XII. INTEGRATING ANALYSIS INTO A WAREHOUSE DESIGN WORKFLOW

Leon F. McGinnis
Timothy Sprock

The Georgia Institute of Technology
School of Industrial & Systems Engineering
Atlanta, GA 30332-0205 USA

Abstract

Supply chain analyses, including those related to material handling systems, are typically purpose-built to answer specific questions and therefore have many different implementations depending on the question, the instance data, and the solver. The purpose-built nature of these models makes it difficult to integrate them into an iterative design workflow. Despite the myriad analysis implementations, the fundamental structure of these systems and their problem domain remains unchanged, suggesting that perhaps analyses could be automatically generated on demand, given an appropriate specification of the particular system to be analyzed. We apply model-based systems engineering (MBSE) methodologies to explore this possibility in the context of functional warehouse design.

1 Introduction

The domain of industrial engineering is broad, but a large segment of that domain can be described as *discrete event logistics systems* or DELS. These are systems consisting of a network of resources through which units of handling flow and are transformed by those resources performing processes with distinct start and end times. The resources might be machine tools and the units of handling might be blanks transformed to machined parts; the resources might be transport devices and warehouses, and the units of handling might be pallets of goods transformed from one location to another; the resources might be health care providers and the units of handling might be people transformed from "sick" to "healthy".

Industrial engineering research offers many tools and methods for analyzing DELS, but widely accepted generic design methods—and in particular, tool chains—for designing DELS have remained elusive. Some ideas for modeling and designing warehouses and material handling systems are presented in [1] and [2]. [1] explores the fundamental changes that model driven architecture (MDA) methodologies would bring to the engineering of material handling systems and discrete event logistics systems (DELS) in general.

This paper addresses the problem of translating design methods into tool chains to support warehouse design decision making. Specifically, we focus on applying current research on

formal domain modeling for the DELS and creating model-to-model (M2M) transformations to provide designers with automated access to design-specific analysis tools. Our goal is to show that analysis methods can be made computational within a generic tool chain to support design decision making. [2] proposes a methodology for creating a warehouse design by focusing on formalizing the semantics of warehouse requirements, functional design, and embodiment design. The formalism presented there is intended to bridge the gap between process oriented research on warehouse design and mathematical approaches. This paper will start with that formalism, and modify and extend it to allow automated analysis model generation, an essential requirement for creating a true warehouse design tool chain.

The remainder of this paper is organized as follows: section 2 reviews the design process for material handling systems; section 3 discusses formal domain modeling, and in particular introduces the functional requirements network (FRN); section 4 provides useful analysis of FRNs to support the design process with supporting use cases; and section 6 concludes with a discussion of future research opportunities.

2 Design Process for Material Handling Systems

Design is the specification of an artifact intended to accomplish specific goals, using a primitive set of components and subject to constraints [3]. Conceptually a design methodology should provide a structure to the design space, support generating alternative designs within that space, provide methods to effectively search the space, and enable the analysis of alternative designs with regard to criteria and constraints. Because there are few constraints imposed upon the design and configuration of a specific warehouse, the design space tends to be rather large and unstructured and therefore difficult to search. This difficulty results in ad-hoc approaches to the design process, which rely on the tacit knowledge of experts and reuse of existing, tested designs that have been deemed successful.

In recent years, there has been some progress toward the goal of developing computer aided warehouse design methods and tools. For example, there is now a large body of research on analytic models to support specific design decisions (see, e.g., [4] for a thorough literature review). There have been a number of investigations of the warehouse design process itself, including empirical studies [5]–[7], as well as attempts to formalize the process [8]–[14].

To bridge the gap between process-oriented research on warehouse design and the mathematically oriented approaches, McGinnis [2] proposes an object-oriented and axiomatic warehouse design methodology. In this proposed methodology, the design process is structured as two distinct phases: *functional design*, which specifies *what* must be done in the warehouse—its *functions*—to fulfill its mission, and *embodiment design*, which specifies *how* the functions will be implemented—the *systems*—and what resources will be required.

The design process consists of first understanding the context of the system being designed, then performing functional requirements analysis, functional design, and finally embodiment design. The context of the warehouse is the interaction of the warehouse with its external environment, i.e., the flows of goods in (INFlow) and out (OUTFlow) of the warehouse. The second step, functional requirements analysis, characterizes these flows to understand the requirements for consuming the INFlow and producing the OUTFlow. This results in a requirements profile, i.e., the capabilities and their respective capacities that are required for the warehouse to perform its intended function.

Functional design is the “essence of the warehouse design problem”. This step in the design process specifies the transformation processes that take place within the warehouse at a functional level. This contrasts with traditional design methodologies that attempt to specify the

resources and policies that should be utilized to accomplish a specific set of transformations. Finally, the embodiment design specifies how the functional design will actually be executed by allocating those functions to systems and configuring those systems with the necessary resources and policies to execute their respective functions.

