TOWARD AN ENGINEERING DISCIPLINE OF WAREHOUSE DESIGN

Leon McGinnis, Professor Emeritus
Timothy Sprock, Post-Doctoral Fellow

Stewart School of Industrial and Systems Engineering
The Georgia Institute of Technology

Abstract

Warehouses today are complex dynamic engineered systems, incorporating automation, mechanization, equipment, fixtures, computers, networks, products and people, and they can support the flow of tens or hundreds of thousands of different items to enable fulfilling thousands or tens of thousands of orders daily. In that sense, they represent a design challenge that is not terribly different from the design of other complex dynamic engineered systems, such as a modern passenger airplane, an automobile, or a unique building. What is different is that the design of these other complex dynamic engineered systems typically follows some engineering design *discipline*. Here, we argue for the development of a corresponding engineering discipline of warehouse design.

1 Introduction

The Oxford Dictionaries define *discipline* as "A branch of knowledge, typically one studied in higher education," [1] which implies an organized and codified body of knowledge which can be preserved, transmitted and improved over time. Certainly, that characterizes the various engineering disciplines seen in today's universities. Merriam-Webster defines discipline as "a rule or system of rules governing conduct or activity," [2] and that is typical of the design processes taught in many engineering disciplines.

In fact, a discipline (and domain) specific engineering design methodology can be characterized by:

- a common way of thinking about the designed system, i.e., a common semantic framework;
- a common way of thinking about the design process itself, i.e., a common design workflow (although there may be more than one in a given engineering discipline); and
- a common set of tools and techniques used to perform the design process.

All three elements are essential in order to have a sustainable engineering design community in which knowledge can be shared and improved over time; and the "discipline" can be readily transmitted to new members of the community. The semantic framework enables both effective communication and decision support analysis modeling. A common workflow supports improvements in individual activities of the design process by providing an enabling infrastructure within which the improvements can be evaluated and deployed. In the same way, the use of a common set of tools and techniques means that best practices and innovations can be easily deployed throughout the design community.

To give but one example, in the design of mechanical systems, there is a common semantic model for representing individual mechanical components, *constructive solid geometry* (see, e.g., [18]), in which the representation of a given object is constructed from *primitive* objects, such as cuboids, spheres, rods, prisms, etc., using Boolean operations. There are other representation schemes as well, but CSG serves to make the point. One common workflow for the design of a particular system, e.g., an automatic transmission or an excavator, begins by identifying the purpose and the market, and then seeking designs that maximize value. For an excavator, the purpose is "digging" and the market is quite varied in terms of the price that can be charged for an excavator with particular capabilities in terms of reach, payload, dig cycle, and total cost of ownership. The design workflow starts by identifying the functions to be realized, identifying the architectural alternatives (number of links and type of power, e.g.), then configuring each alternative to maximize some estimate of value. The tools used in the design workflow might include physics-based simulation models, such as finite-element analysis (FEA), to evaluate the performance of system models where exact analytical results are unattainable. Due to the mature nature of these simulations, approximations of the system model can be integrated early in the design process with high-fidelity simulations being conducted on the final product; an excellent example is the use of crash simulations of complete automobile designs [8].

In these examples, one sees the elements of a discipline of engineering design. There are textbooks which are used in courses where students learn the semantics, learn to use computational tools based on these semantics, and learn a workflow that can be followed for any design problem within the prescribed domain.

One might reasonably ask "Why doesn't something similar exist to support warehouse design?" Over the past 30 years, a large body of knowledge has been created around analyzing warehouses, from performance estimation of various systems, to configuration of equipment, to the assessment of alternative control policies. See, e.g., the reviews in [3], [4], [6], [10], and [17]. Yet there remains no identifiable *engineering discipline* of warehouse design, and as Gu et al. [10] point out, there is very little research supporting the creation of such a discipline.

Warehouse design is arguably difficult, because there are so many degrees of freedom (how many ways are there to store pallets, e.g.) and so few "hard" constraints. In addition, there are multiple criteria driving warehouse design and poorly articulated "requirements". But difficulty alone cannot be the reason, since designing integrated circuits with billions of features, or designing a modern passenger jet also are difficult, yet supported by a robust engineering design discipline. What has driven the creation of other engineering design disciplines is need—when design simply cannot be done manually, then tools and associated methods are essential. In the past, warehouse design could be done by "experts" who accumulated many years of experience, and had an innate understanding of "how to do it," and their design decisions could be supported by relatively ad hoc spreadsheet calculations, and tested by hand-crafted simulation models. That is no longer adequate, especially in an era where innovative new approaches are being explored for integrating automation into the order fulfillment process.

