Generic and reusable structure in systems dynamics modelling: roadmap of literature and future research questions

Sondoss ElSawah[1,2], Alan McLucas[3], Mike Ryan[3]

[1]Post-doctoral researcher, School of Information Technology and Electrical Engineering, University of New South Wales, Canberra, Australia

[2] Adjunct Fellow, Integrated Catchment Assessment and Management Centre, Fenner School of Environment & Society, Australian National University, Canberra, Australia

[3] Senior lecturer, School of Information Technology and Electrical Engineering, University of New South Wales, Canberra, Australia

UNSW Australia at the Australian Defence Force Academy Northcott Drive, Canberra ACT 2600

Ph. +61 (0)2 6125 9021, +61 (0)4 3030 3946

Fax 61 (0)2 6125 8395

s.elsawah@unsw.edu.au

Abstract

In the systems dynamics literature, some research efforts have focused on the idea of developing generic and reusable modelling elements. This literature is largely fragmented, which makes it challenging, especially for newcomers, to synthesise what has been achieved and to identify potential research directions. Key insights from the paper are that there is little research into the process of designing reusable structures, and there is a clear gap between the design of these structures and how they can be used effectively in practice. We envisage that emerging research directions such as exploratory system dynamics modelling and XMILE may present opportunities to refresh interest in the topic.

Keywords: generic structures, molecules, system dynamics-meta model, object-oriented

Introduction

In the first issue of the Systems Dynamics Review, Paich (1985) presented the topic of “generic structures” as a research problem, and put forward to the system dynamics community questions about their meaning, research value, utility to policy makers, and the approach to identify and validate them. A decade later, Lane and Smart (1995) tracked the evolution and application of the concept in system dynamics literature. They argued that the concept carries multiple interpretations and offers various contributions depending on how it is applied to different activities in the systems dynamics fields.

Since then, there have been some research attempts tackling the topic. Work done broadly falls into two lines: (1) studies that aim to identify generic structure as a theory to link structure and behaviour, and (2) studies that aim to improve modelling efficiency and domain-relevance. Both directions share the view of generic structures as useful vehicles for transferring knowledge. Looking at the area, one can conclude that the work seems fragmented with no clear roadmap of what has been achieved, and little focus in potential research directions. In this paper, we aim to contribute to addressing this perceived gap.

The paper is structured around three objectives. Firstly, we present concepts and approaches focused on the topic of generic and reusable structures in systems dynamics. We take (Lane and Smart paper, 1995) as a starting point for our review, and will only refer to work done earlier if it is essential for the paper's argument and flow. Secondly, we organize reviewed studies into three areas according to where they contribute to the development of the concept: (1) definition and conceptualization, (2) formal model building and software implementation, and (3) validation and evaluation. Finally, we discuss gaps and propose ideas for future research questions.

Concepts and approaches

In this section, we present an overview of research work done on using generic and reusable model structures. We organize the discussion around the concepts and approaches emerging from the literature, and illuminate their differences (See Table 1), and relationships (See Figure 1). Studies broadly fall into two research lines. First are studies that aim to identify generic structure as a theory to link structure and behaviour, such as work related to Lane's three interpretations of generic structures. Second are studies that aim to improve modelling efficiency and domain-relevance. This line of work includes worked related to object-oriented extensions of system dynamics, and system dynamics meta-modelling. We think the work done on the molecules includes aspects of both.

Table 1: An overview of the key concepts focuses on developing generic and reusable approach for system dynamics modelling

Concept/approach	Definition	Representation	Key strengths	Key limitations
Canonical situation models	A general model which encapsulates the essential structure required to explain the dynamic behaviour at particular problem situation	Stocks and flows	Fully-tested and calibrated models that can be parametrized to allow for case-specific experimentation	The fact that the model is readily brought into the process may (in some cases) trigger lack of confidence in the model and underpinning assumptions.