Ultimately this design process focuses on formally identifying the capability requirements of the warehouse, establishing the essential functions the warehouse must perform, and then embodying these functions in a system specification, which includes configuring a specific set of resources with specific behaviors to provide specific capabilities. A formal specification of the system design provides an essential foundation for integrating analysis tools throughout the design process.

3 Modeling Material Handling Systems

McGinnis [1] explores the fundamental changes that model driven architecture (MDA) methodologies would bring to the engineering of material handling systems, and discrete event logistics systems (DELS) in general. This section will discuss recent advances in applying MDA approaches to modeling and analyzing DELS including formal domain modeling and model transformation methodologies.

3.1 Formal Domain Modeling with SysML

Formal domain modeling has been used in software engineering for designing and developing systems, and relies on domain specific languages (DSLs) and visualization to enable a systems engineer to focus on abstract modeling of the target domain [15]. In 2006, OMG published the initial standard for the Systems Modeling Language, or SysML, [16] to create a general-purpose modeling language for systems engineering applications. Using SysML, it is possible to capture the structure, behavior, dynamics, and requirements of a system; furthermore, all four of these modeling aspects are derived from the same underlying specification of the system. 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 [17]–[22].

To tailor the modeling language to a specific domain, SysML provides two mechanisms for creating DSLs: model libraries and profiling [23]. Prior research has established the application of this methodology for creating a DSL in SysML, describing the system of interest using the DSL, and finally transforming a description of a system model specified using the DSL into a target analysis language, such as discrete event simulation [24]–[27].

As a complement to the DSL, a reference architecture provides a template for stakeholders to reuse in creating other system architectures in the same domain. Cloutier et al. [28] suggest that patterns are one means to document reference architectures, since patterns are well recognized for their capability to make implicit knowledge explicit. A reference architecture for material handling systems would provide formally defined semantics for specifying the design of a particular system and a platform on which to construct automated access to analysis tools.

3.2 Automated Generation of DES

Due to the inherent complexity of MHSs, analytic models are often intractable and require significant assumptions to model the observed “real world” behavior. In the design context, discrete event simulation is an attractive option for evaluating the expected behavior and performance of the system of interest. However, there are significant hurdles to building, running, and analyzing simulation models due to the range of simulation analysis tool

capabilities and characteristics [29]. As a result, simulation is usually done only once, for the final design selected, rather than done routinely in the course of searching the design space.

The thesis explored in this paper is that by adapting the MDA approach from software engineering, augmented by the use of patterns, we can make simulation more readily accessible to the designer throughout the design workflow. Naturally, this requires providing the designer with a tool chain that has the functionality required to specify the functional and embodiment designs.

MDA advocates the creation of a platform-independent model of software at a higher level of abstraction than the target code or hardware, which can be transformed as needed into platform-specific (particular programming language and hardware) and executable models; adapting MDA would separate the development of a DELS design model from its analysis using a specific simulation software package; the analysis implementation would be constructed automatically by a reusable model to model (M2M) transformation. Current M2M methods, such as QVT [30] or MOFM2T [31], can execute transformations from SysML to an XML or structured text specification of the target analysis, e.g., AnyLogic™ [32] or SimEvents™ [33] respectively. However, the difficulties that arise in applying current M2M methodologies for code generation to generating discrete event simulation, led to our development of model generation techniques based on creational software patterns [34]. This transformation methodology utilizes object oriented programming methods coupled with a network abstraction to automatically create discrete event simulations for the DELS domain. A critical step in implementing these ideas is the abstract and formal specification of the functional requirements network.

3.3 Functional Requirements Network

The primary function executed by a warehouse is to transform an inbound stream of replenishment shipments, its INFlow, into a stream of outbound fulfillment shipments, its OUTFlow. However, the exact transformation process depends on the nature of the inbound and outbound streams and thus lacks a single universal specification. Therefore the functional design stage focuses on specifying the structure of this transformation process. McGinnis [2] and Govindaraj et al. [35] have proposed capturing the resulting structure as a functional requirements network (FRN), which consists of warehouse functions captured as logical nodes and flows captured as the edges between those functions. In this section, a formal definition of the FRN is captured as a reference architecture, which consists of three components specified using SysML: 1) a profile for token flow networks, a subclass of the classical network definition; 2) a definition of the node and edge components of the functional requirements network; and 3) a pattern for the specification of flow throughout the FRN.

The network abstraction is common throughout the DELS domain and it provides access to a well-understood analysis framework and the associated tools. The token flow network (TFN) [36] extends the basic network definition to provide formal semantics for flow and behavior within a network. The TFN provides flow network semantics for specifying the FRN (Figure 1). In addition to extending the definition of nodes and edges to flow nodes and flow edges, the TFN introduces the concept of a token as an abstract semantic for any entity that can move through the network. An example of the token semantics is that a *SKU* sitting in stock is required by an *orderline* corresponding to an *order*. Each of these levels of aggregation can be thought of as an independent token flowing through the network depending level of granularity desired during the design process. The flownode defines what is consumed and produced by the transformation process; i.e. a mixed pallet assembly node will consume *n* cartons and produce *m* mixed pallets.