Perhaps it is time to begin to develop, teach, and practice such a discipline. The purpose of this paper is to describe one possible way to begin the process of creating an engineering discipline of warehouse design. To be successful, the process must lead to:

1. a commonly used frame of reference and semantics for specifying a warehouse design, expressed in a language that is accessible to all members of the warehouse design discipline community;
2. a capable and complete warehouse design work flow (at least one, perhaps more), i.e., a clearly understood and implementable set of actions in a repeatable process for articulating the need and goals, and translating them into a warehouse specification; and
3. a commonly used library of "plug-and-play" computational tools for both analyzing particular aspects of a warehouse specification and optimizing configuration where possible, which implies a commonly used frame of reference and semantics for describing these computational tools.

The semantics, workflow, and computational tools must be in a form that can be distributed, used, tested, and enhanced by any member of the warehouse design discipline community. As these three elements emerge, they will provide the basis for "independent

collaboration," i.e., for one researcher's use of these elements and conforming to them, to produce results that can be used, improved and extended by any other researcher in the discipline. This is absolutely critical for creating useful computational approaches to supporting warehouse design.

The remainder of the paper is organized as follows. First, we briefly describe some initial results in developing semantics (Section 2), a design workflow (Section 3), and computational tools (Section 4). These are far from completely satisfying the requirements described above; rather they illustrate what is needed and represent "seeds" around which more complete solutions can evolve. Then, we suggest the fundamental challenges most in need of research and development (Section 5).

2 A Reference Model for Warehouse Design

The focus here is on warehouse design, but warehouses are part of a larger domain of systems which have been described as discrete event logistics systems, or DELS [12]. DELS are characterized by a **product** moving through a network of **resources**, where **processes** transform the product in some way. A warehouse can be thought of as two DELS working in tandem. The first transforms supplier shipments into goods ready to pick – the product -- through a set of processes that include—but are not limited to—unloading, staging, receiving, transporting, and putting into storage. The second transforms a customer order into a customer shipment—the product—through a series of processes, which include—but are not limited to—order assignment, retrieving, accumulating, transporting, packing, shipping, staging, and loading. The linkage between these two systems is the goods in storage available for retrieval, the product output from one system becomes a resource input to the downstream system.

The product, process, and resource pattern is used to define the physical aspects of a DELS, or its *plant*, and consequently, what the DELS is capable of doing, i.e., how it *may* behave, or what behaviors it can exhibit. An additional layer of control determines how it *will* behave in a given scenario. These definitions for both plant and control are captured in a reference model, which serves as a template for constructing future system models and a repository for capturing and documenting warehouse knowledge [5].

2.1 Plant – Product, Process, Resource, and Facility Semantics

For most DELS systems, the product, process, and resource definitions are intertwined (Figure 1). The PPRF pattern for DELS can be extended to create more concrete system models. In the warehousing context, the product, or order, families can be defined by the additional processes that they must go through, e.g. kitting, labeling, or other

customization. The warehouse reference model should define broad classes of processes, e.g. picking methods or process, and the associated required resources, e.g. goods-to-picker and picker-to-goods retrieval systems jointly define a process and resource.

On the other hand, stock or consumable resources (inventory to be picked) can be classified by, e.g. its unit of handling (UoH), environmental requirements (e.g. freezer, refrigerated, hazardous, etc.), or pick frequency. Finally, the resources are configured within the facility, which integrates decisions such as department sizing and layout as well as determining conveyor paths, etc.

The reference model also provides a platform to develop reference models for sub-classes of material handling or fulfillment systems, i.e. it may make sense to define new classes, or new model libraries, of products, processes, and resources for pharmaceutical distribution centers or break-bulk cross docks, for example.

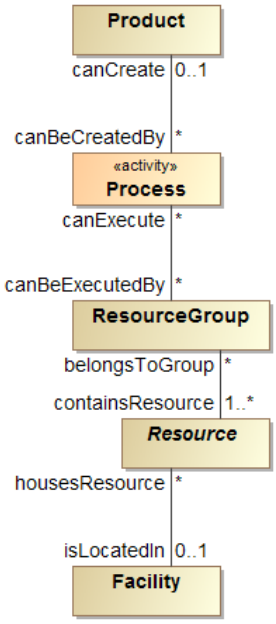

Figure 1 The product, process, resource, and facility pattern defines the behavior of a DELS

The abstraction, however, also provides a bridge to analysis models [23], and provides the basis for reusable infrastructure for generating analysis models and providing interoperability between the system definition and libraries of plug-and-play analysis models and their corresponding tools (Section 4).

2.2 Operational Control

In the context of warehouses, control decisions include—but are not limited to—selecting a put-away location for incoming goods, releasing customer orders into the fulfillment process, grouping customer orders, and routing pickers. Obviously, control decisions are critical to the effective operation of a warehouse.

