Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

SIMULATION MODEL GENERATION OF DISCRETE EVENT LOGISTICS SYSTEMS (DELS) USING SOFTWARE DESIGN PATTERNS

Timothy Sprock
Leon F. McGinnis

Georgia Institute of Technology
School of Industrial and Systems Engineering
755 Ferst Drive NW
Atlanta, GA 30332-0205 USA

ABSTRACT

To provide automated access from a formal system model to multiple analysis tools, such as discrete event simulation or optimization, we extend current model-based systems engineering (MBSE) methodologies by introducing a new model to model transformation method based on object-oriented creational patterns from software design. Implemented in MATLAB's discrete event simulation tool, SimEvents, we demonstrate the methodology by generating two distinct use cases based on a distribution supply chain and manufacturing system.

1 INTRODUCTION

Automated access to multiple computational analyses from a single model of the system of interest is typical in many engineering disciplines, e.g., in mechanical design both finite element analysis (FEA) and computational fluid dynamics (CFD) from a single CAD model. However, in contemporary practice, supply chain logistics analyses are often purpose-built to answer specific questions, with an implicit system model and many possible analysis implementations depending on the question, the instance data, and the solver. Automated and cost-effective access to multiple analyses from a single conceptual model of the target supply chain logistics system would provide much broader support for operational decision making and system optimization, and that is the motivation for the research reported here.

Over the past twenty years, the software engineering community has faced similar issues, and one approach to solving the problem has been the development of automated code generation technologies. One of the most widely-known is Model Drive Architecture® (OMG MDA® 2003) promoted by the Object Management Group (OMG ®). In MDA, a representation of the desired computation is referred to as the "platform independent model" or PIM while the desired implementation is the "platform specific model" or PSM.

In both mechanical design and software engineering, one key to automation is a formal model of the system of interest, i.e., the mechanical part or the desired computation. The target model in both cases is inherently formal, either as a mathematical formulation or as executable code. Thus, it seems reasonable to infer that a basic requirement for automating the generation of simulation models will be a formal statement of the source model, i.e., the system of interest. Given that, the remaining challenge is to devise a generic and re-usable method for translating the source model into the target model.

This research has been motivated by the desire to achieve similar results for the automation of OR analysis model generation, by adapting OMG's model transformation approach. There are, however, significant differences between generating code and generating discrete event simulations in commercial

off the shelf (COTS) tools. These differences provide a unique set of challenges for adapting existing model transformation methodologies (discussion in Section 3.4).

In this paper, we describe a transformation methodology, rooted in object oriented design patterns, which exploits the network structure inherent to many supply chain logistics systems. The proposed transformation methodology is implemented using the SimEvents® tool, integrated with MATLAB®; however, the approach can be extended to other simulation platforms and other types of analyses (discussion in future work, Section 6).

2 DISCRETE EVENT LOGISTICS SYSTEMS

Discrete event logistics systems (DELS) are a class of dynamic systems that are defined by the transformation of discrete flows through a network of interconnected subsystems (Mönch et al. 2011). The DELS domain includes systems such as supply chains, manufacturing systems, transportation networks, warehouses, health care delivery systems, etc. DELS are inherently complex systems due to the large scale of the networks, the dynamic nature of interactions between actors, and the randomness of both the external and internal environments. This makes any decision making process difficult and implies the need for a wide range of modeling and analysis methodologies.

2.1 The DELS Conceptual Model

An object oriented modeling (OOM) paradigm is a natural way to model a complex system because it builds upon the domain expert's ability to view a system as collections of related objects, including attributes of those objects, sub-components of those objects, and groupings of similar objects (Coad and Yourdon 1991). OOM facilitates modular design and promotes reusability. OMG's SysML™ (2012), an extension of the UML, provides an OOM environment that is used in many system engineering design paradigms, such as Model-Based Systems Engineering (MBSE) (Estefan 2007) or Object-Oriented Systems Engineering Method (OOSEM) (Friedenthal, Moore, and Steiner 2009). SysML has proven to be a powerful modeling language for system applications in a diverse range of engineering domains, such as electrical, mechanical, and industrial (Huang, Ramamurthy, and McGinnis 2007; Johnson et al. 2007; Peak et al. 2007; Shah 2010; Thiers and McGinnis 2011; Wu et al. 2011)

The formal domain modeling used in software engineering for designing and developing systems has been extended to the DELS domain, resulting in a conceptual model for supply chain engineering that integrates a SCOR-compliant domain specific language (DSL) with an object-oriented supply chain reference architecture specified in SysML (Sprock and McGinnis 2014). SCOR (2012) is a well-known diagnostic and benchmarking tool for evaluating and comparing supply chain activities and performance. Using SysML, SCOR's commonly reused elements have been captured as a collection of stereotypes, and the patterns for process flows as activity diagrams. Additionally, the process-oriented SCOR model has been augmented with a model for the physical architecture of the supply chain. Since there is a small intersection between the supply chain and MBSE communities, implementing SCOR in SysML bridges some of the semantic gap by making the conceptual model as familiar as possible to all stakeholders.