The flow edge restricts what types of tokens flow across it; e.g., the flow from a **RCVpallet** node to **palletSTORE** node will be restricted to tokens for which the UoH is a pallet.

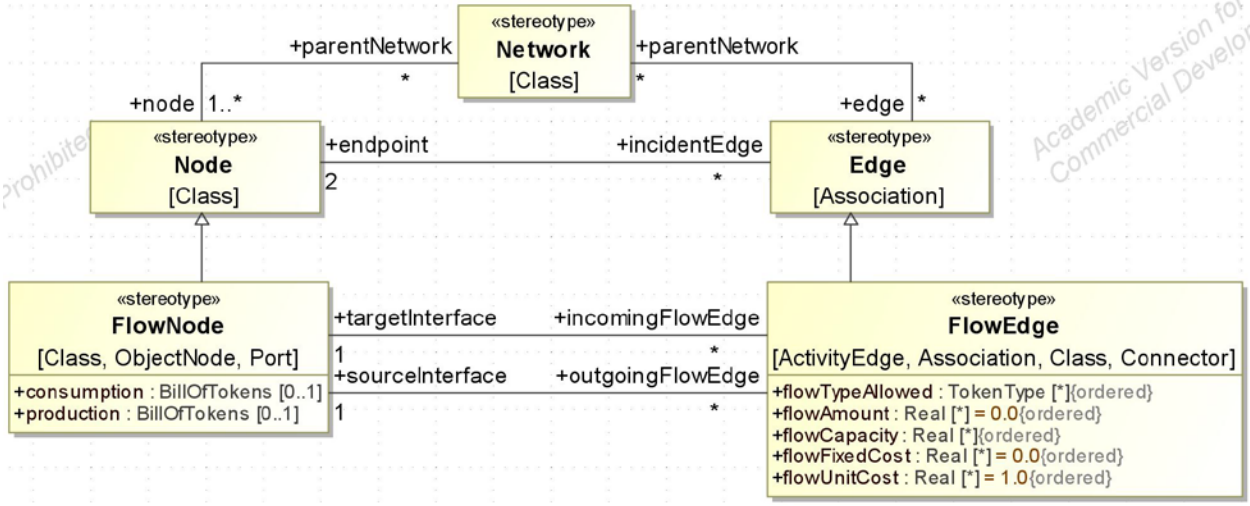

Figure 1 Subset of Token Flow Network Definition

"The functions of a warehouse are abstract processes which make abstract changes to an abstract flow" [2]. Figure 2 consolidates the process and flow definitions presented in McGinnis [2] with the TFN profile applied to provide the network structure. However, there are two extensions: 1) *family* is the superclass of order family and SKU family, which are specified during the functional requirements analysis process, and provides an additional level of aggregation, and 2) *Flow* can be subclasses into *INFlow* and *OUTFlow*, with their origin and destination redefined to supplier and customer nodes respectively. This unifies all the FRN semantics specified in the original paper.