The reference model for DELS also includes (operational) control, which can be achieved through identifying a canonical set of control questions that define a comprehensive functional specification of all the control mechanisms that a controller needs to be able to provide. We believe the canonical set of control questions is: (1) “should a task be served?” (admission); (2) if so, then “when should the task by serviced?” (sequencing); and, (3) “by which resource?” (assignment); (4) finally, “where should the

task be sent after it's complete?" (routing); (5) as well as the resource-related "when does the state of a resource need to be changed?".

In the warehousing context, **the canonical questions have the following interpretation:**

(1) admission control may evaluate available resource capacity, both inventory on hand and fulfillment capacity, to decide whether or not to release an order;

(2) sequencing customer orders include decisions such as coordination (of orders outbound on the same truck), batching (wave planning and batching), delaying (backordering orders), and splitting (splitting a complete order to be picked from different zones);

(3) many interrelated resource allocation problems including assigning labor and material handling equipment to load and unload the carrier trucks and pick and put-away products into storage. However, there are also auxiliary resources such as docks, sorter lanes, and storage locations that need to be assigned to trucks, orders, and SKUs to be stored, respectively;

(4) routing, or process planning, includes building routes for pickers and optimal routing for AMHS, as well as routing orders through the facility to the different processes required or when additional processing is required such in the case of exceptions or quality inspection; and

(5) changing the capability or capacity of resources includes replenishment of stocked inventory, maintenance on automated systems to maintain capacity, or anticipatory moves and pre-positioning of inventory or AMHS pickers/vehicles.

These control decisions must be determined by the controller in some way, e.g., using state-based rules, using an optimization model, or some other mechanism. As a useful abstraction, we can think of the controller's decision problem formulated in terms of decision variables, constraints, and an objective, for each of the five questions. In other words, we can think of an abstract analysis model for each question.

The control questions are captured in the reference model as a design pattern that maps the decision variables in the controller's decision problem to a particular controller behavior and to an execution mechanism in the plant. Figure 2 presents a controller architecture that accommodates these concepts.

This mapping between the execution mechanism in the system model and the analysis model in the controller can be used to catalogue the extant literature on operations control by organizing the extensions to the base system, i.e., each analysis model extends the base model by incorporating unique product, process, and resource behaviors, properties, and constraints. Therefore, a standard representation of each of these classes of operational

control problem is important to enable a uniform interface to solution tools and opportunities for interoperable, or plug-and-play, analysis tools.

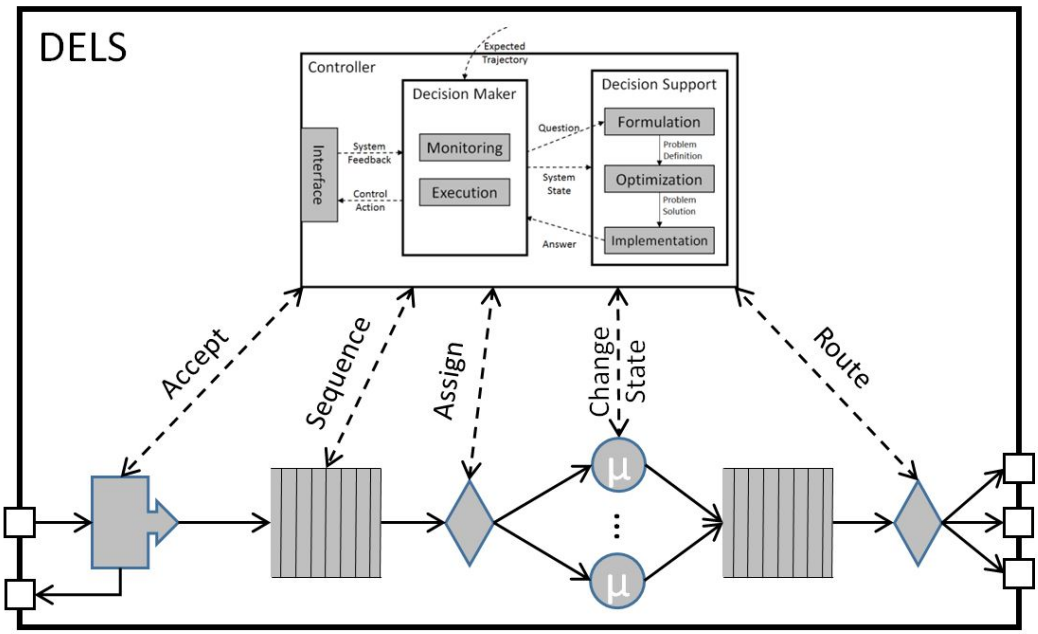

Figure 2 This stylized process map captures the control mechanisms in the base system, the functional architecture of the controller, and the interaction between the plant and control.

