Proceedings of the 2013 Winter Simulation Conference
R. Pasupathy, S.-H. Kim, A. Tolk, R. Hill, and M. E. Kuhl, eds

A CONCEPTUAL DESIGN TOOL TO FACILITATE SIMULATION MODEL DEVELOPMENT: OBJECT FLOW DIAGRAM

Allen G Greenwood

Mississippi State University
260 McCain Engineering Building
P. O. 9542
Mississippi State, MS, 39762, USA

Pawel Pawlewski

Poznan University of Technology
Dept. of Management Engineering
ul.Strzelecka 11
Poznan, 60-965, POLAND

Grzegorz Bocewicz

Koszalin University of Technology
Śniadeckich 2
Koszalin, 75-453, POLAND

ABSTRACT

This paper describes a diagramming methodology, referred to as an Object Flow Diagram (OFD), that is intended to be a key component in the conceptual design of a discrete-event simulation model. It provides an effective means for representing salient system elements and their relationships. It draws upon other popular system diagramming methods, such as IDEF0 and IDEF3, to bring the relevant aspects of these tools to the simulation modeler. It is intended to be easy to apply, with few symbols and constructs, yet robust and comprehensive enough to represent a wide variety of systems. It is simulation software neutral and thus provides a basis for model development in any language. A simple example is used to illustrate the approach. The methodology has been used in industry projects and in simulation courses.

1 INTRODUCTION

It is important in any modeling, but especially in simulation modeling, to have a clear understanding of the system being considered. Modeling is basically representing, in some abstract way, the key elements and interactions that drive a system's behavior as well as its structure. Therefore, it is paramount for modelers to effectively capture and represent their understanding of a system so that they include the correct and the salient aspects of the system in their model. Similarly, modelers need to communicate their understandings and representations to others.

Diagrams, symbolic visual representations of information, effectively support critical thinking and problem solving (Kosslyn 1980; Larkin and Simon 1987). Larkin and Simon (1987) indicate diagrams are superior to verbal descriptions for problem solving since they group together information that is used together, use location to group information about a single element, and support a large number of perceptual inferences.

Modeling is a process – a set of coordinated activities that transform input into output. As a process, modeling transforms needs, knowledge, and data into measures, new knowledge, and satisfied needs. In simulation, the key transform in this process is the simulation model. Execution of the model provides measures of system performance under varying conditions that provide the basis for analysis that ultimately supports decisions and actions. Of course, the quality of the results depends on the quality of the input – the definition and specification for the model.

In simulation, two types of models – conceptual and programmed – are oftentimes used during the modeling process. In a broad sense, one definition of a conceptual model is "a non-software specific description of the simulation model (that will be, is or has been developed), describing the objectives, inputs, outputs, content, assumptions and simplifications of the model." (Robinson 2008). The programmed model is the conceptual model converted to some simulation software environment (e.g. FlexSim, Arena, Witness) for the purpose of execution and analysis.

As in product design, conceptual modeling is a necessary precursor to detailed design. It is dangerous and oftentimes wasteful to begin detail design or modeling without a clear understanding of the problem, system operation and behavior, and objectives of the project (needs of the decision maker). While there is general agreement in the modeling community that a conceptual model needs to precede a programmed model, there is little agreement as to what that conceptual model should be (Robinson 2006; Robinson 2012).

We consider a critical part of a conceptual model to be a visual representation or diagram that depicts the key elements in the system that need to be considered and the relationships among the elements, both in terms of structure and behavior. The diagram should provide a generic description of the system and not represent the elements in terms of any specific simulation software constructs. The diagram is meant to depict how a system works at the level of detail required to meet the objectives of the simulation project. We believe that most conceptual models are, or are composed of, some form of diagram.

The conceptual model, and any supporting diagrams, is not just for modelers. For most models the system is too complex for any one person to have a full understanding of the system and its behavior. As a result, most modeling projects have many stakeholders with widely varied backgrounds, most of whom are non-modelers. Therefore, an effective way is needed to present a collective understanding of the system. Again, a diagram is an effective way to describe a system to a variety of stakeholders. It also provides an effective means to facilitate model validation with the stakeholders.