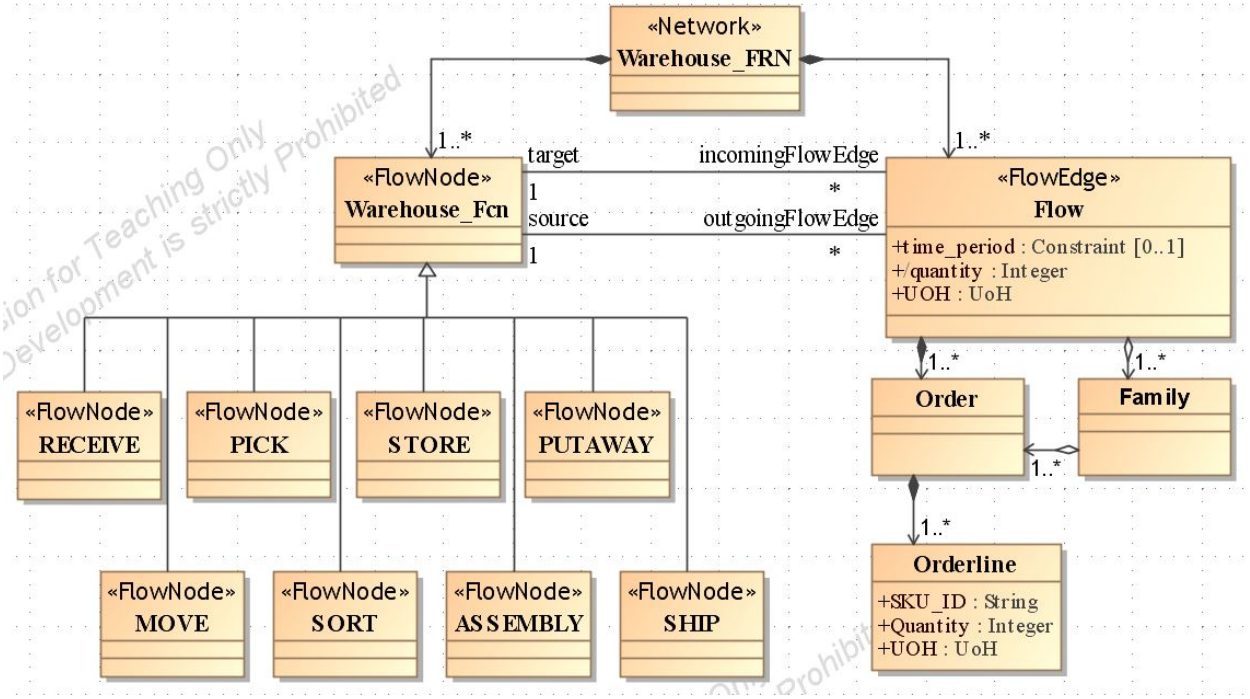

Figure 2 Functional Requirements Network Definition

The last component of the reference architecture for the FRN is an implementation pattern for creating a FRN (Figure 3). This pattern captures several principles that are important to the specification of the FRN and that support the warehouse design process:

- Each OUTFlow and INFlow are associated with a single Order Family and SKU family respectively. Also, each OUTFlow must have a corresponding (order) Assembly function and each INFlow must have a corresponding STORE function. (Axioms 5&6, [2])
- It provides a template for creating the flows that are incident to each function.

Throughout the design process, the FRN is constantly altered and refined by adding, removing, and combining functional nodes. The pattern suggests what corresponding changes need to be made to the set of edges when there is a modification made to the nodes.

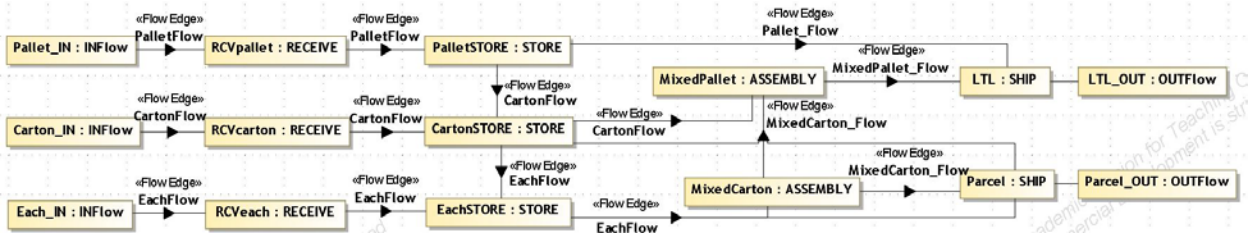

Figure 3 An Implementation Pattern of the FRN

The following example illustrates the usage of this pattern. Suppose a new SKU family is introduced thereby creating a new INFlow, then automatically a new STORE node is required, as well as flow edges from RECEIVE to STORE. In addition, the new STORE node requires the set of flows incident to the STORE node; e.g., replicating the **CartonSTORE** node requires replicating the CartonFlow edges to **Mixed Pallet ASSEMBLY** and **Parcel SHIP**.

The reference architecture supports the system design process by providing a formal structure to the FRN and patterns to accelerate the creation process and reduce mistakes. While only a small subset of the TFN is used to specify the FRN, the complete specification is extensive and supports the embodiment design process, which transitions a FRN into a system specification. Moreover, the formal network abstraction is fundamental to providing automated access to analyses and is the cornerstone of methodology for automatically generating discrete event simulations presented in section 3.2.

4 Analysis of Functional Requirements Networks

A formal definition of the FRN is a useful tool because it can be used to develop methods for executing the functional design process and integrating supporting analyses. It should be clear that the ability to capture the functional design is not the same as having the ability to develop the functional design. In this section we address the following questions:

- If automated access to analyses tools was available on demand, what analyses would be useful in the design process?
- How would you incorporate the feedback from these analyses into the design process?

4.1 Design and Implementation of Analysis Tools

In the early stages of the design process, the capability and capacity requirements of the warehouse can only be described in terms of the flows. Capability is related to the physical requirements for handling and storing goods and orders, which are derived from the characteristics defined by the functional requirements analysis. From figure 3, if pallets flow into the warehouse through **RCVpallet**, then that function must have the capability to lift and move

pallets. Capacity, however, is derived from the flow rate data. Therefore, a fundamental analysis for the FRN would be to determine the capacity requirement of each functional node by computing the INFlows and OUTFlows.