Abstracted microstructures	Elementary structures of all system dynamics model necessary to explain the system behaviour	Stocks and flows	Useful for communicating the fundamental sources of dynamics especially for teaching and model conceptualization purposes	They cannot be used for meaningful policy analysis until the model is fully built
Counter-intuitive archetypes	A conceptual map that boils down some key insights about the relationships between feedback interactions and unintended consequences	Causal Loop Diagrams	Useful for communicating about the value of feedback thinking	Difficult to convert into a numerical model
Molecules	Standard pieces of structure organized into a hierarchical system linked by the inheritance relationsho[	Stocks and flows	The hierarchical structure of molecules promotes an incremental understanding of complexity and different aggregation levels as the model evolves	The model users still have to build the model from scratch using stocks-and-flows, which (in some cases) may be cognitively challenging and not very relevant to the domain.
System dynamics meta model	High-level architecture that includes domain objects and relationships	High level programming language	The model users can interact with the model in their own domain language.	The underpinning system dynamics model is a black box for users.
System dynamics object oriented Components	Software programs that include the model equations and diagrams (including but not limited to	Stocks and flows	The model users can interact with the model in their own domain language, but can navigate through the hierarchal	High investment and skills are required in designing and developing domain components

	stocks and flows diagrams)		structure to view the underpinning system dynamics model	

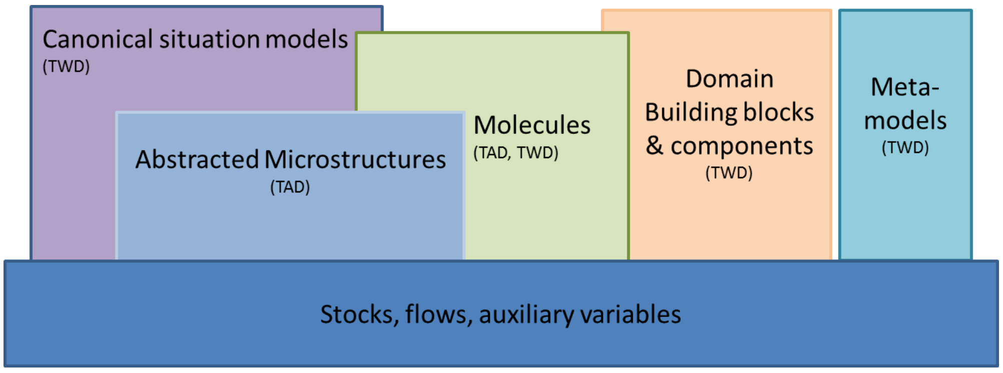
Canonical situation models
(TWD)
Abstracted Microstructures
(TAD)
Molecules
(TAD, TWD)
Domain Building blocks & components
(TWD)
Meta-models
(TWD)
Stocks, flows, auxiliary variables

Figure 1: The relationship between the various concepts (Notes TAD: Transferable Across Domains, TWD: Transferable Within Domains, counterintuitive archetypes are not included because of their different nature from the rest of concepts)

Lane's three interpretation of generic structures in system dynamics

Lane and Smart (1995) paper made the first attempt to organize the discussion around the concept of generic structures with three key contributions. First, it tracks the historical evolution of the generic structure concept in system dynamics literature. Second, it uses this historical context to unpack the concept into three different interpretations: canonical situation models, abstracted micro-structures, and counter-intuitive archetypes. Then, it discusses issues related to the application and validation of the three interpretations.

Canonical situation models

Canonical situation models are defined as 'general models' which encapsulate the essential structure required to explain the dynamic behaviour at particular problem situation. Canonical situation models are numerical model that have been fully formulated and calibrated, and can be specified to a particular case study. A suite of rigorous empirical tests (e.g. sensitivity analysis, family member test) are essential means to distinguish between a canonical situation model and any other formal model.

Examples of classical canonical situation models are Forrester's (1969) urban development model, and Meadows' (1970) production cycles. More recent system dynamics developments branded as canonical situation models in literature include: the acceptance–rejection behaviour (Ulli-Beer et al., 2010); police arrest-domestic violence (Hovmand et al., 2009); resource misallocation among social, asocial, and control parties (Saeed and Pavlov, 2008); and cycles in airlines market (Liehr et al., 2001).

Abstracted microstructures

Abstracted microstructures are the elementary structures of all system dynamics models, including canonical situation models. Andersen and Richardson (1980, p 99) provided a catalogue of microstructures referring to them as elementary structures. Examples include: first-order negative loop and first-order positive loop. Some of these microstructures are implemented as ready-made functions in system dynamics modelling software tools, such as Vensim and Powersim Studio. Microstructures are powerful means for boiling down the dominant structures necessary to explain the dynamic behaviour. From a theoretical viewpoint, microstructures explain the system behaviour at a micro-level (e.g. household behaviour). The macro-behaviour (e.g. market behaviour) is generated when microstructures interact in a macro-model (e.g. canonical situation models). From management viewpoint, these structures present the knobs or leverage points to alter the systemic behaviour. For example, Liehr et al. (2000) identified first-order negative loop with two delays as the microstructure leading the oscillatory behaviour of airlines business, and suggested ordering policies for managing the cycles.