The reference model, product/process/resource semantics, and plant/control characterization of DELS provide a starting point for developing a formal semantic model, or modeling framework, for warehouse description, and thus for warehouse design.

2.3 Using the Reference Model

The warehouse reference model provides the semantic foundation for the community to contribute methods and tools organized around a single domain reference model. However, the test of any proposed semantic model is its ability to support both the specification of a warehouse, the process for creating a warehouse specification, and the development of tools to support warehouse design decision making.

Figure 3 illustrates how the reference model can be used as a pattern to specify elements of structure and behavior, in this case for a storage system from which items will be picked to fulfill customer orders. The pattern suggests components that need elaboration or how existing components can be composed into a system definition, but does not detail exactly how to build or select those components, e.g. the overall structure of the *StorageDept* is

understood even though the specific storage and pick equipment are not yet specified, nor are the mechanisms for making the controller decisions (Figure 3).

One of the primary goals for developing the reference model and associated design methods is to define rules and constraints for valid system definitions in such a way that the search process can be partially or completely automated. For example, the *CrossAisle*, *Shape*, and *Aisle* components of the *StorageDept* in Figure 3 can be used can be used to automate the manipulation of the underlying network structure (Jeff Smith IMHRC 2014).

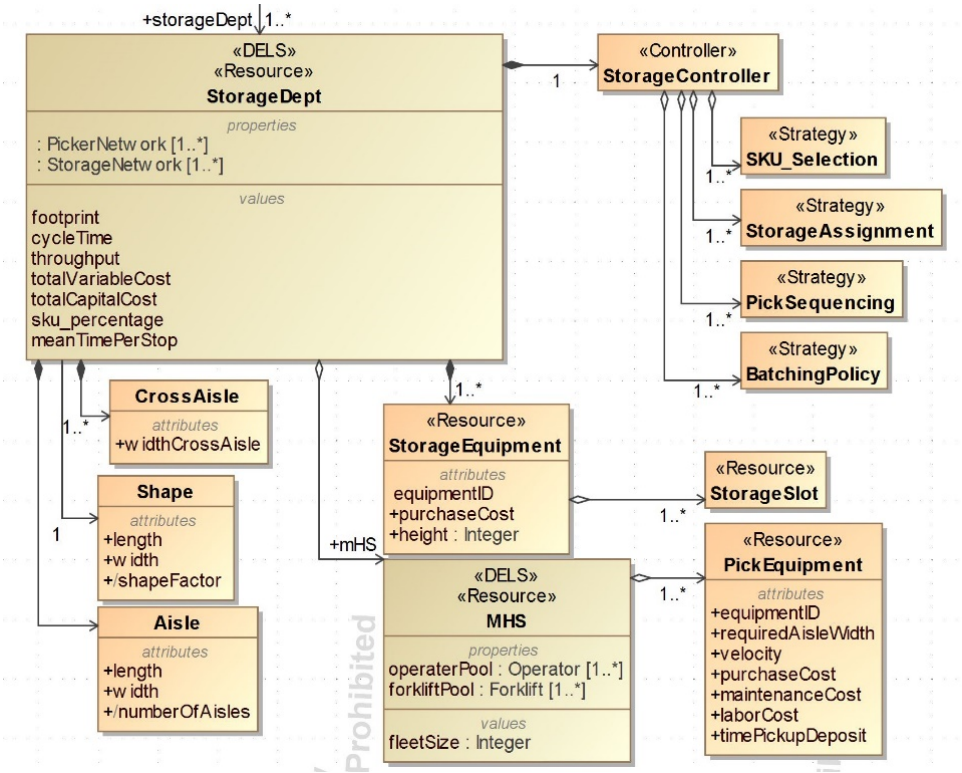

Figure 3: A system model for a forward pick area can be elaborated from the generic reference model.

3 A Design Workflow

To be useful, a design workflow must be specific enough to identify the activities to be performed at each stage of design, and the product of each stage of design. It also must accommodate iteration in some fashion, as design is seldom a simple path from start to finish, but often requires backtracking to reconsider prior design decisions. The goal of a design workflow is not to identify the “one best design” but to enable reliable and

repeatable search of the design space and support analyzing and comparing the best designs found in that search.

One approach to developing a warehouse design workflow is to follow the general workflow proposed in classical design texts, such as [16], i.e., to first clarify the design task (*define the context*) and *analyze the requirements,* then create a *functional design,* then create a *design architecture,* specify the *technology* for each architectural element, and finally, *size and configure* each selected architecture. Each of these stages are linked together by the system model constructed from the reference model and domain specific language.

