Published in final edited form as:

Manuf Lett. 2018 ; 15(Pt B): . doi:10.1016/j.mfglet.2018.02.002.

Self-Similar Architectures for Smart Manufacturing and Logistics Systems

Timothy Sprock^{a,*}

^aNational Institute of Standards and Technology, 100 Bureau Dr, MS-8260, Gaithersburg, MD 20899

Abstract

Self-similar architectures simplify design and interoperability for smart manufacturing and logistics systems. Self-similar architectures for these domains extend existing models of self-similar controller architectures to include unified models, or abstractions, of plant behavior and planning and operational control functions. This paper proposes modeling system units as discrete event logistics systems (DELS), an abstraction that unifies heterogeneous system models for the manufacturing and logistics domain. Similar DELS then can be composed into self-similar architectures for larger, more complex DELS.

Keywords

Smart Manufacturing; Discrete Event Logistics Systems (DELS); System Modeling; Operational Control

1. INTRODUCTION

Challenges in designing and operating smart manufacturing and logistics systems include interaction and coordination between heterogeneous systems, such as material handling systems, storage and retrieval systems, and production systems. Interoperability challenges extend to heterogeneous planning and operational control software systems, such as manufacturing execution systems (MES), transportation management systems (TMS), and warehouse control systems (WCS).

Agent-oriented system architectures, such as fractal and holonic manufacturing systems, have been proposed to address these challenges [1, 2, 3]. Self-similar system architectures, a feature of fractal manufacturing systems (FrMS), are an attractive idea for designing smart manufacturing and logistics systems. Self-similar systems are exactly or approximately similar to a part of itself; for example, a factory can be similar to each department, which is similar to each work cell. The similarity of system components improves the composability and interoperability of cyber-physical components. It reduces the complexity of system design and improves maintainability of the system.

*Corresponding author: timothy.sprock@nist.gov (Timothy Sprock).

The proposed methodology models smart manufacturing and logistics systems as composed of architecturally similar units performing specialized functions: first, decompose each production and logistics system into smaller, functionally-distinct units (Section 2), then model each unit using a common abstraction to achieve similarity (Section 3), and finally compose architecturally-similar, functionally-diverse units into self-similar systems (Section 4).

2. Decomposition of Production and Logistics Systems

Manufacturing and logistics systems are compositions of smaller operational units, each contributing distinct functional capabilities to the whole system. For example, supply chains are composed of production systems connected by transportation and storage systems (Figure 1), aligning with the basic *MAKE*, *MOVE*, and *STORE* functions. Each system component can be described as a Discrete Event Logistics System (DELS). DELS are dynamic systems that transform discrete flows through networks of interconnected resources [4, 5]. DELS are found at all levels of the enterprise, including large, complex systems such as supply chains, manufacturing systems, transportation networks, and warehouses.

The decomposition process can be applied recursively if each smaller unit retains decision-making authority. Decision-making implies that each unit, or DELS, has higher-level planning and detailed operational control functions. This broadens DELS to cover more granular systems such as individual departments of factories or warehouses, workstations, production lines, automated storage and retrieval systems (ASRS), and material handling systems.

Designing similar DELS architectures requires a consistent definition for each component. The next section describes a common abstraction for DELS and uses it to construct a reference architecture for new system components.

3. Unified, “Similar” System Description

The concept of self-similar architectures for manufacturing originates in fractal manufacturing systems (FrMS) [6, 3]. FrMS models describe a functional controller architecture for each fractal unit that achieves properties such as self-organization, self-optimization, and self-similarity. Our research focuses on developing self-similar system architectures for DELS. A reference architecture is a pattern to create new systems, or components. The reference architecture for DELS extends similar system specifications beyond functionally, or architecturally, similar controllers to include uniform definitions of the system behavior, including both plant and control.

Traditionally, each kind of DELS is addressed with dedicated domains, including domain-specific languages and reference models. However, these systems share a common abstraction — a set of *products* flowing through a set of *processes* being executed by *resources* configured in a *facility* (Figure 2) [5]. *Tasks* authorize DELS to execute processes. This DELS abstraction can be specialized to describe systems such as: (1) a workcell with machines connected by conveyors and queues; (2) factory with work-cells connected by material handling, such as forklifts or automated guided vehicles (AGVs), and material

stocking points; and (3) supply chains connecting factories with multi-modal transportation systems and warehouses.

Planning and control of specialized DELS often rely on dedicated domain-specific decision-support software. Unified models of system behavior enable operational control decision-support methods, such as scheduling, to be integrated across heterogeneous systems. The DELS abstraction includes functions for controlling the flow of tasks and resources through the system (operational control functions). Informally, these are: (1) ‘should a task be served?’ (*admission*); (2) if so, then ‘when should the task be serviced?’ (*sequencing*); and, (3) ‘by which resource?’ (*assignment*); (4) finally, ‘what process step does the task require next?’ (*dynamic process planning*); (5) as well as the resource-related ‘in which state does a resource need to be?’ (*change-state*). For each control function, the operational control abstraction links decision-making by the controller to actuators in the plant that execute the decision [7].