Counter-intuitive archetypes

Counter-intuitive archetypes are conceptualizations of how mismanaging feedbacks may lead to counterintuitive outcomes. Archetypes have gained popularity after Senge's bestseller book "The Fifth Discipline" (Senge, 1990). Senge made effective use of archetypes to present some traps that mangers usually fall into when overlooking system feedback and delays. Wolstenholme (2003) defined a set of four core archetypes, including: underachievement, out of control, relative achievement, and relative control.

Unlike canonical situation models and abstracted microstructures, archetypes can neither be simulated to generate the system behaviour nor allow for policy testing (Dowling, 1995). They are best treated as learning and communication aids which give qualitative insights of what might happen, rather than a theory of what will happen.

Molecules and modelling by replacement

The philosophy behind the molecules is inspired by aspects of generic structures in both the system dynamics literature and modular and reusable structures in computer science. Eberlein and Hines (1996) built on the early work to identify elements of structure by (Richardson, 1981) and (Richmond, 1985). They admit that using the term 'molecule', in the same sense of atoms, to describe and organize universal structures used in different application areas. Yet, molecules are different from the interpretations of generic structures discussed earlier in two ways. First, molecules are conceptualized in a way to allow for creating a catalogue of parent-child relationship between molecules. This relational system encapsulates the 'object-oriented programming' thinking from computer science, and illuminates the interface points between molecules. Second, molecules do not necessarily include stocks, rates, and feedback structures. For example, the 'close gap' molecule is a network of auxiliary variables, which interfaces with the "first-order smoothing" molecule through the 'gap variable' (See Figure 2 for an illustration). Unlike the generic structures, molecules aim to organize pattern structures into an internally consistent system, rather than explain the relationship between structure and behaviour or derive a learning insight. Molecules vary from universal structures (e.g. first-order smoothing) to domain-dependent, mostly in production and logistics management.

The first attempt to catalogue molecules resulted in defining and characterizing about 50 molecules. Molecules are systematically described in terms of their structure, the problem they solve along with technical notes and caveats. This largely follows the formalization of design patterns used in software development to describe software programs and modules (Gamma 1995). Molecules are implemented using Vensim software (Vensim Molecules 2.02). We could only find very few cases that explicitly report utilizing molecules as building blocks in their models, such as: O'Regan (2001), (Sabounchi 2011), (Abdelgawad 2011). Of course, molecules may well be used in other cases but without explicit reporting

More recently, Hines et al. (2011) built on the concept of modelling molecules to develop an approach for constructing models by replacement. The approach has three key features. First, a hierarchical catalogue (taxonomy) is developed of 200 conceptually and mathematically-valid molecules. The most fundamental molecules are stocks, flows, and policies. Molecules are linked into a hierarchy using generalization and specialization relationships. For example, the bathtub is specialization of a stock which has a single inflow and outflow. This structure facilitates navigation through the catalogue. Second, molecules in a working model are substituted with other child molecules in the hierarchy resulting in conceptually and mathematically valid model. For example, the bathtub can be replaced with an ageing chain. The modelling library automatically expands by storing and cataloguing new specializations at their right position in the hierarchy for future use. The approach is implemented in supply chain management application, but the authors invite testing the approach's transferability to other domains.