3.1 Context of the System

At the highest level of abstraction, a warehouse consumes shipments from suppliers and produces shipments to customers. The first step in functional design is to characterize these two types of shipments, in terms of the relevant attributes of frequency, composition and size or amount. For example, a warehouse may receive 100 full truck loads per day, each truckload consisting of 26 pallets, each containing an average of 48 cartons of a single SKU from a population of 10000 SKU, and it may ship 15600 customer orders per day, with an average of 4 lines per order, each line averaging 2 cartons, requiring 200 third party carrier pickups per day.

From this, we could identify the following functions, and corresponding flow rates:

- **receive** 2600 pallets per day, of 100 different SKUs
- **put-away** 2600 pallets per day
- **retrieve** 124800 cartons per day, from 62400 locations
- **assemble** 15600 customer orders per day
- **ship** 15600 customer orders per day

3.2 Functional Requirements Analysis

These flow rates are clearly too large for the designer to consider each individual flow, so the natural next step is to aggregate data in some fashion. The *functional requirements analysis* design stage identifies families of SKUs and orders based on some kind of "similarity", e.g., similar size, similar flow rate, etc. The assumption is that on the inbound side of the warehouse, all SKUs in a family will be treated the same way. Similarly, it makes sense to try to aggregate similar orders on the outbound side, based, e.g., on the outbound unit of handling. Some orders might be full pallet, some partial pallet, some over-packs, and some single cartons. As before, all order types in the same family would be treated the same with regard to packing and shipping.

3.3 Functional Design

The essence of functional design is to specify "what must happen" to transform the inbound SKUs into outbound orders without worrying about "how it will happen". The process transforms the knowledge about the INFlows and OUTFlows into an abstract conceptualization of the transformation process that occurs in between. This conceptual functional flow can be captured in a functional requirements network [14], [15], and [20].

Design patterns, such as the functional requirements diagram in (Figure 4), are a component of the system reference model to not only provide guidance on constructing new FRNs but also can be augmented with rules/constraints for checking the validity of the new FRN. For example, the axioms articulated in [14] can be embodied in a design pattern included in the reference model; e.g. every OUTFlow (order family) must be accompanied by an ASSEMBLY function and every INFlow (SKU family) must be accompanied by a STORE function.

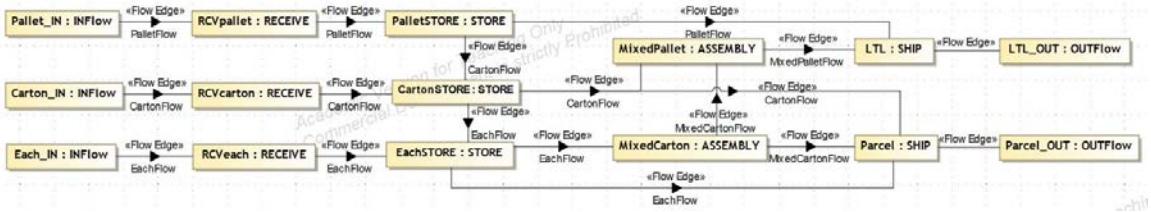

Figure 4 The warehouse reference architecture includes a pattern for constructing new functional requirements networks.

Capturing the functional design as a network enables the design to be analyzed using flow network based computational analysis models. A formally (or at least computationally) specified system model can provide the basis for integrating computational tools which transform the candidate design into a computational analysis model, such as one that can analyze the flows or queuing network properties of the proposed functional design (Section 4).

3.4 Embodiment Design

The functional requirements and functional design consider the product (orders and order families) and the abstract process to transform inbound SKU (inventory stock resources) into those products. The embodiment design must specify the resources that will execute those process as well as the configuration of those resources within a facility, all while refining the description of the transformation process (the coupled replenishment and fulfillment processes embodied by the couple DELS). This is where most of the research

in the area of warehouse design takes place and typically assumes a given set of design decisions produced implicitly by the two preceding design steps.

The reference architecture can be extended to include model libraries of system components, e.g. resource behaviors, process models, product flow definitions, and control behaviors, policies, and algorithms; that are essential to facilitating embodiment design. This is a practical application of the knowledge capture capability of formal domain modeling.

4 Computational Tools

The computational tools to support design must enable a seamless design workflow, including necessary design iteration. One approach to achieving this seamless integration is to conceptualize the design process from an object modeling perspective, and identify the required *interfaces* of the needed computational components. With well-defined interfaces, it becomes possible to achieve the kind of plug-and-play use necessary to support the design workflow.

Supporting the design workflow with computational tools first requires defining what types of methods and tools are useful at each stage of the design. As illustrated in Figure *6*, there are many specialized methods for optimizing isolated components of a warehouse, e.g. facility layout, resource/fleet sizing and composition, order picking/routing, etc. Selecting the correct or best analysis tool from a collection of specialized method requires identifying the set of decision variables, the level of abstraction appropriate to the level of detail in the design, and any particular constraints imposed by the system model, both endogenous and exogenous of the design process itself.