Because very few analysis tools are designed using the same formalism, the ability to directly exchange information between analysis tools is limited. The DELS conceptual model provides a common formalism to exchange information between tools and thereby construct multiple transformations between different analysis tools.

2.2 Discrete Event Simulation

Due to the inherent complexity of DELSs, analytic models are often intractable and require significant assumptions to model the observed "real world" behavior. For these analysis use cases, discrete event simulation is an attractive option to evaluating the expected behavior and performance of the system of interest. However, there are significant hurdles to building, running, and analyzing simulation models.

One is the range of simulation analysis tool capabilities and characteristics (Son, Jones, and Wysk 2000), which means there is no "canonical" statement of a simulation problem, as there is, e.g., for optimization. This difficulty is compounded by the lack of model/data separation. In optimization, for example, a single formulation of the transportation problem can be reused for datasets having different numbers of sources, sinks, and transportation arcs. The corresponding simulation models, however, would have to be individually created in some way, either manually or via automation. Both of these challenges provide a great incentive to automate the generation of discrete event simulations in a generic and reusable way.

In this research, we propose a method to automatically generate the DES structural model in SimEvents. Since a single simulation model can be reconfigured to answer multiple questions and perform multiple analyses, we leave the task of specifying the appropriate analysis to answer the question at hand to future work; e.g. simulation configuration parameters, input scenarios, output results to collect, etc. SimEvents was selected because of its integration with the MATLAB workspace and OOP environment, which provides an extensible scripting language and facilitates the implementation of the proposed methodology (specific discussion to follow in Section 4). Also because of its integration with popular optimization and statistics tools, MATLAB is also an attractive integrated development environment to implement the one model-many tools paradigm.

3 AUTOMATED MODEL GENERATION

3.1 Historical DES Generation Approaches

Traditionally, automatic simulation generation approaches have relied on templates and ad-hoc scripts to generate simulation models in popular COTS tools. Cope et al. (2007) provide some review on integrating DES and supply chain modeling as well as efforts on generic and automatic simulation generation. Son, Jones, and Wysk (2000) propose a methodology to automatically generate simulation models from neutral libraries of simulation components for manufacturing job shops. However, there also has been research that focuses on developing methodologies that do not rely on COTS DES tools. Chatfield, Harrison, and Hayya (2006) recognize that a well-defined library of simulation objects and a well-defined schema to store the instance data can enable the generation of the DES through a simple mapping. They implement this strategy with their own object-oriented supply chain simulation tool and an XML-based system representation. Biswas and Narahari (2004) also model the supply chain infrastructure with a library of carefully designed objects with well-defined interactions. They provide an intermediate workbench model that potentially provides automated access to a range of analysis models, including DES. This strategy is also realized through the development of their own tools. These three methodologies provide insight into good strategies for developing an automated simulation generation methodology for COTS tools.

3.2 M2M Transformations and The Meta-modeling Architecture

MDA advocates the creation of a platform-independent model of software at a higher level of abstraction, which can be transformed as needed into platform-specific and executable models; adapting MDA would separate the development of a DELS model from its implementation in a specific simulation software package; the implementation would be constructed automatically using a reusable model to model (M2M) transformation.

Conceptually, the model transformation methodology involves an architecture that organizes the models involved in the transformation, and the transformation process itself, which consists of a mapping and associated transformation rules. The OMG four-layer meta-model architecture provides a hierarchical organization to this process; the four layers are MOF, SysML/UML, User Model, and Instance Model, illustrated in Figure 1.

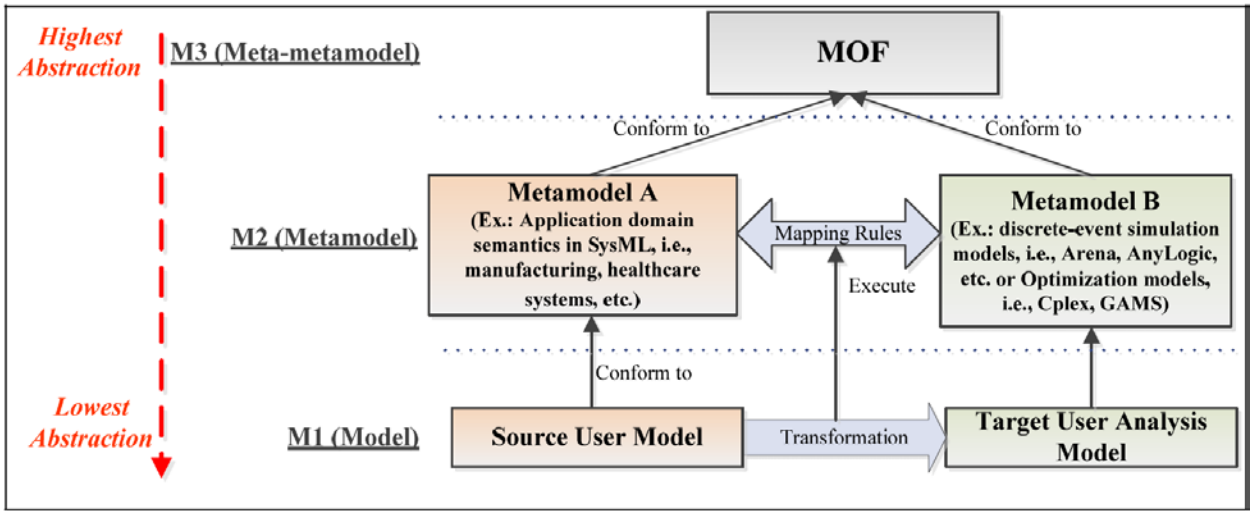

Figure 1: OMG M2M transformation framework (Batarseh and McGinnis 2012a).