However, propagating all of the flows through the system is not a trivial task for two intertwined reasons: 1) the flow of material into the warehouse is a push system, whereas the flow of material out is a pull system driven by customer orders, which then suggests that 2), as an example, it is more natural to specify that 75% of cartons flowing into **MixedPallet Assembly** are picked from pallets in **palletSTORE** and 25% are mixed cartons from **MixedCarton Assembly** (Figure 4). Because the FRN is specified as a network, the transformation method described in section 3.2 can automatically generate a discrete event simulation that is capable of answering the analysis question. Figure 4 and Table 1 demonstrate the SimEvents [33] DES generated for a specific use case and the output results of the simulation.

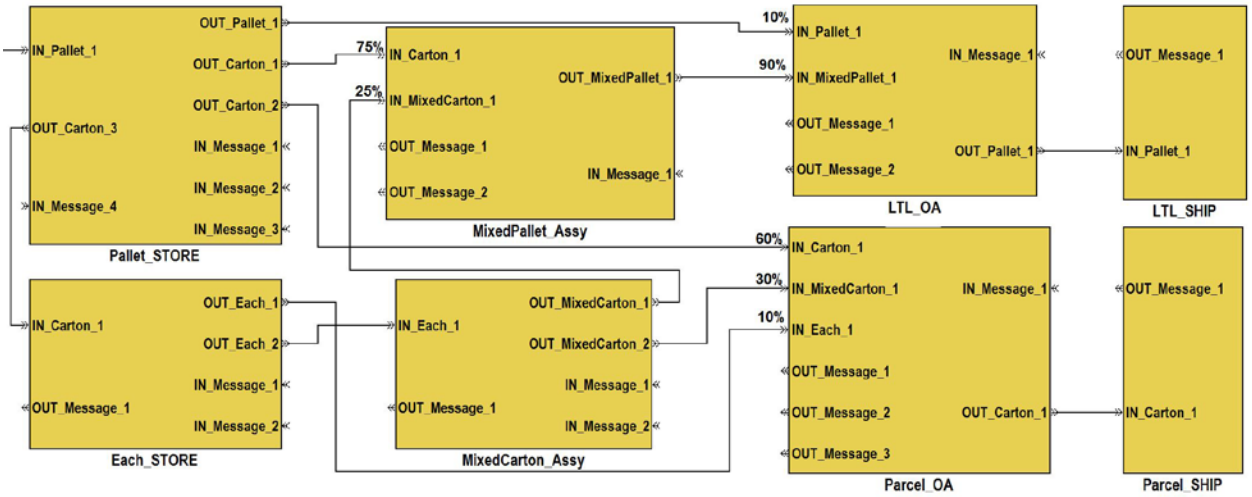

Figure 4 Discrete Event Simulation of a FRN generated in SimEvents
(Message Flows Omitted for Clarity)

The specification of the use case is as follows: the system sees an OUTFlow of 1000 pallets / unit time via LTL shipping and 5000 cartons / unit time via parcel ship. We specify the flows into each functional node in the natural way described above; i.e. LTL order assembly sees an OUTFlow of 1000 pallets and preliminary analysis indicates that 10% of those pallets will be directly from pallet store and 90% will be mixed pallets from mixed pallet assembly. The results of the simulation, Table 1, are merely estimates because they simplify the pallet to carton to each consumption ratios by ignoring SKU composition, but nevertheless demonstrate the capability of the analysis tool.

Table 1 Output Results of the DES

	PalletSTORE	EachSTORE	MP_Assy	MC_Assy	LTL_OA	Parcel_OA
Pallets	1775/100	---	---	---	100/1000	---
MixedPallet	---	---	0/900	---	900/0	---
Carton	0/38,525	1775/0	33750/0	---	----	3000/5000
MixedCarton	---	---	225/0	0/1,725	---	1500/0
Each	---	0/17,750	---	17,250/0	---	500/0

INFlow/OUTFlow of each UoH (Row Header) to/from each flow node (Column Header)

Why generate a discrete event simulation to answer a rather simplistic analysis question? By designing different but interchangeable analysis components to answer different questions, answering different analysis questions is as simple as switching the analysis component types that are generated within the DES. With a full suite of analysis components, the system engineer will have the ability to automatically answer a wide range of questions about the current system design and incorporate that feedback into the design process.

4.2 Incorporating Analysis Results into the Design Process

With relevant analyses accessible throughout the design process, the methodology itself can be refined to incorporate that knowledge. Iterating the design process—functional requirements analysis, then functional design, then embodiment design—allows incorporating new analysis results into incremental improvements to the design. This section presents examples of integrating analyses to improve the design process

Functional requirements analysis focuses on discovering the important traits for creating SKU and order families. It, however, does not provide an exact method for specifying the families that lead to the optimal FRN. While some families, such as frozen items or pallet flows, are obvious, the refinement of the family definitions is still considered an art. This presents an opportunity to incorporate statistical methods such as factor analysis to iteratively refine the family definitions to improve the system design. Whereas technology selection and configuration is performed during embodiment design, often simple analyses performed during functional design, such as estimating the necessary cycle time to achieve a desired throughput, can provide insight into the feasibility of a particular functional design even before beginning the embodiment design process.