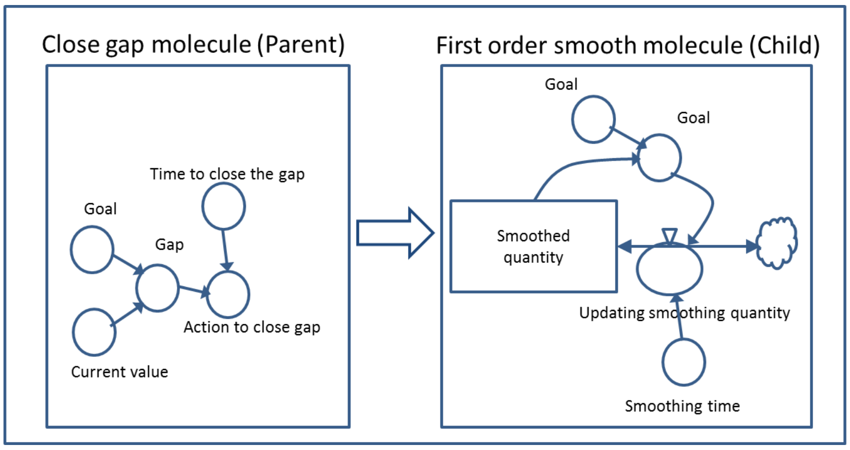

Figure 2: Illustration of Parent-Child relationship between two molecules

Object-oriented extension for system dynamics models

There have been calls for using object-oriented thinking to develop domain-specific building blocks in order to support model development and learning in system dynamics. Vavik and Myrtveit (1995) argued that domain objects have the potential to improve learning by: (1) providing a 'cognitive hook' that the learners can use to relate the model to their real systems,

(2) allowing users to change model structure not only parameters; and (3) allowing users to interrogate the model at different levels of abstraction.

Building blocks are implemented using classes or components which can be organized into a library that can be reused within and between models. High-level domain building blocks denote real world objects (e.g. factory). Ahmed (1997) makes a distinction between a component and generic structures or molecules in system dynamics is that the former is not abstract structure, but represents a well-known domain object. He presents architecture for a hierarchical component-based catalogue in business management area.

The object oriented approach has added a key development into the way reusable structure are viewed by extending the concept to include a series of diagrams that can be used as visualization and learning aids to communicate about the component (Myrtveit, 2000). This is useful distinction as it explicitly triggers thinking into other necessary means (e.g. videos, URLs) that can be clipped to the component depending on the audience and context. Despite its promising potential, very few attempts have been made in this direction, including (Bauer, 2005) and (Powers, 2011).

System dynamics meta-modelling

Similar to the object-oriented extension approach, system dynamics meta-modelling aims to leverage the advancement in software development to improve modelling efficiency and improve model's relevance to domain experts who may not be technical experts. The meta-modelling approach has three main components. First is the domain model which uses classes to define domain entities, their properties, and relationships. Classes are defined using system dynamics constructs (stocks, flows, and auxiliary variables). The architecture of the domain model is designed based on the functional requirements of each class. The second component is model instantiation where the domain classes are used to create instances to represent a specific case problem. The third component is the model-transformation language which compiles and executes instances of the domain model.

In the modelling process, the expert modeller works with the domain experts to create the domain model, which can then be reused to analyse similar problems within the same domain. End user can specify the parameters to tailor the model to a particular application. They do not have to interact directly with the stocks-flows structure of the system, and cannot change the mathematical relationships hard-coded in the system dynamics model.

Barros et al. (2001, 2002) use the meta-modelling approach to model the software development process. The authors compared the meta-modelling and object-oriented extension approach, and argued that the former is easier and more accessible for domain experts because it isolates the end-user totally from the stock-and-flow level of detail. Moreover, they argued that the relationships between components are built using stocks and flows, and therefore mixing different abstraction levels. Similar to the object-oriented approach, very few examples are presented in literature, such as Manataki and Zagrafos (2009) who present a meta-model for assessing the performance of airport terminal systems.

Discussion and future research directions

In this section, we start by providing a roadmap of literature by organizing studies into three areas (See Table 2):

(1) Define and conceptualization: studies which tackle the concept of generic and reusable structures, with the purpose of defining or organizing definitions, discussing uses, presenting conceptualization.
(2) Formal model development: studies which tackle the technical challenges and solutions related to formal development and software implementation of generic and reusable structures.
(3) Validation and implementation: studies which tackle the validation and implementation issues related to generic and reusable structures.

Table 2: Roadmap of literature in the area of generic and reusable structures

Authors	Definition and conceptualization	Formal model development	Validation and implementation
Lane and Smart (1995)	**X**		
Vavik (1995)	**X**		
Ahmed (1997)	**X**		
Lane (1998)			**X**
Liehr et al. (2001a)	**X**		
Liehr et al. (2001b)	**X**	**X**	
Eberlein and Hines (1996)	**X**		
Hines (2010)	**X**	**X**	
Myrtveit (2000)		**X**	
Barros et al. (2001, 2002)		**X**	
Tignor and Myrtveit (2000)	**X**		
Wolstenholme (2003)	**X**		
Sotaquira and Gerly (2004)	**X**		
Bauer (2005)		**X**	
McLucas (2005)			**X**
Saeed and Pavlov (2008)	**X**		
Manataki and Zagrafos (2009)		**X**	
Hovmand et al. (2009)	**X**		
Ulli-Beer et al. (2010)	**X**		
Powers (2011)		**X**	

Our general observation on the work done in the area is that it represents largely fragmented, one-off efforts, and barely scratches the surface of the potential of using generic and reusable structures to improve the effectiveness and efficiency of system dynamic modelling. In the following, we discuss some of the issues and opportunities for future research.