Since a conceptual model defines and describes system boundaries that will be captured by the model, diagrams provide an effective means to convey scope and to clarify what will, and equally important what will not, be included in the model. Finally, diagrams can effectively provide context for the many and varied stakeholders in a simulation project.

This paper provides a diagramming methodology that can be, and has been, applied to facilitate the conceptual design of discrete-event simulation models. Prior to defining the methodology, a brief discussion of alternative diagramming approaches is provided. The paper concludes with an illustrative example of the application of the methodology.

2 DIAGRAMMING ALTERNATIVES

Since simulation modeling and analysis is applied to a wide variety of systems – from manufacturing to logistics to healthcare and other service providers – any diagramming methodology must be quite generic and flexible. Also, since the stakeholders in a simulation project are typically quite diverse, the diagramming methodology needs to be able to accommodate varying degrees of details and perspectives.

There are many types of diagrams that are used to help represent systems. Those closely related to simulation modeling range from simple flow charts that depict detailed steps in a process to large, complex enterprise architectures that depict key aspects of enterprises.

Discrete-event simulation (DES) modeling includes some special aspects that need to be considered in a conceptual model diagram. Two broad classes of resources -- fixed and mobile (or shared) -- are common in most simulation models and provide the overall structure of the model. Each of these classes has unique and dynamic features and behaviors that need to be represented in a static diagram. Just as important, if not more so, are the relationships among the class element since most DES models focus on flows, albeit mostly process or items flows, between objects. However, in addition to entity flows, information flows and other types of communications between objects must also be considered. There are numerous software products available to transform conceptual models into a programmed model, each of

which uses a different modeling approach and terminology. Having a neutral diagramming approach facilitates communication among different software users and creates a common language for those who are not experts in a particular software.

Most existing diagramming approaches do not lend themselves to effectively developing conceptual simulation models. While many of these approaches provide some support, or at least notions of what to address in a diagram and how to represent it, they either are too simplistic or too extensive (contain too many unneeded features and/or foundational requirements) for discrete-event simulation conceptual modeling, Simple flow charts are often used in the early stages of simulation projects, but they are not rich enough to capture many of the salient system features needed for simulation modeling. Since they are often used in detail code development, they might drive modelers toward too much detail at this stage of the modeling process. Value-stream maps (for example, Jones and Womack (2002)) are also oftentimes used as a starting point for simulation project, but again they do not provide enough detail for simulation modeling.

There are many approaches on the opposite end of the spectrum. Enterprise modeling tools, such as CIMOSA, GRAI, and ARIS, are too broad and massive for use in most simulation projects. See, for example, Giachetti (2010) and Petrie (1992) for information about these tools. Aspects of UML (Unified Modeling Language), a widely used notation for software developers, can be applied to simulation modeling, such as the Component or Object Diagrams for structure and the Activity Diagram for behavior. However, they do not include all of the aspects needed to meet the needs of most simulation modelers and having all aspects resident in a single diagram, as we propose, is much more effective. See, for example, Fowler and Scott (2000) for information about UML tools. Similarly aspects of SysML (Systems Modeling Language), a general purpose modeling language for systems engineering, can be applied to simulation modeling, but as with UML, the toolset is too broad and has too much overhead, yet is incomplete for most simulation applications. For more information on SySML, see for example Holt (2008) and Friedenthal (2008).

The DEVS (Discrete-Event System Specification) formalism provides a visual approach to model specification. However, we believe the representation is typically more applicable later in the model development process and has limited use during conceptual design and in model formulation. For more information on DEVS, see Hwang and Zeigler (2006) and Zeigler et al. (2000).