Access to simple, yet effective, analysis tools throughout each stage of the design process can prevent costly rework. For example, suppose the design is initialized as follows: the system receives pallets and stores them in a pallet store, and then cartons are picked from the pallet store and assembled into mixed pallets to be shipped out by LTL. We may arrive at the embodiment design and realize that the technology required to fulfill the requirements of the pick from storage function is infeasible or too expensive; rather than make small ad-hoc changes to the design, we would like to iterate and revise either the functional requirements specification or the functional design. By returning to the functional requirements analysis, we determine that we can divide the Carton SKU family into Heavy Cartons, which should be on the bottom of every pallet; Fast Moving Cartons; and Slow Moving Cartons. The resulting FRN is depicted in Figure 5.

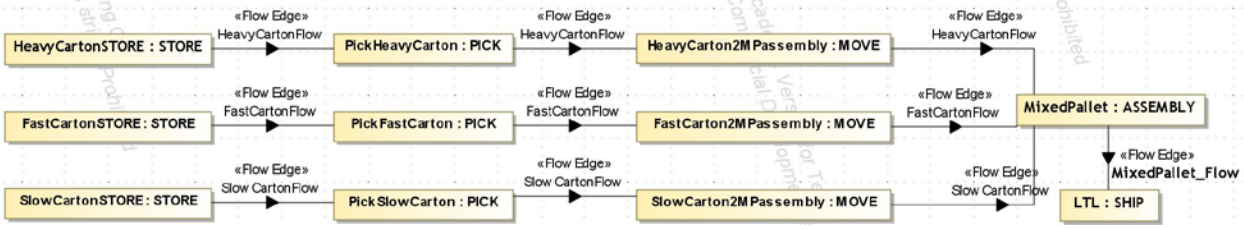

Figure 5 Incorporating Improvements to the SKU family specifications into the FRN

As each step of the design process is dependent on the decisions made in the previous step, it is often difficult to know whether a particular design decision, whether it is a specific portioning of families or allocation of functions to systems, will result in good system performance. Automated access to analyses provides a means to design in an iterative process by incorporating or developing for new information at each stage of the design process. At the beginning of this

process, very little is known about the system, so the analyses that can be performed are simple and possibly trivial. However, as the system design is refined and elaborated, the ability to generate a DES to answer analysis questions becomes more valuable.

4.3 Transitioning to the Embodiment Design Phase

In addition to generating different analysis components, the generation of DES supports a natural transition to the analysis of embodiment designs too. During this transition, the FRN is partitioned into groups of functions which are then allocated to subsystems. One example may be to group the **pallet STORE**, **pick-carton-from-pallet RETRIEVE,** and **mixedPallet ASSEMBLY** functions into a single subsystem. This subsystem can be embodied as follows: the picker uses a pallet jack to go through the pallet store to pick cartons and build a mixed pallet during his tour. However, it can also be embodied as a subsystem where an ASRS brings cartons from the storage area to the picker who then assembles a mixed pallet from a fixed position. Even without the final resource and policy configuration, these subsystem specifications can then be transformed into DESs to execute preliminary performance analyses and support the embodiment design process (Figure 6).

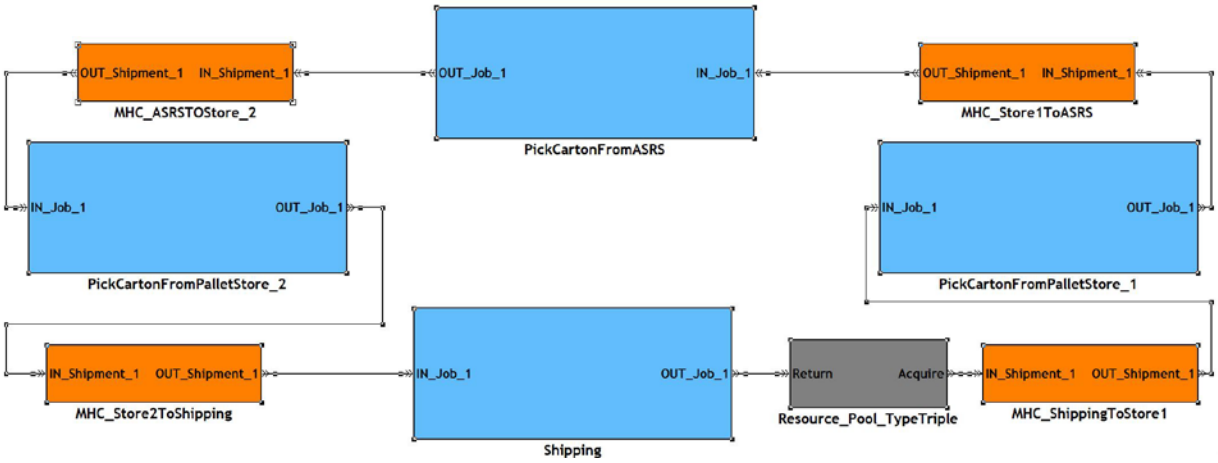

Figure 6 Discrete Event Simulation of Embodied System Design Generated in SimEvents

The allocation process provides a seamless transition from functional design to embodiment design, and integrates the embodiment design process into the iterative design process. With the ability to create and refine the embodiment design and generate simulations that reflect those designs, the system engineer can quickly iterate the design process.