What is the actual value of using generic and reusable structures?

The general hypothesis driving the research in this topic is that generic and reusable structures provide a useful vehicle to transfer knowledge within and across domains. At the field level, there is some evidence to support this assertion. Generic structures have facilitated the dissemination of system dynamics models into other modelling fields, such as agent-based model and discrete-event simulation. In studies where the research effort focuses on developing a hybrid model, it seems wise and convenient to make use of existing general model, especially if it is well-accepted and widely used. For example, (Schieritz 2003) use Sterman's (2000) production system model to integrate agent-based modelling and system dynamics in supply chain management. The same production system model is integrated with discrete event simulation by (Venkateswaran 2005).

At a modelling project level, there is still a lack of operational understanding of how generic and reusable structures can be effectively used in practice, and the actual value they add to the project. Whereas Lane and Smart (1995) hinted the idea of generic structures as a toolbox which contributes to the different activities in system dynamics (model conceptualization, formal model development, domain understanding, and teaching), subsequent work could not put more flesh on the skeleton. Very few exceptions include Liehr (2001) who made an attempt for proposing an approach to operationalize the use of different generic structures into the formal model development, but was not illustrated by a case study nor followed by further work. Similarly, Hines (2010) proposed a hypothetical scenario for using the modelling by replacement approach.

To address this gap, research into this area need to be supported by case studies which provide transparent understanding of how these structures are employed to support both modelling and teaching activities. Ideally, these case studies need to be intertwined with an empirical assessment approach to evaluate and monitor the utility of using different parts of the toolbox. To promote useful 'fitness-for-purpose' understanding of these approaches, case studies need to focus on:

(1) Questions that influence the modeller's selection and use of an appropriate approach and tools to leverage the value of generic and reusable structures at different project contexts, such as: how does communication at different level of model's abstraction influence the end user's domain understanding and learning about sources of dynamics? For example, the meta-model approach completely hides the complexity of the underlying system dynamics model. Although this harnesses the benefits of rapid model development through model instantiation, it still isolates the user from the conceptualization of the dynamic model. On the other side, modelling by replacement

offers an ‘incremental understanding’ of the model’s structure through replacing substitutable molecules (Yasarcan, 2010).

(2) Questions that influence the modeller’s understanding of the differences in expected learning effects among individuals and groups. For example, what are the effects of using a modular approach at individual (e.g. mental models accuracy) and group levels (e.g. group interaction, quality of information exchange)? What are the differences between invoices and experts on using generic and reusable structures? Kolfschoten et al. (2010) found that the use of modelling building blocks has improved the learning efficiency of invoice modellers, compared to experts who did not seem to trust how the building blocks work.

(3) Questions that influence the modeller’s design of the overall process where the generic and reusable structures are used, such as: what other tools (e.g. documentation) and methods (e.g. professional facilitation) are required and at what point of the modelling process?

What are generic and reusable structure designed?

So far, the work done on using reusable components is limited in scope to high-level discussions about the value of using component-based technologies (e.g. Tignor and Myrtveit, 2000), and technical tutorials of how to build components (e.g. Myrtveit, 2000). There is little understanding of the thinking process underpinning the design of these components and their interfaces, which make it hard to implement the approach. Some of the questions that need to be answered to address this gap are:

- What are the design guidelines for developing reusable structures that provide learning insights about the dynamics of the system (Kasputis and Ng, 2000)? For example, how can the modeller design interfaces between components in a systematic way that allows for meaningful experimentation about information flows?
- What is the process of meaningfully decomposing an existing system dynamics models (e.g. canonical situation model) into a set of building blocks (Balci et al., 2011)?
- How to incorporate end user and modeller requirements into the design of these building blocks (Verbrack et al., 2002)? For example, ‘reusability’ is a broad notion that need to be translated into specific technical requirements that can be incorporated into the system dynamics model at different levels (Sotaquira and Gerly, 2004)
- How can we make use of findings from cognitive science and experimental dynamic decision making literature to inform the design of these structures (e.g. Sterman, 2010)? For example, McLucas and Ryan (2005) used the cognitive complexity index to examine the cognitive load of trying to understand the molecules defined in (Hines 1996). The authors found that molecules’ cognitive complexity index far exceeded human cognitive limitations.