Since discrete-event simulation models are typically process-based, then a reasonable foundation for modeling diagrams are constructs used in business process modeling. As described below, our approach is process based. Both Onggo (2009) and Onggo and Karpat (2011) use a portion of Business Process Model and Notation (BPMN) in their approach. BPMN is Object Management Group's (OMG) graphical notation that "provides business with the capability of understanding their internal business procedures … and communicate these procedures in a standard manner." (OMG 2013). See also OMG (2010) and Briol (2010) for more information on BPMN

Onggo and Karpat (2010) propose the use of BPMN as a foundation for a conceptual model diagramming methodology. While their approach could be used in discrete-event simulation, albeit the constructs are a bit limited, their focus is on agent-based simulation models.

3 OBJECT FLOW DIAGRAM METHODOLOGY

As the name indicates, the proposed Object Flow Diagram (OFD) methodology primarily focuses on the key objects in a system that need to be modeled via simulation and the relevant flows that occur between those objects.

The OFD methodology is introduced in Beaverstock et al. (2012) as a conceptual modeling approach to facilitate the simulation model development process. It is intended to help the modeler document the system being considered and think through many of the initial modeling issues, e.g. defining system boundaries, identifying key components, assessing the level of detail needed. This paper more fully de-

fines and describes the approach, as well as providing extensions and refinements to the basic methodology.

The OFD methodology is designed to be software neutral so that it can be used by modelers with a wide range backgrounds and can be an effective means of communication with project stakeholders that have little knowledge of simulation software. Being software neutral also helps modelers avoid moving too quickly to a modeling approach or programming prematurely without adequate system understanding and problem definition. As a result, the OFD approach is not conducive, by design, to automatic model code generation.

Many of the symbols used in an OFD are based on the IDEF (Integration DEFinition) methodology and especially IDEF0. See Giachetti (2010) for a summary of the IDEF methodology. As shown in Figure 1, a symbol using IDEF0's ICOM notation uses all four sides of the symbol. That is, the left side is for inputs to, in a simulation model's case, the resource; flow items that are transformed or consumed by the operation. The right-side of the symbol is for outputs from the resource; flow items that are produced by the operation. The bottom side is for "mechanisms," other resources used to support the operation, such as operators, empty containers, etc. The top side of the symbol is for "controls," those things that constrain operations or are conditions used by functions in the model, e.g. messages and downtimes. OFDs, like IDEF0 diagrams, are hierarchical – details of an object can be expanded in a lower-level representation of the object.

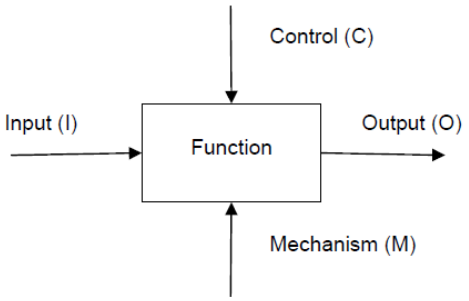


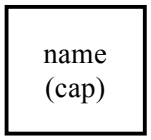
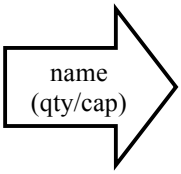
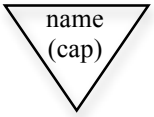
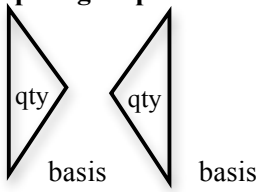
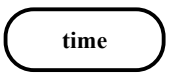
Figure 1: IDEF0 ICOM notation (B4).

As indicated earlier, the methodology, symbols, and terminology are simulation software neutral. Use of OFDs can help simulation modelers develop better models regardless of the software they use to implement their models.

Most discrete-event simulation models focus on a series of operations (process, transport, store, group/ungroup) that are performed on discrete items (parts, people, packages, etc.) as they flow through a system of resources. Therefore, the OFD primarily takes the perspective of the item flowing through the system. However, it considers mobile (typically shared) resource flows as well. While some simulation software refer to the items flowing through the model as entities or transactions, we refer to these as flow items – we believe this to be a more descriptive term.