The transformation methodology consists of two components: the transformation definition, which is a mapping between classifiers in the source and target models, and then a transformation engine that transforms instances of each classifier in the source model to a corresponding and conforming instance in the target model (Figure 1). A survey of model transformation approaches is given by Czarnecki and Helsen (2006).

3.3 Current State of the Art

Current M2M methods, such as OMG's QVT (2011) or MOFM2T (2008), can execute transformations from SysML to an XML or structured text specification of the target analysis, e.g., AnyLogic™ or SimEvents™ respectively. Prior research has established the viability of this methodology to create a DSL in SysML, then describe the system of interest using the DSL, and finally transform a description of a system model specified using the DSL into a target analysis language, such as discrete event simulation using Arena®. (Batarseh and McGinnis 2012a; Batarseh and McGinnis 2012b; McGinnis and Ustun 2009; McGinnis et al. 2011). An illustration of this mapping process for the DES tool AnyLogic is provided in Chapter 8 of Huang (2011).

3.4 Challenges in Applying the OMG/MDA Approach

Difficulties arise in applying current M2M methodologies for code generation to generating discrete event simulation. Because of its imperative nature, applying MDA's M2M approach to the widely varying tool specifications results in ad-hoc transformations that aren't reusable or extensible; essentially defeating the original intention. Moreover, the MDA transformation approach requires the target analysis tool to store its models in a well-structured and accessible format, for which there is a published schema. Many popular simulation tools fail to satisfy this requirement. However, often these simulation tools provide an object oriented or imperative programming environment that allows the modeler to construct the simulation indirectly (not using the typical drag-and-drop model construction interface). This provides a path to developing a transformation that does not rely on the translation of XML to another structure text format.

Second, whereas code generation is a translation of syntax, the typical simulation model is only an approximate morphism of the corresponding system model, which makes the PIM to PSM transformation exceptionally difficult. That is, it requires significantly more simulation blocks and elements to express a simple function or structure, and the mapping is different for each different simulation language. In

response to the difficulty and complexity of the mapping process for DES tools, Huang (2011) suggests that one possibility for handling this complexity would be to provide reusable behavioral and structural libraries in the target simulation language.

Two additional issues with automating the transformation process are identified in McGinnis and Ustun (2009). First is the management of large amounts of instance data where the total number of unique elements is small, yet the set of instances of each element is large. This instance data usually is natively stored in a relational database, and it is desirable for a M2M transformation to ingest the relational database as the source model's instance data. This implicitly suggests that there is a low priority on completely specifying the user model in SysML, in favor of relying on the conceptual model to provide schema to the relational database and patterns on how the elements will be assembled in the target model. Secondly, additional information is required to transform a conceptual model, which needs to be captured in the conceptual model itself or provided during the transformation process. Whereas current M2M methods used to generate simulations are typically imperative, there is an opportunity to elaborate the transformation with additional knowledge on the construction of the target analysis.

Finally, most common DES language models do not conform rigidly to the four-layer meta-model architecture—in fact, it is rare indeed to find a published meta-model. However, the MOF Core explicitly states the four-layer architecture is not supposed to be rigid, but merely should be reflective and allow navigation between a classifier and its instance. This is particularly convenient because it is not clear that discrete event simulation languages, as currently implemented, clearly conform to that architecture. This opens the possibilities for a more natural transformation architecture that maps classifier to classifier and then transforms instance to instance without being constrained by the rigid four-layer architecture.

4 THE PROPOSED M2M TRANSFORMATION METHODOLOGY

4.1 The Transformation Architecture

The proposed model transformation methodology consists of an architecture that organizes the transformation process, a two-level approach to transformation, and—in a departure from MDA—the use of software factories (Gamma et al. 1995) rather than contemporary M2M tools for executing the final transformation.

The architecture, shown in Figure 2, presents a two-step transformation that generates an intermediate model before generating the simulation, and is a departure from the standard MDA M2M approach. This addition of the intermediate model provides two advantages. First, it provides a method to extend the functionality of the target simulation language. Second, it provides flexibility and reusability by allowing the user to change the target analysis tool by implementing a different transformation in the second step.

The architecture also captures the transformation process as a two-level transformation. The higher-level transformation maps the abstract DELS concepts, such as subsystem definitions, from SysML to the target analysis tool, and the transformation outputs the DELS model library for the tool. The lower-level transformation maps classifiers in the conceptual model in SysML to members of the DELS model library, and the transformation takes instance data from a relational database and outputs simulation blocks for the target tool. Essentially, the second transformation is a formal and reusable method for executing a 'when you see one of these in the instance data, generate one of these in the target language' process.