These questions imply the need for a rigorous and systematic modelling process which brings together systems dynamics, systems engineering, and software development (e.g. Mclucas et al., 2010); guided by lessons from developments in the wider simulation literature (e.g. Verbrack and Valentin, 2008).

Emerging opportunities: Exploratory modelling and XMILE

An unexplored promising area is the use of generic structures to support exploratory systems modelling through automatically generating and reconfiguring system dynamics models. Kwakkel and Pruyt (2013) have coined the term of Exploratory System Dynamics Modelling and Analysis (ESDMA) to describe the approach of using system dynamics models to explore a large range of plausible scenarios in deep uncertainty situations, to address questions such as: "How the system may look like under what conditions?". So far, applications are used in 'investigative mode' where the analysis starts with a problem structure. The use of well-designed and validated building blocks opens the opportunity for extending the methodology to allow for generating and testing different model configurations, and therefore exploring plausible structural changes.

Finally, the work on modelling domain building blocks is an investment that can be justified for modellers and the community through reusing beyond small number of projects and case studies (Leach, 2012). One barrier is the different platforms used for implementation. The use of system dynamics standards (XMILE) presents an opportunity for wider adoption and cost-sharing (Eberlein and Chichakly, 2013).

References

Abdelgawad AA., Snaprud MH, Krogstie J. 2011. Accessibility of Norwegian Municipalities Websites: A Decision Support Tool. In *Proceedings of Computer Modeling and Simulation (EMS), Fifth UKSim European Symposium*. IEEE.

Ahmed U. 1997. A process for designing and modelling with components. In *Proceedings of the 1997 International System Dynamics Conference*, Istanbul, Turkey, System Dynamics Society.

Andersen DF, Richardson GP. 1980. Toward a pedagogy of system dynamics. *TIMS Studies in the Management Sciences* **14**: 91–106.

Balci, O, Arthur, JD, Ormsby, WF 2011. Achieving reusability and composability with a simulation conceptual model. *Journal of Simulation*, **5**(3): 157-165.

Barros, MDO, Werner, CML, Travassos, GH. 2001. From metamodels to models: Organizing and reusing domain knowledge in system dynamics model development. In *Proceddings of the 2001 International System Dynamics Conference*. Atlanta, System Dynamics Society

Barros, MDO, Werner, CML, Travassos, GH. 2002. A system dynamics metamodel for software process modeling. *Software Process: Improvement and Practice* **7**(3-4): 161-172.

Bauer C, Bodendorf F. 2005. Component-Based Composition of System Dynamics Models. In *Proceedings 19th European Conference on Modelling and Simulation*.

Dowling AM, MacDonald RH, Richardson GP. 1995. Simulation of Systems Archetypes. In *Proceedings of the 1995 International System Dynamics Conference*, Tokyo, System Dynamics Society.

Eberlein RL, Chichakly KJ. 2013. XMILE: a new standard for system dynamics. *System Dynamics Review* **29**(3): 188-195.

Eberlein RL, Hines JH. 1996. Molecules for modelers. *In Proceedings of 1996 International System Dynamics Conference*, Cambridge, MA, System Dynamics Society.

Forrester JW. 1969. *Urban Dynamics*. MIT Press, Cambridge.

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

Hovmand PS, Ford DN, Flom I, Kyriakakis S. 2009. Victims arrested for domestic violence: Unintended consequences of arrest policies. *System Dynamics Review*, **25**(3): 161-181.

Hines J, Malone T, Gonçalves P, Herman G, Quimby J, Murphy-Hoye M, Rice J, Patten J, Ishii H. 2011. Construction by replacement: a new approach to simulation modelling. *System Dynamics Review* **27**(1): 64-90.

Kolfschoten, G, Lukosch, S, Verbraeck, A, Valentin, E, de Vreede, GJ. 2010 Cognitive learning efficiency through the use of design patterns in teaching. *Computers & Education* **54**(3): 652-660.

Kwakkel JH, Pruyt k. 2013. Using system dynamics for grand challenges: The ESDMA approach. *Systems Research and Behavioral Science*. oi:10.1002/sres.2225

Lane DC. 1998. Can we have confidence in generic structures?. *Journal of the Operational Research Society*, ***49***(9): 936-947.