To formulate an analysis model to answer a particular question, each of these specialized methods requires extracting an appropriate view of the system model that

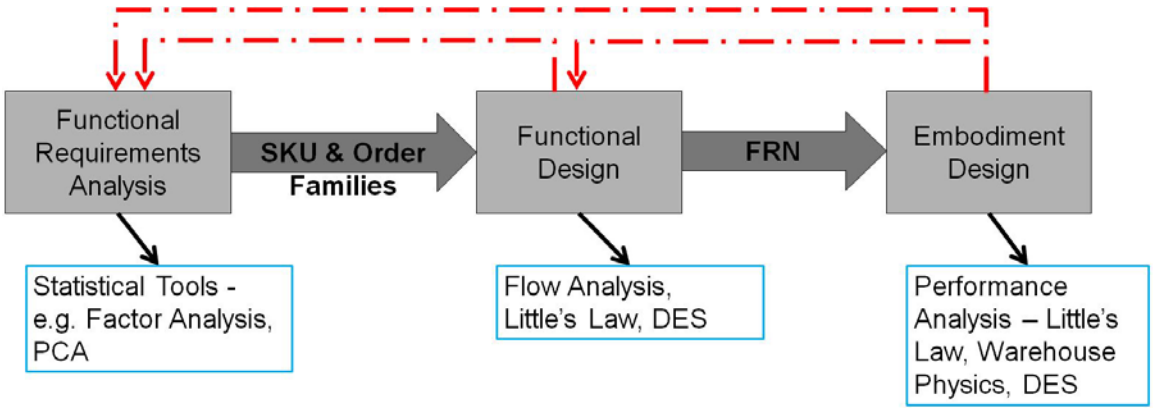

Figure 5 Each stage of the design process has a set of analysis tools that support decision making at that layer of detail.

contains the system information in the appropriate level of detail, fidelity, or aggregation. After solving the analysis problem, there is also the challenge of translating or implementing the solution in the system model, whether it is a network reconfiguration or an optimized set of control rules.

4.1 Using Decomposition Approaches to Search the Design Space

In the past, the warehouse design workflow has been conceptualized independently of any associated computation. In contrast, if we look at contemporary engineering design in general, the workflow and the computational tools are evolving together. For example, in complex systems design, multi-disciplinary design optimization methods incorporate many tools to address the computational expense and organizational complexity of optimizing these systems, including decomposition and hierarchical optimization methods, approximation techniques, and tools to facilitate efficient organization of models and data [19]. Similarly, hierarchical design and control procedures decompose the problem to reduce the complexity, see, e.g. [9], [7], and [22]. While decomposition doesn't necessarily reduce the total computational effort, the decomposition can align with domain specialties and allow for concurrent design; i.e., within each subsystem, domain specialists can select appropriate approximations for the current design phase and tailor them to the specific application as necessary.

One way to decompose the design space is by design stages, e.g. functional requirements analysis, functional requirement network (design), and embodiment design. Within the embodiment design stage where most of the finer details are resolved, the design space can be decomposed further to address the network structure, behavior (product, process, resource, and facility descriptions), and control layers separately. Figure 6 illustrates how such a decomposition might be applied to elaborate the functional design in Figure 3. The approach illustrated in Figure 5 supports an object oriented approach where

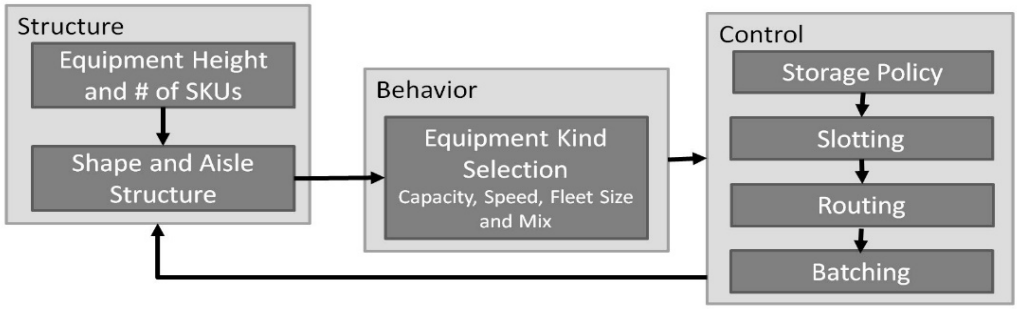

Figure 6: One way to decompose the design process is to use the reference architecture as a template.

the design problem is decomposed into design sub-tasks that correspond to the structure of the Storage System in Figure 3, i.e. each component of the *StorageDept* can be designed separately and then integrated into the system model. One question that arises from decomposing the design process is "is there a canonical design hierarchy?", i.e. should we always decompose the system in the exact same way or are there at least a set of principles guiding the decomposition process?