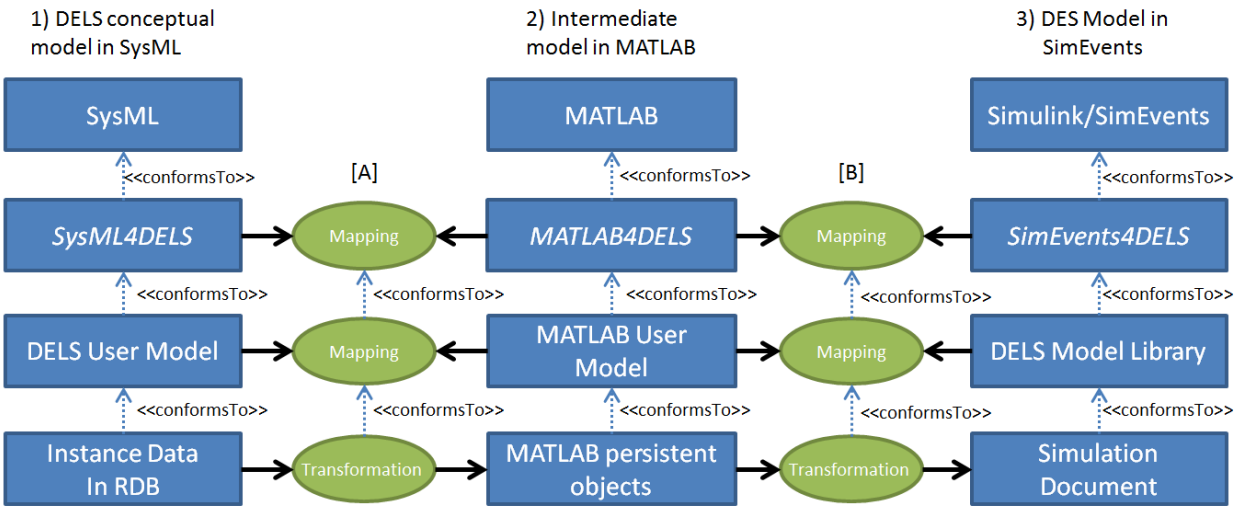

Figure 2: Two-step, two-level transformation.

4.2 Benefits of this Architecture

While designed as object-oriented languages, the OO functionality of the provided modeling constructs is limited in most COTS tools. One desirable extension would implement simulation objects with extensible properties and methods, which may, for example, allow them to store the results of an optimization routine and make corresponding modifications to themselves. Therefore, one benefit of this architecture is the opportunity to extend the functionality of the target simulation language. We have implemented a method that pairs a MATLAB object with its corresponding SimEvents component by requiring the MATLAB object to store a pointer to the SimEvents object. Then we can treat the pair of objects as if they were a single object and the functionality of a class of SimEvents blocks can be extended by adding new properties and methods to the MATLAB class definition.

For example if the target analysis requires a specific metric to be reported by a SimEvents objects, its corresponding MATLAB object executes the desired modification. Moreover rather than writing the metric to a global table or the MATLAB workspace, the MATLAB object stores the information which can be accessed during the analysis of the simulation results or returned to an optimization routine. More germane to the generation of the simulation itself, the SimEvents blocks are extended to provide methods to clone and modify themselves, which is integral to executing the transformation from the intermediate model to the SimEvents model.

Second, the transformation enables a solution to the instance data problem presented above by transforming instance data stored in a relational database, such as MS Access, to persistent objects in the MATLAB workspace, and finally to SimEvents blocks. Moreover, the schema of the relational database is contained in the DELS User Model, and the database tables can be automatically generated using an implementation of *UML2RDBMS* (OMG QVT 2011). This is a particularly important development because it provides a bridge from sources of industrial data to the simulation platform. In the process, this conversion, together with the coupling method described above, explicitly enables the OOP techniques used to generate the simulation.

Due to the complexity of a limited number of highly reusable components in DELS models, it is advantageous to build and debug the reusable simulation components in their native simulation environment and then incorporate them into a model library. One important aspect of the DELS Model Library in SimEvents is the development of a robust interface block, which is automatically generated depending on the flow types and message handling protocols of the target subsystem. This interface allows each construct in the model library to be reused repeatedly with modification through

parameterization or variation points; e.g. the method to instantiate a direct ship transportation channel with a single source and destination is the same as a milk-run consolidation transportation channel with *n* sources and one destination.

There are two distinct benefits from designing the model library this way. First, the simulation distinguishes between the types of flow between subsystem blocks, which provides flexible and extensible tools for designing protocols for routing entities throughout the network. Second, users are able and encouraged to create and register their own variants of the subsystem blocks, which are cloned from the model library during the generation process.

Finally, we adopt an OOP approach to implementing the transformation which is in contrast to the imperative and declarative languages defined by the QVT standard. This approach is a natural extension of the benefits discussed in this section. Specifically, the object oriented data storage method complements the pairing of MATLAB and SimEvents objects, and allows us to embed useful machinery into the transformation. This results in a transformation that is more reusable and extensible than current model transformation languages. In the next section, we provide detail on the proposed method, which integrates each of the developments discussed in this section, to transform the intermediate model into the target simulation model.