Looking forward, the reference architecture presented above should be extended to define the embodied subsystem itself. The subsystem specification should include the processes performed, the resources used to execute those processes, as well as all the necessary policies and associated control mechanisms; this establishes the services that the subsystem will provide to the system.

5 Conclusion

The design of material handling systems, and discrete event logistics systems in general, would benefit from the implementation of a formal methodology to specify the system architecture and the integration of analyses throughout the design process. Several methodologies have been proposed to formalize the design process; however there still is a gap between the design and the analysis of material handling systems. This paper introduces a method to bridge between the design process and supporting analysis tools. The method has two components: the token flow network provides a formal structure to the functional requirements network, which is the output

of the functional design process; and an automated transformation based on the token flow network that generates simulation models to answer specific analysis questions about the functional design.

With analysis tools such as discrete event simulation accessible throughout the design process, this research demonstrates a method to integrating those tools into the design workflow itself. Even with simple analyses, such as flow rate analysis, the engineer can effectively integrate that information to improve the system design and can support an iterative design process.

Throughout this paper, we have been careful to avoid suggesting that this methodology will make design decisions and automate the design process. While many aspect of the design process remain an art, we contend that having instantaneous access to performance and behavioral information can improve the design process and the design itself. This suggests two closely related extensions for future research

One promising extension of this research would be to integrate the access to analytics presented here with the empirically-based design approach presented in [5]. The most appealing aspect to integrate would be their usage of matrix solution guides. Whereas their approach asks the designer to analyze key parameters and then look up the empirically optimal technology solution in a matrix, our methodology provides automated access to the required analysis.

Another possible path would be to incorporate design principles into the reference architecture as constraints, and then automate the usage of patterns and constraints from the reference architecture to specify and search the design space. Since we design our systems and analysis components for modularity, an intuitive extension would be to design a methodology that is intended to assemble these components into meaningful systems. These methods are used extensively in platform-based engineering and product line design, which are analogous to mass customization in the manufacturing world.

References

[1] L. McGinnis, “The Future of Modeling in Material Handling Systems,” in *11th International Material Handling Research Colloquium - 2010. Material Handling Industry of America*, 2010.

[2] L. McGinnis, “An object oriented and axiomatic theory of warehouse design,” in *12th International Material Handling Research Colloquium—2012. Material Handling Industry of America*, 2012.

[3] P. Ralph and Y. Wand, “A proposal for a formal definition of the design concept,” in *Design requirements engineering: A ten-year perspective*, Springer, 2009, pp. 103–136.

[4] J. Gu, M. Goetschalckx, and L. F. McGinnis, “Research on warehouse design and performance evaluation: A comprehensive review,” *European Journal of Operational Research*, vol. 203, no. 3, pp. 539–549, Jun. 2010.

[5] J. M. Apple, R. D. Meller, and J. A. White, “Empirically-based warehouse design: can academics accept such an approach,” in *11th International Material Handling Research Colloquium - 2010. Material Handling Industry of America*, 2010.

[6] D. A. Bodner, T. Govindaraj, K. N. Karathur, N. F. Zerangue, and L. F. McGinnis, “A process model and support tools for warehouse design,” in *Proceedings of the 2002 NSF design, service and manufacturing grantees and research conference*, 2002, pp. 1–8.

[7] M. Goetschalckx, T. Govindaraj, D. A. Bodner, L. F. McGinnis, G. P. Sharp, and K. Huang, “A review and development of a warehousing design methodology, normative model, and solution algorithms,” in *Proceedings of the 2001 Industrial Engineering Research Conference, Dallas, Texas*, 2001.