The symbols that are used in an OFD, along with a brief functional definition of each symbol, are presented in Tables 1 through 3. Table 1 contains the symbols used to represent basic operations – these include process, transport, store, and group/ungroup. Since many operations include known delays, we include the duration symbol in this category. Thus, an operation symbol and duration symbol are oftentimes used together. The operations symbols all use the ICOM (input, control, output, and mechanism) notation from IDEF0. That is, the sides of the symbols are significant as described above.

Table 1: Symbols denoting basic operations.

OFD Symbol	Functional Description
Process 	Transformation of a flow item. Typically involves a known delay and thus includes a duration symbol. *name* = identifier *cap* = capacity of the resource; maximum number of items that can be contained within the object. If capacity is not an issue, denote (inf) for infinite.
Transport 	Transportation or movement of a flow item. Typically involves a known delay and thus includes a duration symbol. However, if associated with a shared (mobile) resource, the duration may be determined by the speed of the mobile resource and distance traveled. Also, distance transported may be annotated on the symbol. *name* = identifier *qty/cap* = the quantity of items moved at a time and/or the total capacity of the resource (maximum number of items that can be contained within the object). If capacity is not an issue, capacity denoted as (inf) for infinite.
Storage 	Temporary holding area for flow items. *name* = identifier *cap* = total capacity of the resource (maximum number of items that can be contained within the object). If capacity is not an issue, capacity denoted as (inf) for infinite.
Group/Ungroup 	Combining items into a single unit, or conversely, splitting a unit into multiple items. *qty* = number of items grouped or number of items ungrouped or split (copied) *basis* = criterion for grouping or ungrouping, e.g. combine based on product type. May involve a known delay before or after grouping; thus it may include a duration symbol. Oftentimes the symbols are associated with another operation and thus if the symbol on the left is placed on the left or input side of an object, it denotes a combine operation. Similarly, if the symbol on the right is placed on the right or output side of the object, it denotes a split or separation operation.
Duration 	Activity or task performed by a basic operation object. Therefore, oftentimes used in conjunction with another symbol. There may be multiple processes conducted sequentially, such as a setup and then a basic process activity. *time* – time to complete an activity

Note, for all operations symbols:

1. They usually represent a fixed-location physical resource.
2. For all, except the Duration symbol, the sides are significant. They use IDEF0's ICOM (Input-Control-Output-Mechanism) notation.

- Left-side is for inputs to the resource; the primary item(s) that are transformed or consumed by the object.
- Right-side is for outputs from the resource; the primary item(s) that are produced by the object.
- Bottom-side is for "mechanisms," other resources used to support operations of the object.
- Top-side is for "controls," those things that constrain operations or are conditions for functions of the object

Table 2 contains symbols that are flow related. This includes the flow of items (often referred to as routing) and the flow of resources (mobile or shared). Since routing implies alternative paths, decision points and criteria need to be specified. This is done through the Decision symbol. Flow items need to be created and destroyed beyond the system boundaries. Therefore, there must be a source to create the flow items and a sink to remove them from the model at the appropriate time. These are denoted by the Source and Sink symbols in an OFD. Mobile resources travel or flow through the system as well and their paths need to be identified and specified. Usually, processing objects need more than just physical flow items; they often need information, and other forms of communication, from other objects.

Table 3 contains the remaining OFD symbols – they support the operation- and flow-oriented symbols. Resources – both fixed and mobile – are not always available; therefore, there is a Downtime symbol that indicates the time between breakdowns and the duration of the breakdown. Performance measures are used to decide which of the alternatives being considered is better. Simulation can collect information on any measure of performance and most software automatically collect statistics on many common measures. However, it important to identify early in the modeling process what measures are the most important. Those measures are noted on the OFD through a special symbol. The OFD is intended to be used more like a sketch than a final drawing. Therefore, annotation is encouraged. A Note symbol is used for clarification; a Questions symbol is used to call out a point of uncertainty that needs further definition or specification.