4.3 Generative Methods Based on Creational Patterns

One of the strategic goals in the development of a model transformation method is to promote reusability across multiple, if not all, DELS subdomains. This reusability is achieved by relying heavily on the network flow abstraction discussed above and exploiting design patterns that abstract the instantiation process. Therefore, a significant contribution of this paper is the development of a generative method that exploits the network structure to construct a reusable and extensive model transformation method through the usage of creational patterns adapted from the software engineering domain.

Creational patterns abstract the instantiation process by encapsulating the knowledge about which concrete classes the system uses and hiding how instance of these classes are created and assembled (Gamma et al. 1995). Therefore, deploying this method provides flexibility on what is created and how it is created, and ultimately allows configuring a system with objects that vary widely in structure and functionality. While there are several different strategies for assembling these creational patterns into a useful transformation method, Figure 3, each of these strategies begins with the abstract factory pattern.

1) Abstract Factory pattern
2a) Single parameterized concrete factory
2b) Multiple concrete factories, one for each product to be created
3) Prototyping from Model Library
4) Builder Classes to make modifications

Figure 3 Organization of Generative Process.

The abstract factory pattern is used to take advantage of the network flow abstraction, and is implemented as the NetworkFactory, NodeFactory, and EdgeFactory classes (Figure 4). The AbstractFactory pattern declares an interface for creating products, such as CreateNode(), and declares an abstract class for each of the products that can be produced by the AbstractFactory, the corresponding Node class. A significant amount of reusability is realized by designing the transformation process at an abstract level and then relying on differentiated concrete subclasses to instantiate a wide variety of concrete products that conform to the flow network abstraction.

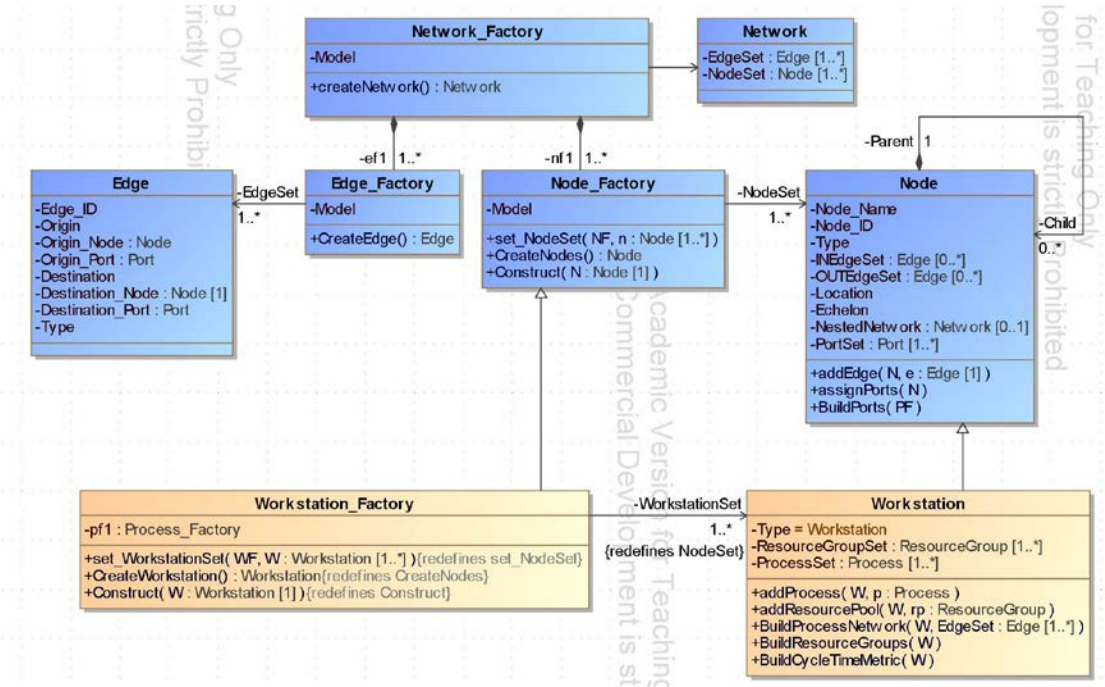

Figure 4: UML Schema of M2M Method Based on Creational Patterns.

With the type of product known and a method to create it in the simulation, the last component to the generative process is a method to modify the internal machinery of each product. The builder pattern is well-suited to this task by creating a Director object and configuring it with a Builder object, which makes systematic modifications to the product. This is useful since we created our simulation from stock objects cloned from a model library, and now we need to make several small modifications to the blocks. We implement the builder pattern through two basic methods. Figure 5 illustrates the first method, where after creating the stock Workstation object the *Workstation Factory* switches to the Director role and instructs its builder object, the *Workstation* object, to make modifications, such as building the resource groups or the cycle time metric, to the corresponding SimEvents block. The key aspect of this pattern is that the Builder modifies the product step by step under the Director's control; moreover, it does this through a uniform interface, the "Construct" operation, which improves modularity by encapsulating the way a complex object is constructed and represented.