[8] B. Rouwenhorst, B. Reuter, V. Stockrahm, G. J. Van Houtum, R. J. Mantel, and W. H. M. Zijm, "Warehouse design and control: Framework and literature review," *European Journal of Operational Research*, vol. 122, no. 3, pp. 515–533, 2000.
[9] L. McGinnis, "Developing a Reference Model for Warehouse Specifications," presented at the IIE Research Conference, Houston, TX, 2004.
[10] L. McGinnis, M. Goetsch, and G. Sharp, "A Comprehensive Model of Traditional Warehouse Design," in *9th International Material Handling Research Colloquium—2006. Material Handling Industry of America*, 2006.
[11] L. McGinnis, "Facility Design Workflow Management," presented at the IERC, Vancouver, Canada, 2008.
[12] G. Sharp, M. Goetschalckx, and L. F. McGinnis, "A systematic warehouse design workflow: focus on critical decisions," in *10th International Material Handling Research Colloquium - 2008. Material Handling Industry of America*, 2008.
[13] M. Goetschalckx, L. F. McGinnis, and G. Sharp, "Modeling Foundations for Formal Warehouse Design," in *10th International Material Handling Research Colloquium - 2008. Material Handling Industry of America*, 2008.
[14] F. Friemann, M. Klennert, and L. F. McGinnis, "Model Based Systems Engineering in Warehouse Analysis and Design," presented at the IERC, Cancun, Mexico, 2010.
[15] S. A. Friedenthal, R. Griego, and M. Sampson, "INCOSE Model Based Systems Engineering (MBSE) Initiative," presented at the INCOSE 2007 Symposium, San Diego, 2007.
[16] OMG SysML 2012, "OMG Systems Modeling Language Version 1.3." Object Management Group, 2012.
[17] E. Huang, R. Ramamurthy, and L. F. McGinnis, "System and simulation modeling using SysML," in *Proceedings of the 39th conference on Winter simulation: 40 years! The best is yet to come*, 2007, pp. 796–803.
[18] T. A. Johnson, C. J. J. Paredis, R. Burkhart, and J. M. Jobe, "Modeling continuous system dynamics in SysML," in *2007 ASME International Mechanical Engineering Congress and Exposition*, 2007.
[19] R. S. Peak, R. M. Burkhart, S. A. Friedenthal, M. W. Wilson, M. Bajaj, and I. Kim, "Simulation-based design using SysML—part 1: a parametrics primer," in *INCOSE intl. symposium, San Diego*, 2007.
[20] A. A. Shah, "Combining Mathematical Programming and SysML for Component Sizing as Applied to Hydraulic Systems," Master of Science - Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, 2010.
[21] G. Thiers and L. McGinnis, "Logistics systems modeling and simulation," in *Proceedings of the 2011 Winter Simulation Conference (WSC)*, 2011, pp. 1531–1541.
[22] D. Wu, L. L. Zhang, R. J. Jiao, and R. F. Lu, "SysML-based design chain information modeling for variety management in production reconfiguration," *Journal of Intelligent Manufacturing*, pp. 1–22, 2011.
[23] B. Selic, "A systematic approach to domain-specific language design using UML," in *Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC'07. 10th IEEE International Symposium on*, 2007, pp. 2–9.
[24] O. Batarseh and L. F. McGinnis, "System modeling in SysML and system analysis in Arena," in *Proceedings of the 2012 Winter Simulation Conference*, 2012, p. 258.
[25] O. Batarseh and L. F. McGinnis, "SysML to discrete-event simulation to analyze electronic assembly systems," in *Proceedings of the 2012 Symposium on Theory of Modeling and Simulation-DEVS Integrative M&S Symposium*, 2012, p. 48.
[26] L. McGinnis and V. Ustun, "A simple example of SysML-driven simulation," in *Proceedings of the 2009 Winter Simulation Conference (WSC)*, 2009, pp. 1703–1710.
[27] L. McGinnis, E. Huang, K. S. Kwon, and V. Ustun, "Ontologies and simulation: a practical approach," *Journal of Simulation*, vol. 5, no. 3, pp. 190–201, 2011.
[28] R. Cloutier, G. Muller, D. Verma, R. Nilchiani, E. Hole, and M. Bone, "The Concept of Reference Architectures," *Systems Engineering*, vol. 13, no. 1, 2009.

[29] Y. J. Son, A. T. Jones, and R. A. Wysk, “Automatic generation of simulation models from neutral libraries: an example,” in *Proceedings of the 2000 Winter Simulation Conference*, 2000, vol. 2, pp. 1558–1567.
[30] OMG QVT, “OMG MOF 2.0 Query/View/Transformation Specification (OMG QVT) Version 1.1.” Object Management Group, Jan-2011.
[31] OMG MOFM2T, “OMG MOF Model to Text Transformation Language (OMG MOFM2T) Version 1.0.” Object Management Group, Jan-2008.
[32] *Anylogic*. The AnyLogic Company. http://www.anylogic.com/.
[33] *SimEvents*. Mathworks. http://www.mathworks.com/products/simevents/.
[34] T. Sprock and L. F. McGinnis, “Simulation Model Generation Using Software Design Patterns,” in *Proceedings of the 2014 Winter Simulation Conference*, Savannah, GA, 2014.
[35] T. Govindaraj, E. E. Blanco, D. A. Bodner, M. Goetschalckx, L. F. McGinnis, and G. P. Sharp, “Design of warehousing and distribution systems: an object model of facilities, functions and information,” in *2000 IEEE International Conference on Systems, Man, and Cybernetics*, 2000, vol. 2, pp. 1099–1104.
[36] G. Thiers, “A Model-Based Systems Engineering Methodology to Make Engineering Analysis of Discrete-Event Logistics Systems More Cost-Accessible,” Georgia Institute of Technology, Atlanta, GA, 2014.