4 APPLYING OBJECT FLOW DIAGRAMS

Since the methodology is intended to be used to help modelers identify and understand the key components of a system and their interactions, and to convey that information to project stakeholders, there is considerable latitude in how it is applied. After all, it is a model itself, albeit a conceptual model, and just like the resulting simulation model, it should only be as complex as it needs to be to be in order to address the problem at hand. Therefore, the OFD symbols should be used sparingly, it should only be as detailed as needed to adequately represent the system so that it can be modeled appropriately.

Table 2: Symbols denoting flows between operations.

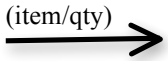
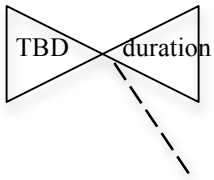
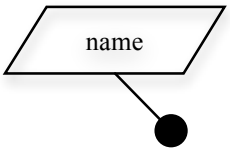
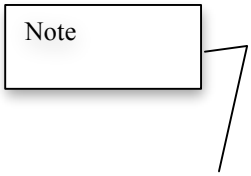
OFD Symbol	Functional Description
Flow/Route (item) 	Link between operations; route that items flowing through the models take. The following two parameters are optional - it depends on if clarification is needed. For example, if a single type of item is flowing through the model, then this can be stated as a note on the diagram and not on each link. *item* = name of the item(s) following the route *qty* = quantity of items that flow together Flows usually require time for items to move between objects; therefore, a duration symbol may be included. However, if the flow is made through a shared (mobile) resource, the duration may be determined by the speed of the mobile resource and distance traveled. Also, distance transported may be annotated on the symbol.
Decision 	Decision rule typically used to either route an item out or "pull" an item into an operation. *criterion* = rule used to make the selection Typically added to an operations symbol. If placed on the left or input side of the object, it denotes a pull operation; if placed on the right or output side of the object, it denotes a routing decision.
Source 	Creates items that flow in the model. Defines the input system boundary. *freq* = time between arrivals *qty* = quantity of items that enter the system at each arrival event
Sink 	Destroys items that flow through the model; output system boundary.
Shared (mobile) resource 	Performs activities at multiple operations. Typically involves processing or moving items, repairing equipment. *name* = identifier *qty* = number of resources needed to perform an operation
Travel (resource) 	Path followed by shared (mobile) resource to perform activities at, or between, various objects.
Communication 	Link between objects that involves information, messages, etc. These links do not represent the flow of an item.

Table 3: Supporting symbols

OFD Symbol	Functional Description
Downtime 	Denotes resources, both fixed and shared(mobile), may become unavailable, or down. Downtimes can occur for a variety of reasons, e.g. breakdowns, preventative maintenance, operator breaks, shift schedules. *TBD* = time between downtimes *duration* = how long the resource is down Objects may undergo several types of downtime (e.g., planned and unplanned breakdowns). Multiple downtimes can be represented with multiple symbols or with a single symbol and a note describing the downtimes.
Performance Measure 	System variable that is of particular interest. An output of a simulation model; sometimes referred to as a response variable or dependent variable. *name* = identifier Attached to the object of interest; e.g. "average content" might be associated with a Storage symbol or "utilization" might be associated with an operation or a shared (mobile) resource.
Note/Comment 	Annotation that provides information about the system, an object, a flow, etc.
Question 	Annotation that raises one or more questions that need to be addressed. It identifies the need for additional information, or denotes issues that need to be discussed.

While an OFD can be created in many ways, the following are the typical and suggested ordered steps for initially creating an OFD. Since the conceptual design process is iterative, the construction of an OFD is also quite iterative. It is constantly being modified and updated as more is understood about the system and as more detail is needed. Detail is oftentimes added hierarchically in order to enhance readability and meet the needs of various stakeholders.