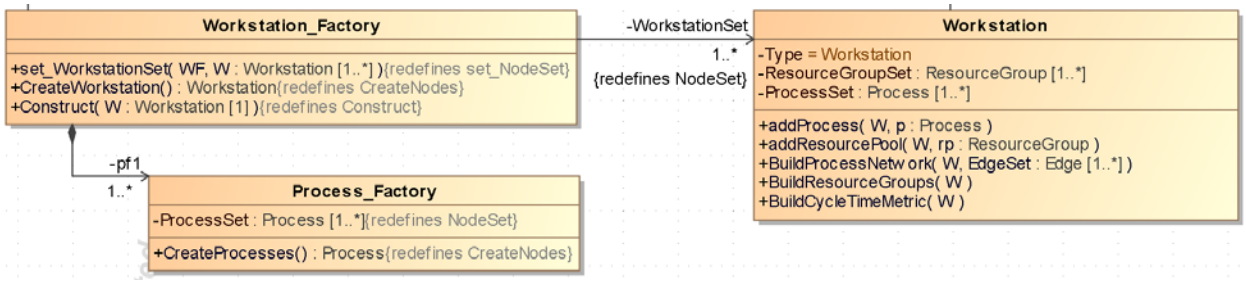

Figure 5: Example for creating and modifying workstation classes.

The development of a new model transformation method had several common, but important requirements. First, the method needs to be reusable to generate models in any subdomain within the

DELS domain, e.g. supply chain logistics, warehousing, manufacturing, etc. Second, the method needs to be extensible by being capable of handling additional refinements to an existing domain. The method achieves a high degree of reusability by relying on the network flow abstraction underlying each of the target domains. Model transformation extensibility is achieved with variation points and the management of subsystem variants as well as specialized builder classes to achieve precise refinements to the internal machinery of a subsystem.

5 TWO EXAMPLE USE CASES: DISTRIBUTION SUPPLY CHAIN & WORKSTATION CONFIGURATION

By basing the generative approach on the abstract network definition, we can move seamlessly across multiple domains, perhaps representing different levels of abstraction or detail. In this section, we provide two use cases for deploying this generative methodology. Each is intended to illustrate a different, but broad, set of use cases that are specifically enabled by this tool chain. By enabled we mean to discuss what attributes and extensions of the use cases make the generative methodology useful. As a broad statement, we intend to illustrate use cases where we make structural changes to the simulation rather than simply exploring a parameterized trade space.

The first use case is a two-echelon spare parts distribution supply chain (Figure 6) where the supply chain is structured as a small set of centralized distribution centers (1 in this case) that source directly from suppliers and distribute inventory to forward depots (usually geographically dispersed) which serve a set of customers in their customer bases. In this use case, the typically set of analysis questions focus on optimal inventory stocking levels that minimize total cost while attaining a contractual service level.

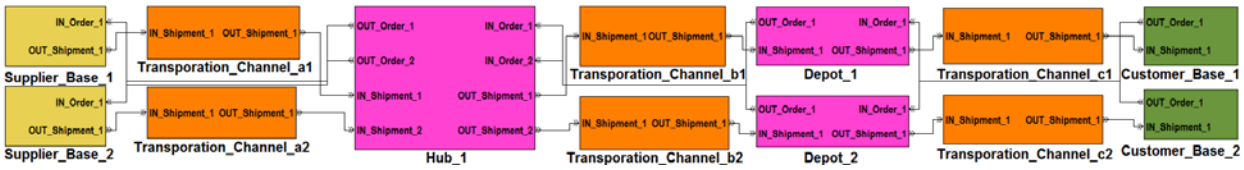

Figure 6: Distribution supply chain model created in SimEvents.

However, several additional analysis questions can be examined effectively only by modifying the structure of the simulation. 1) A facility location problem introduces questions about the number and location of depots and assignment of customers to each depot. 2) Transportation related extensions include the possibility of additional expedite channels running directly from supplier to depot or hub to customers or consolidating multiple direct channels into a milk-run type channel. 3) Examining the resiliency and reliability of the network by removing channels or reassigning customers to depots in response to an event. Each of these use cases requires making modifications to the existing simulation document or generating a new network structure rather than parameterizing the base model.

The second use case focuses on allocating the steps of a manufacturing process plan to workstations which execute the operations (Figure 7). This use case has several interesting implications. From a modeling paradigm perspective, it is difficult or impossible to separate concerns within a discrete event simulation environment. For example, we may want to generate a simulation of the process plan (transformed from an activity diagram) separate from or side-by-side with the workstation configuration. This would allow the system designer to perform preliminary analyses on the process network prior to allocating each process to a workstation.

The second implication is closely related to the assembly line balancing problem, where we try to allocate processes to workstations to optimize a set of metrics while adhering to precedent and capability constraints. We can modify the structure of this network by adding additional workstations in parallel, splitting a workstation into two in series, or through the reallocation of processes. Each of these modifications, which are representative local search neighborhoods in a broader optimization meta-

heuristics, generally requires you to regenerate the entire simulation or significant portions of the network.

These two use cases demonstrate that this methodology is reusable across several domains, extends the capabilities of this particular simulation language, and provides new capabilities for constructing and executing analysis use cases.

