (2002). Guidelines for designing simulation building blocks. In Proceedings of the 2002 Winter Simulation Conference, San Diego, CA., 2002 (pp. 563-571). IEEE.
Wang, J., Chang, Q., Xiao, G., Wang, N., & Li, S. (2011). Data driven production modeling and simulation of complex automobile general assembly plant. Computers in Industry, 62(7), 765-775.
Yilmaz, L. (2004). On the Need for Contextualized Introspective Models to Improve Reuse and Composability of Defense Simulations. The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology, 1(3), 141-151.
Zeigler, B. P., Praehofer, H., & Kim, T. G. (2000). Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems (2nd). Elsevier/Academic Press.
Zeleny, M. (1987). Management Support Systems: Towards Integrated Knowledge Management. Human Systems Management, 7(1), 59-70. Zins, C. (2007). Conceptual approaches for defining data, information, and knowledge. Journal of the American Society for Information Science and Technology, 58(4), 479-493.

AUTHORS BIOGRAPHY

Yilin Huang is a Postdoctoral Researcher at the Section Systems Engineering and Simulation, Faculty of Technology, Policy and Management, Delft University of Technology, Netherlands.

Mamadou D. Seck is an Assistant Professor at the Department of Engineering Management and Systems Engineering, Old Dominion University, USA.

Alexander Verbraeck is a Full Professor at the Section Systems Engineering and Simulation, Faculty of Technology, Policy and Management, Delft University of Technology, Netherlands, and a part-time full professor in supply chain management at the R.H. Smith School of Business of the University of Maryland, USA.