1. Denote input and output system boundaries. This is based on where flow items enter and leave the system being considered and modeled. Thus, indicate the items' source(s) and sink(s). Typically, a source includes the specification of the frequency and quantity of arrivals.
2. Indicate all key physical fixed resources (e.g. machines, workstations, conveyors, storage areas) using the appropriate OFD symbol.
3. Connect the resources with directional arrows denoting the flow of items through the system using the Flow/Route (item) symbol.
4. Identify points in the flow where choices in the flow need to be made. These points are noted using the decision symbol. The criterion for the decision is also specified.

5. Identify points in the flow where items are grouped or ungrouped (e.g. for transport or through assembly) and denote in the diagram using the OFD symbols. Specify the quantity of items that are combined.
6. Identify where known delays occur (e.g. for processing or transport) and denote in the diagram using the OFD Duration symbol. Specify the time duration (for processes, this is typically a probability distribution).
7. Associate shared resources (also referred to as mobile or dynamic resources) with processing and transport operations.
8. Identify any flows between objects other than item flows, e.g. communication and information flows.
9. Link shared resource associations to form resources paths using the Travel (resource) symbol. This step identifies missing shared resource paths since most shared resources operate in cycles or travel in "loops."
10. Identify and note on the diagram any planned or unplanned resource downtimes that need to be addressed in the model.
11. Identify and note on the diagram the key performance measures that should be captured by the model. These are usually the factors that are considered when deciding among competing alternatives.
12. Annotate the diagram with comments, notes, and questions.

5 ILLUSTRATIVE EXAMPLE

A simple manufacturing cell is used to illustrate the application of the OFD methodology. Of course, OFDs can and have been applied to a variety of operations systems, such logistics, healthcare, and other service industries.

Figure 2 provides an OFD representation of the example system; a plan view of the system is shown in the insert. The cell contains five pieces of processing equipment (labeled SR1-SR5), two fixed transporters (conveyors labeled SR6 and SR7), six storage locations (labeled I1 through I5 and Q-SR4), and two operators (labeled A and B). Component items (i1 through i4) go through several processing and assembly operations. In order to illustrate hierarchy, the turntable (SR3) is broken down into its main components and thus is represented as SR3.1, SR3.2, and SR3.3. These are its processing positions; TT represents the turntable resource.