Applying this decomposition approach requires a default value, or proxy model, for each component while independently, iteratively, and simultaneously improving the design of all the other components. The plug-and-play nature of these components relies on object-oriented modeling principles and well defined interoperability between the components and the system model. While progress can be made on optimizing each component of the system independently, an explicit system model guarantees that the solutions can be integrated into a coherent and complete system model that can be verified and validated (by simulation or otherwise).

5 Fundamental Challenges

There remain many fundamental challenges to making warehouse design resemble an engineering design methodology; below we highlight several challenges that align with the system modeling and semantics, design workflow, and computational tools components.

Complete System Model: Interestingly enough, one might argue that computational power is cheap enough and cloud computing powerful enough that the search tools are sufficient for the task at hand, but there needs to be further research and effort on modeling the behavior and control of these systems. These models need to be integrated and interoperable with the system model and enabling a specification of behavior as designed and exogenous effects including uncertainties in the order profile, for any candidate system design.

Modeling Operational Control: While interfaces to control optimization algorithms and formalisms for expressing control behaviors are integral to formulating analysis models and executing the result, these methods begin to define a pathway from designing control behaviors, to testing them, and finally transplanting the control code directly into the warehouse management and control systems. However, this design methodology requires researching methods and implementing the corresponding tools that support design and verification on complex control behaviors, both on the optimization and evaluation (simulation) methods.

Generating Synthetic Data: In addition to modeling the behavior of the warehouse using the PPRF language, a major challenge in warehouse modeling and design is the dynamic and uncertain nature of the "product" requirement, or the future order stream.

Designing systems capable of handling the uncertainty requires the ability to search over the likely future product mix and demand and the only way to do this is to be able to generate synthetic order steams that are indistinguishable from real order streams. This "sampling" of the prospective *INFlow* and *OUTFlow* of the system is not as straightforward as generating random numbers from a probability space.

Value Driven Approaches to Multi-Objective Design: As we think about the design workflow and the tools and methods needed to support it, we will need to reconsider how we conceptualize the warehouse design problem. It is not simply a matter of "meeting throughput requirements," or "minimizing cost." Rather it is creating the warehouse design that maximizes value. One important element of a research agenda will be understanding how value is or should be defined, and how that definition of value can drive the warehouse design workflow.

Design-focused Analytics Toolbox: Given an agreed upon system model and design workflow, there remains a need for repository of plug-and-play analysis models that conform to that language/reference model and enable that design workflow. Developing a useful toolbox also requires defining what types of methods and tools are useful at each stage of the design. The process of developing these tools also incorporates the challenge of cataloguing and benchmarking the existing knowledge and then making it available in a form that is broadly applicable and reusable.

Following up on the comment about current search capabilities discussed earlier in this section, in McGinnis [14], one strategy for doing functional requirements analysis is to “Try many different attribute selections, complete the functional and embodiment design for each, and choose the one that leads to the best embodiment design,” but the author feels that “the first strategy is unlikely to be practical, due to the time and cost of detailed functional and embodiment design.” Is this still true? Will it remain true for the foreseeable future?

However, to accomplish this level of automated and integrated computation support this is requires evaluating the quality of the existing methods and tools against how well they meet the needs of designing complex systems and identifying requirements for new methods and tools, e.g. multi-fidelity simulation tools that conform to the reference model and language and are capable of capturing behavior and control.

6 Conclusions

We need not start from scratch in working toward these goals. Some initial efforts to develop reference models have been reported [13], [14]. Similarly, there have been efforts to integrate simulation tools into a warehouse design process [15], [20]; and to develop a hierarchical design method using multi-fidelity simulation methods [21].

What is needed moving forward is a comprehensive research agenda addressing all three of the required elements for a discipline of engineering design for warehouses. Initial efforts to create a domain specific language (DSL) for warehousing are promising, but much remains to be done, both in terms of defining structure, but especially in terms of defining control. Perhaps the most challenging aspect is the way we think about warehouse design, or the warehouse design workflow, because this is largely an unexplored issue in the field. The development of "plug and play" analysis tools depends upon the development of agreed-upon interfaces, which themselves depend upon an agreed-upon semantics.

It may seem like these are impossible challenges, but that is a backward-looking perspective. Today, the platforms and tools to support the required collaborative development are readily available, and often free. For example, GitHub provides a platform for a community to develop and share open source software. Object modeling is well supported by the Eclipse community, and Ecore is an open-source alternative to SysML. These are not the only technology options. If we can agree on what analyses are needed, and agree on the interfaces to these analyses, then we can begin to build, test, and distribute computational modules to support warehouse design.