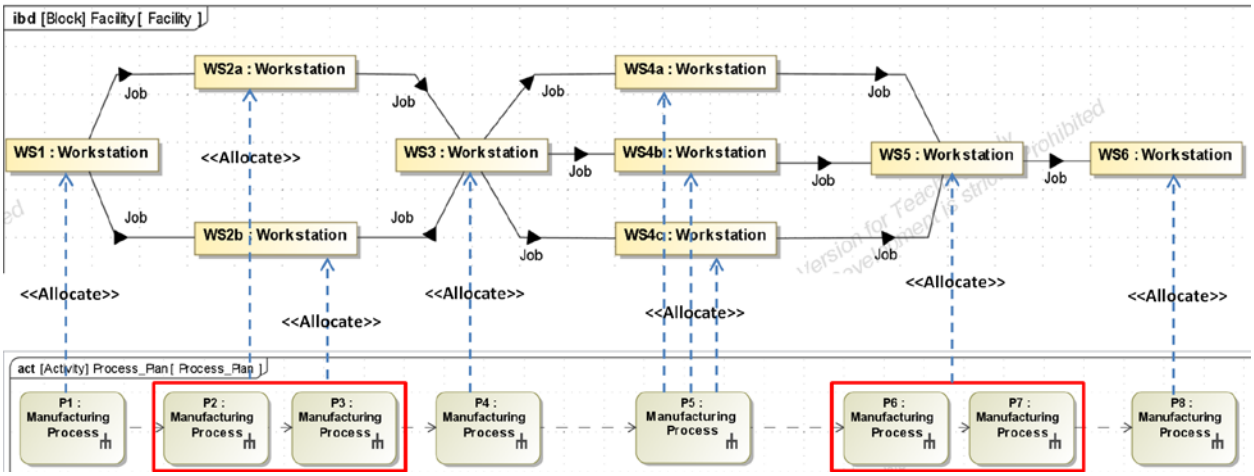

Figure 7: Workstation configuration model captured in SysML.

6 CONCLUSION AND FUTURE WORK

The goal of this research is a methodology that connects a conceptual model for discrete event logistics systems (DELS) to desired analysis tools, such as discrete event simulation using SimEvents. This methodology relies heavily on object oriented creational patterns and a network abstraction to construct an extensible and reusable transformation for the target domain. A future research goal is to develop a conceptual model for DELS that is more abstract, which would allow a greater degree of simplicity and reusability of the M2M transformations. Two additional considerations include the applicability and usefulness to other analyses and the process of extending the methodology to accommodate those analyses.

The applicability and usefulness of the proposed methodology is scoped by three qualities of the target use case. First, can the use case be abstracted to a network flow problem? For many analyses applicable to the DELS domain, this isn't a particularly restrictive requirement. Second, generative methodologies are most useful when the structure of the network is changing at each iteration of an analysis. Use cases that can be explored through parameterization tend to be less effective aside from cases when the model is prohibitively large to build by hand. Third, the analysis model language must support generation through an object oriented API. In the case of SimEvents, we can manipulate the model document from the MATLAB command line.

One of the major strengths of basing the generative methods on the abstract factory pattern is that we are abstracting the object creation process itself. Earlier we exploited this trait by creating a variety of concrete SimEvents products that were derived from the same abstract node class. However, we can also exploit it by porting the entire process itself to another platform by replacing the entire set of concrete factories with a new set that generates different analysis all together. One example would be to generate the schema in VBA, and then use the DSL for Arena to generate Arena process blocks. We can also extend the intermediate model in MATLAB to support the generation of optimization models through the IBM's CPLEX® API. This use case is particularly compelling because it demonstrates that our method can provide access to multiple types of analysis models, not only simulation, while preserving the integrity of the transformation model itself.

REFERENCES

Batarseh, O., and L. F. McGinnis. 2012a. "SysML to Discrete-Event Simulation to Analyze Electronic Assembly Systems." In *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation*, Article 48.

Batarseh, O. and L. F. McGinnis. 2012b. "System Modeling in SysML and System Analysis in Arena." In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, 2924-2935. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Biswas, S., and Y. Narahari. 2004. "Object Oriented Modeling and Decision Support for Supply Chains." *European Journal of Operational Research* 153 (3): 704–26.

Chatfield, D. C., T. P. Harrison, and J. C. Hayya. 2006. "SISCO: An Object-Oriented Supply Chain Simulation System." *Decision Support Systems* 42 (1): 422–434.

Coad, P., and E. Yourdon. 1991. *Object-Oriented Design*. Englewood Cliffs, NJ: Yourdon Press.

Cope, D., M. S. Fayez, M. Mollaghasemi, and A. Kaylani. 2007. "Supply Chain Simulation Modeling Made Easy: An Innovative Approach." In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 1887–1896. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Czarnecki, K., and S. Helsen. 2006. "Feature-Based Survey of Model Transformation Approaches." *IBM Systems Journal* 45 (3): 621–645.

Estefan, J. A. 2007. "Survey of Model-Based Systems Engineering (MBSE) Methodologies." Technical Data. Pasadena, California: Jet Propulsion Laboratory.

Friedenthal, S., A. Moore, and R. Steiner. 2009. *A Practical Guide to SysML: The Systems Modeling Language*. Elsevier.