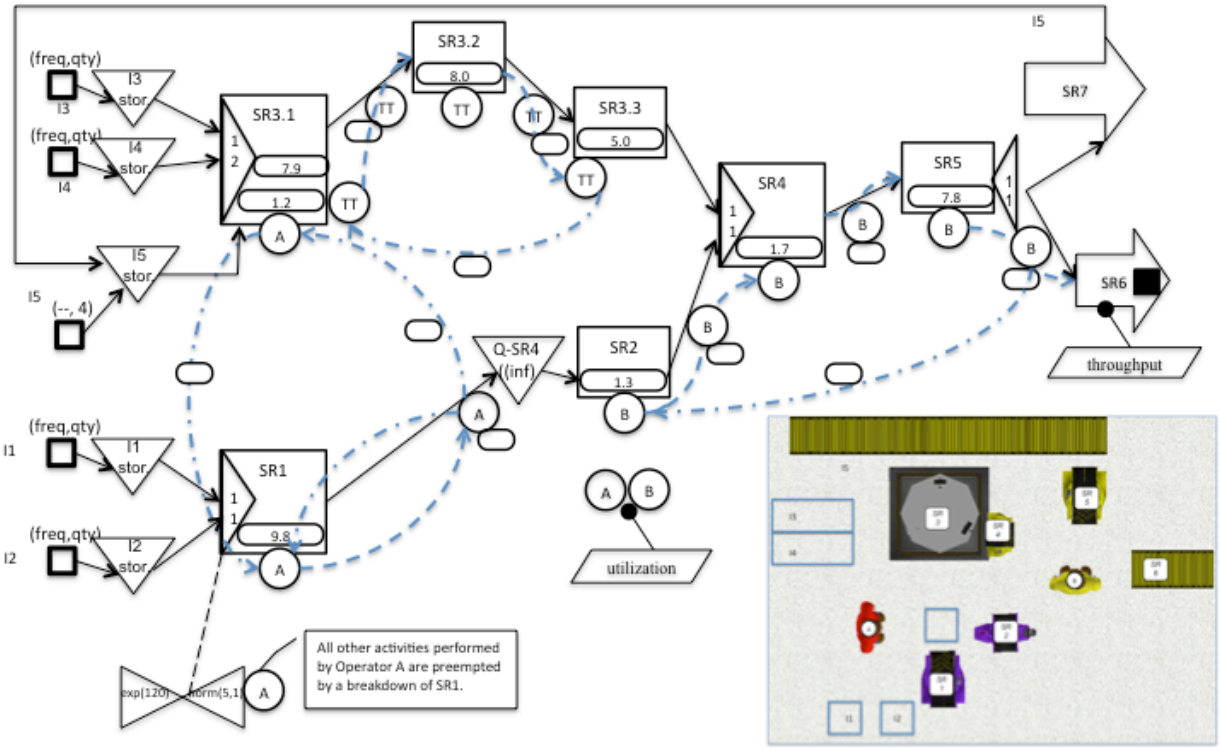


Figure 2: OFD of example system (shown in insert).

The cell operation begins with the assembly of two items, i1 and i2, at workstation SR1 by operator A. The assembly is then moved to SR2 by Operator A. Operator A also sets up the turntable machine at position SR3.1. In addition, Operator A repairs machine SR1 when it breaks down.

Concurrently to the above operations, component items i3 and i4 are processed at the turntable (operations at SR3.1, SR3.2, and SR3.3) using fixture i5. When complete, the subassembly is automatically transferred to SR4. Operator B moves the subassembly from SR2 and combines it, at SR4, with the subassembly produced by the turntable. Once the assembly is complete, operator B moves the product to SR5 for final processing. After processing, operator B separates the fixture and assembly and move the assembly to the outgoing conveyor, SR6. The fixture is automatically returned to its storage location via conveyor SR7.

The Note symbol is used to indicate that a breakdown of SR1 will preempt Operator A so that SR1 will be repaired as quickly as possible. Similarly, the Performance Measure symbols are used to indicate that the throughput of the cell and the utilization of operators A and B are important measures.

The shared or mobile resource paths are constructed from the locations of the calls for service from the fixed resources. It is important to note that if only the items' flows are considered, then some of the shared (mobile) resource travel segments would not be identified. It is only by connecting the activities for operators A and B and turntable TT that resource paths or cycles are completed. Those "missing" flow segments are shown with dot-dashed flow lines, rather than the dashed flow lines that are created by the items' flows. This notion is an important extension of the methodology. Approaches that are only process driven, i.e. processes cause the demands on resources(e.g. a part moves to a machine) will overlook situations where the resources have tasks to complete which are not flow related, e.g. return to a previous position to await a call from another resource to perform another task.

6 CONCLUSIONS

The Object Flow Diagram provides an effective means to represent systems being modeled using discrete-event simulation. OFDs are succinct, comprehensive, software-neutral diagrams that provide a solid foundation for conceptual model development and an effective means for communication with stakeholders that are not simulation experts. While based on concepts from other diagramming methodologies in process, enterprise, and computer system modeling, OFDs focus on the aspects of systems that are the most relevant to discrete-event simulation modelers.

REFERENCES

Beaverstock, M., A. Greenwood, E. Lavery, and W. Nordgren. 2012. *Applied Simulation Modeling and Analysis using FlexSim*. 3rd ed. Orem, Utah: FlexSim Software Products, Inc.

Briol, P. *The Business Process Modeling Notation BPMN 2.0 Distilled*. Ingenierie des Processus.net.

Fowler, M., and K. Scott. 2000. *Design of Enterprise Systems*. Boca Raton, Florida: CRC Press.

Friedenthal, Sanford. 2008. *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann / The OMG Press.