As never before, the time is right for this kind of community-based initiative to bring real engineering discipline to the field of warehouse analysis and design.

7 References

[1] anon, 2016a, http://www.oxforddictionaries.com/us/definition/american_english/discipline.

[2] anon 2016b, http://www.merriam-webster.com/dictionary/discipline.

[3] Apple, J.M., Meller, R.D. and White, J.A., 2010. “Empirically-based warehouse design: Can academics accept such an approach.” *Progress in Material Handling Research*.

[4] Baker, P. and Canessa, M., 2009. “Warehouse design: A structured approach.” *European Journal of Operational Research*, *193*(2), pp.425-436.

[5] Cloutier, R., Muller, G., Verma, D., Nilchiani, R., Hole, E. and Bone, M. (2010), “The Concept of Reference Architectures.” *Systems Engineering*, 13: 14–27.

[6] De Koster, R., Le-Duc, T. and Roodbergen, K.J., 2007. “Design and control of warehouse order picking: A literature review.” *European Journal of Operational Research*, *182*(2), pp.481-501.

[7] Goetschalckx, M., McGinnis, L., Sharp, G., Bodner, D., Govindaraj, T. and Huang, K., 2002. “Development of a design methodology for warehousing

systems: hierarchical framework." In *Proceedings* of the I*IE Annual Conference.* Institute of Industrial Engineers.

[8] Golson, J., 2015. "GM's Using Simulated Crashes to Build Safer Cars". *Wired*. Available Online: www.wired.com/2015/04/gms-using-simulated-crashes-build-safer-cars.

[9] Gray, A.E., Karmarkar, U.S. and Seidmann, A., 1992. "Design and operation of an order-consolidation warehouse: Models and application." *European Journal of Operational Research*, *58*(1), pp.14-36.

[10] Gu, J., Goetschalckx, M. and McGinnis, L.F., 2010. "Research on warehouse design and performance evaluation: A comprehensive review." *European Journal of Operational Research*, *203*(3), pp.539-549.

[11] Gu, J., Goetschalckx, M. and McGinnis, L.F., 2007. "Research on warehouse operation: A comprehensive review." *European journal of operational research*, *177*(1), pp.1-21.

[12] Mönch, L., Lendermann, P., McGinnis, L.F. and Schirrmann, A., 2011. "A survey of challenges in modelling and decision-making for discrete event logistics systems." *Computers in Industry*, *62*(6), pp.557-567.

[13] McGinnis, L., 2010. "The Future of Modeling in Material Handling Systems", in: 11th International Material Handling Research Colloquium - 2010. Material Handling Industry of America.

[14] McGinnis, L., 2012. "An object oriented and axiomatic theory of warehouse design", in: 12th International Material Handling Research Colloquium—2012. Material Handling Industry of America.

[15] McGinnis, L. and T. Sprock, 2014. "Integrating Analysis into a Warehouse Design Workflow", in: 13th International Material Handling Research Colloquium - 2014. Material Handling Industry of America.

[16] Pahl, G. and Beitz, W., 2013. *Engineering design: a systematic approach.* Springer Science & Business Media.

[17] Rouwenhorst, B., Reuter, B., Stockrahm, V., Van Houtum, G.J., Mantel, R.J. and Zijm, W.H.M., 2000. "Warehouse design and control: Framework and literature review." *European Journal of Operational Research*, *122*(3), pp.515-533.

[18] Saxena, Anupam and Birendra Sahay, 2005, *Computer Aided Engineering Design*, Springer, New York.

[19] Sobieszczanski-Sobieski, J., Haftka, R.T., 1997. "Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments". *Structural optimization* 14, 1–23.

[20] Sprock, T. and McGinnis, L.F., 2015. “Analysis of Functional Architectures for Discrete Event Logistics Systems (DELS).” *Procedia Computer Science*, *44*, pp.517-526.

[21] Sprock, T., and L. F. McGinnis. 2015. “A Simulation Optimization Framework for Discrete Event Logistics Systems (DELS)”. In Proceedings of the 2015 Winter Simulation Conference. Piscataway, NJ, USA: IEEE Press.

[22] Subrahmanyam, Sriram, Joseph F Pekny, and Gintaras V Reklaitis. 1994. “Design of batch chemical plants under market uncertainty.” Industrial & Engineering Chemistry Research 33 (11): 2688-2701.

[23] Thiers, George. 2014. “A Model-Based Systems Engineering Methodology to Make Engineering Analysis of Discrete-Event Logistics Systems More Cost-Accessible." Ph.D. thesis, Georgia Institute of Technology, Atlanta, GA.