Common abstract reference models and architectures for DELS provide similarity between system components, enabling interoperability among heterogeneous DELS, including physical and software components. Interoperability enables similar components to be composed into self-similar systems.

4. Composing Self-Similar Systems

DELS-based similarity of system components enables model-based, object-oriented approaches to designing and assembling self-similar DELS. Specifically, one DELS can be a resource to another (Figure 3). This relationship is modeled formally by combining generalization (DELS is-a-type-of Resource) and composition (DELS has Resources as parts) relationships between DELS and its resources. For example, a workcell views its machines, conveyors, and buffers as resources capable of make, move, and store functions, respectively. Likewise, a supply chain may view its factories, transportation system, and warehouses as resources capable of make, move, and store functions, respectively.

Designing and operating self-similar DELS requires modeling the functional capabilities provided by resources and required by tasks. The capabilities of each resource are exposed to other systems through well-defined interfaces. Then each DELS controller views its plant exactly the same way (“similarly”) regardless of whether those resources are independent, autonomous DELS; coalitions of resources under the direction of a mediator; or simple owned equipment. That is, each DELS plans and controls its resources similarly regardless of hierarchical level in the enterprise.

Similar system components can be composed into a variety of architectures, such as centralized, hierarchical, heterarchical, decentralized (described in [8]), or other architectures such as mediator [9].

5. Conclusion

Self-similar system architectures specialized from a common abstraction streamlines the design and integration of functionally and hierarchically heterogeneous DELS. Similar

components can be composed and reorganized into many network configurations. This enables flexible, gradual transitions from traditional hierarchical architectures to more flexible, agile architectures. Object-oriented, model-based systems engineering methodologies that use unified, self-similar architectural patterns to design self-similar DELS produce systems that are conceptually simpler to design and maintain, especially the control software and cyber-physical components.

Developing and implementing self-similar architectures for DELS requires work on models and methodologies for building them. These include standardized DELS abstractions, reference architectures derived from those abstractions, and standardized methodologies for refining or specializing them into system architectures. Models of existing systems (brownfield implementations) are needed that conform to standard abstractions, enabling them to take advantage of interoperability and integration opportunities. Leveraging these opportunities to develop unified, integrated systems is necessary to make “smart” DELS.

Acknowledgments

The author thanks Conrad Bock for his helpful conversations and comments.

Commercial equipment and materials might be identified to adequately specify certain procedures. In no case does such identification imply recommendation or endorsement by the U.S. National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

References

1. Duffie N. Heterarchical control of highly distributed manufacturing systems. *International Journal of Computer Integrated Manufacturing*. 9(4)1996; :270–281.
2. Van Brussel H, Wyns J, Valckenaers P, Bongaerts L, Peeters P. Reference architecture for holonic manufacturing systems: Prosa. *Computers in Industry*. 37(3)1998; :255–274.
3. Ryu K, Jung M. Agent-based fractal architecture and modelling for developing distributed manufacturing systems. *International Journal of Production Research*. 41(17)2003; :4233–4255.
4. Mönch L, Lendermann P, McGinnis LF, Schirrmann A. A survey of challenges in modelling and decision-making for discrete event logistics systems. *Computers in Industry*. 62(6)2011; :557–567.
5. Sprock, T. Ph D thesis. Georgia Institute of Technology; Atlanta, GA: 2015. A metamodel of operational control of discrete event logistics systems (dels).
6. Tirpak TM, Daniel SM, Lalonde JD, Davis WJ. A note on a fractal architecture for modelling and controlling flexible manufacturing systems. *IEEE Transactions on Systems, Man, and Cybernetics*. 22(3)1992; :564–567.
7. Sprock, T. Patterns for modeling operational control of discrete event logistics systems (DELS). In: Madni, AM, Boehm, B, Ghanem, RG, Erwin, D, Wheaton, MJ, editors. *Disciplinary Convergence in Systems Engineering Research*. Springer International Publishing; 2017.
8. Dilts D, Boyd N, Whorms H. The evolution of control architectures for automated manufacturing systems. *Journal of Manufacturing Systems*. 10(1)1991; :79–93.
9. Maturana FP, Norrie DH. Multi-agent mediator architecture for distributed manufacturing. *Journal of Intelligent Manufacturing*. 7(4)1996; :257–270.

Figure 1.

Each system specializes an abstract template for DELS. Here the specialization aligns with basic functions of system components. These components are assembled into complex systems, such as supply chains.

Figure 2.

The behavior of each functional unit conforms to a common abstraction. This abstraction simplifies the description and design of the planning and control systems.

Figure 3.

Self-similarity is achieved by allowing DELS to be resources to other DELS. This recursive relationship allows similar DELS to be composed into complex system architectures while preserving the similarity of the parent DELS (self-similarity).