Gamma, E., R. Helm, R. Johnson, and J. Vlissides. 1995. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Huang, C.-C. 2011. "Discrete Event System Modeling Using Sysml and Model Transformation." Ph.D. thesis. Atlanta, GA: Georgia Institute of Technology. http://smartech.gatech.edu/handle/1853/45830.

Huang, E., R. Ramamurthy, and L. F McGinnis. 2007. "System and Simulation Modeling Using SysML." In *Proceedings of the 2007 Winter Simulation Conference*, edited by S. G. Henderson, B. Biller, M.-H. Hsieh, J. Shortle, J. D. Tew, and R. R. Barton, 796–803. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Johnson, T. A., C. J. J. Paredis, R. Burkhart, and J. M. Jobe. 2007. "Modeling Continuous System Dynamics in SysML." In *Proceedings of the 2007 ASME International Mechanical Engineering Congress*.

McGinnis, L., E. Huang, K. S. Kwon, and V. Ustun. 2011. "Ontologies and Simulation: A Practical Approach." *Journal of Simulation* 5 (3): 190–201.

McGinnis, L., and V. Ustun. 2009. "A Simple Example of SysML-Driven Simulation." In *Proceedings of the 2009 Winter Simulation Conference*, edited by M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, 1703–1710. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mönch, L., P. Lendermann, L. F. McGinnis, and A. Schirrmann. 2011. "A Survey of Challenges in Modelling and Decision-Making for Discrete Event Logistics Systems." *Computers in Industry* 62 (6): 557–567.

OMG MDA. 2003. "OMG Model Driven Architecture (OMG MDA) Version 1.0.1". Object Management Group. http://www.omg.org/mda/

OMG MOFM2T. 2008. "OMG MOF Model to Text Transformation Language (OMG MOFM2T) Version 1.0". Object Management Group. http://www.omg.org/spec/MOFM2T/1.0

OMG QVT. 2011. "OMG MOF 2.0 Query/View/Transformation Specification (OMG QVT) Version 1.1". Object Management Group. http://www.omg.org/spec/QVT/1.1/

OMG SysML. 2012. "OMG Systems Modeling Language (OMG SysML) Version 1.3". Object Management Group. http://www.omg.org/spec/SysML/1.3/

Peak, R. S., R. M. Burkhart, S. A. Friedenthal, M. W. Wilson, M. Bajaj, and I. Kim. 2007. "Simulation-Based Design Using SysML—part 1: A Parametrics Primer." In *Proceedings of the 2007 INCOSE International Symposium*.

SCOR. 2012. "SCOR: The Supply Chain Reference Model Version 11.0". Supply Chain Council, Inc. www.supply-chain.org.

Shah, A. A. 2010. "Combining Mathematical Programming and SysML for Component Sizing as Applied to Hydraulic Systems". M.S. thesis. Atlanta, GA: Georgia Institute of Technology.

Son, Y. J., A. T. Jones, and R. A. Wysk. 2000. "Automatic Generation of Simulation Models from Neutral Libraries: An Example." In *Proceedings of the 2000 Winter Simulation Conference*, edited by J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 2:1558–1567. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sprock, T., and L. F. McGinnis. 2014. "Towards Automated Access to Supply Chain Analyses." In *Proceedings of the 2014 POMS Annual Conference*.

Thiers, G., and L. F. McGinnis. 2011. "Logistics Systems Modeling and Simulation." In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R.R. Creasey, J. Himmelspach, K.P. White, and M. Fu, 1531–1341. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Wu, D., L. L. Zhang, R. J. Jiao, and R. F. Lu. 2011. "SysML-Based Design Chain Information Modeling for Variety Management in Production Reconfiguration." *Journal of Intelligent Manufacturing*, 24(3): 575–596.

AUTHOR BIOGRAPHIES

TIMOTHY SPROCK is a graduate student in the Stewart School of Industrial and Systems Engineering at Georgia Tech. Before returning to school he was employed by The Boeing Company in Philadelphia, PA and received a bachelor's degree in applied physics and economics from Rensselaer Polytechnic Institute, Troy, NY. His email address is tsprock3@gatech.edu.

LEON F. MCGINNIS is Professor Emeritus in the Stewart School of Industrial and Systems Engineering at Georgia Tech and founder of the Keck Virtual Factory Lab. He is internationally known for his leadership in the material handling research community and his research in the area of discrete event logistics systems. A frequent speaker at international conferences, he has received several awards from professional societies for his innovative research, including the David F. Baker Award from IIE, the Reed-Apple Award from the Material Handling Education Foundation, and the Material Handling Innovation Pioneer award from Material Handling Management Magazine. He is author or editor of seven books and more than 110 technical publications. At Georgia Tech, Professor McGinnis has held leadership positions in a number of industry-focused centers and multi-disciplinary programs, including the Material Handling Research Center, the Computer Integrated Manufacturing Systems Program, the Sustainable Design and Manufacturing Program, and the Tennenbaum Institute for Enterprise Transformation. His current research explores the use of formal systems modeling methods to support systems engineering of discrete event logistics systems. Professor McGinnis is a Fellow of the Institute of Industrial Engineering. His e-mail address is leon.mcginnis@gatech.edu.