Giachetti, R. E. 2010. *UML Distilled.* 2nd edition. Reading, Massachusetts: Addison-Wesley.

Hwang, M. H. and Zeigler, B. P. 2006. "A Modular Verification Framework using Finite and Deterministic DEVS." In *Proceedings of 2006 DEVS Symposium*, pp. 57–65, Huntsville, Alabama.

Jones, D., and J. Womack. 2002. *Seeing the Whole: Mapping the Extended Value Stream.* Brookline, Massachusetts: The Lean Enterprise Institute.

Holt, Jon. 2008. *SysML for Systems Engineering*. The Institution of Engineering and Technology.

Kosslyn, S. M. 1980. *Image and Mind*. Harvard University Press. Cambridge, MA.

Larkin, J. and Simon, H. 1987. "Why a Diagram is (Sometimes) Worth Ten Thousand Words," *Cognitive Science*, 11, pp. 65-99.

OMG (Object Management Group). 2010. "BPMN 2.0 by Example." Version 1. June 2010. http://www.omg.org/spec/BPMN/20100601/10-06-02.pdf

OMG (Object Management Group) 2013. "Business Process Model and Notation." http://www.bpmn.org Accessed Jul 12, 2013.

Onggo, B. S. S. "Towards a Unified Conceptual Model Representation: A Case Study in Healthcare." *Journal of Simulation*, 3, pp. 40-49.

Onggo, B S.S. and Karpat, O. 2011. "Agent-Based Conceptual Model Representation Using BPMN." In *Proceedings of the 2011 Winter Simulation Conference*, Edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Petrie, C. J. (Editor). 1992. *Enterprise Integration Modeling: Proceedings of the First International Conference*. Cambridge, Massachusetts: MIT Press.

Robinson, S. 2006. "Conceptual Modeling for Simulation: Issues and Research Requirements." In *Proceedings of the 2006 Winter Simulation Conference*, Edited by L. F. Perrone, F. P. Wieland, J. Liu, B. G. Larson, D. M. Nicol, and R. M. Fujimoto. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Robinson, S. 2008. "Conceptual Modelling for Simulation Part I: Definition and Requirements." *Journal of the Operational Research Society* 59 (3): 278-290.

Robinson, S. 2012. "Tutorial: Choosing What to Model – Conceptual Modeling for Simulation." In *Proceedings of the 2012 Winter Simulation Conference*, Edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A.M. Uhrmacher. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Zeigler, B., Praehofer, H., and Kim., T. G. 2000. *Theory of Modeling and Simulation, 2nd Ed*. Academic Press, San Diego, CA

AUTHOR BIOGRAPHIES

ALLEN G GREENWOOD is Professor of Industrial and Systems Engineering at Mississippi State University, USA where he teaches systems simulation, enterprise systems engineering, and project management. His research interests/expertise include the design and analysis of production and project systems; simulation modeling, analysis, and optimization; and the design and application of decision-support systems. He is coauthor of *Applied Simulation Modeling and Analysis using Flexsim*. His email address is greenwood@ise.msstate.edu.

PAWEL PAWLEWSKI is an associate professor at the Department of Computing and Management at Poznan University of Technology. His research interests include organization of manufacturing systems, monitoring of operations management, reengineering and IT application for logistics, process modeling, simulation and optimization. He is author or co-author over 100 manuscripts including books, journals and conference proceedings. He is managing director of SOCILAPP Simulation and Optimization Center in Logistics and Production Processes. His email address is pawel.pawlewski@put.poznan.pl.

GRZEGORZ BOCEWICZ is an assistant professor at the Department of Electronics and Computer Science of Koszalin University of Technology in Poland. He obtained a Ph.D. degree in Computer Sciences from the Wrocław University of Technology, Poland. His research interests are in the areas of the operational research, decision support systems, constraints programming techniques. He is the author and co-author over 100 manuscripts including two books, international journals, and conference proceedings. His email address is bocewicz@ie.tu.koszalin.pl.