Lane DC, Smart C. 1996. Reinterpreting 'generic structure': evolution, application and limitations of a concept. *System Dynamics Review* **12** (2): 87-120.

Leach, RJ. 2012. *Software Reuse: Methods, Models, Costs*. AfterMath.

Liehr M, Größler A, Klein M, Milling PM. 2001a. Cycles in the sky: understanding and managing business cycles in the airline market. *System Dynamics Review*, ***17***(4): 311-332.

Liehr M. 2001. Towards a Platform-Strategy for System Dynamics Modeling: Using Generic Structures Hierarchically. In *Proceedings of the 2001 System Dynamics Conference.* Atlanta, System Dynamics Society.

Manataki IE, Zografos KG. 2009. A generic system dynamics based tool for airport terminal performance analysis. *Transportation Research Part C: Emerging Technologies*. ***17***(4): 428-443.

McLucas AC, Ryan MJ. 2005. Combining generic structures and systems engineering to manage complexity in system dynamics modelling. *Journal of Battlefield Technology* **8**(3): 33-40.

McLucas AC, Ryan MJ, Johnston KM. 2010. Where has all the engineering gone?. In *Proceedings of Systems Engineering / Test and Evaluation Conference SETE2010*, Adelaide.

Meadows, DL. 1970. *Dynamics of Commodity Production Cycles*. Boston: Wright-Allen Press.

Myrtveit M. 2000. Object-oriented extensions to system dynamics. In *Proceedings of the 2000 International System Dynamics Conference*, Bergen, Norway, System Dynamics Society

O'Regan BM, Moles R. 2001. An insight into the system dynamics method: a case study in the dynamics of international minerals investment. *Environmental Modelling & Software* **16**(4): 339-350.

Paich M. 1985. Generic structures. *System Dynamics Review* **1** (1): 126-132.

Powers R. 2011. An Object-Oriented approach to managing model complexity. Master thesis. University of Bergen, Norway.

Richardson GH, Pugh A. 1981. Introduction to System Dynamics Modeling with DYNAMO, MIT Press, Cambridge, MA.

Richmond B. (1985). STELLA: software for bringing system dynamics to the other 98%. In *Proceedings of the 1985 2001 System Dynamics Conference.* Keystone, CO, USA, System Dynamics Society

Sabounchi NS, Triantis K, Sarangi S , Liu, S. 2011. Fuzzy Modeling of Linguistic Variables in a System Dynamics Context. In *Proceedings of International System Dynamics Conference,* Washington, DC, System Dynamics Society.

Saeed K, Pavlov OV. 2008. Dynastic cycle: A generic structure describing resource allocation in political economies, markets and firms. *Journal of the Operational Research Society*, **59**(10): 1289-1298.

Schieritz NA, Grobler A. 2003. Emergent structures in supply chains-a study integrating agent-based and system dynamics modeling. In *Proceeding of the the 36th Annual Hawaii International Conference on Systems Science*, IEEE.

Senge PM. 1990. *The Fifth Discipline: The Art and Practice of the Learning Organization*. New York: Doubleday

Sotaquira R, Zabala GC. 2004. Reusability in System Dynamics: Current approaches and improvement opportunities. In *Proceedings of the 2004 International System Dynamics Conference*. Oxford, England, System Dynamics Society.

Sterman JD. 2010. Does formal system dynamics training improve people's understanding of = accumulation. *System Dynamics Review* **26**(4): 316-334.

Sterman JD. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World.* Irwin/McGraw-Hill, Boston.

Tignor WW, Myrtveit M. 2000. Object-oriented Design Patterns and System Dynamics Components. In *Proceedings of the 2004 International System Dynamics Conference.* Bergen, System Dynamics Society

Ulli-Beer S, Gassmann F, Bosshardt M, Wokaun A. 2010. Generic structure to simulate acceptance dynamics. *System Dynamics Review*, 26(**2**): 89-116.

Vavik L, Myrtveit M. 1995. Object based dynamic modelling. In *Proceedings of the 1995 International System Dynamics Conference*, Tokyo, System Dynamics Society.

Venkateswaran J, Son Y. 2005. Hybrid system dynamic-discrete event simulation-based architecture for hierarchical production planning. *International Journal of Production Research* **43**(20): 4397-4429.

Vensim Molecules 2.02 (http://vensim.com/modeling-with-molecules-2-02/#molecules-and-archetypes

Verbraeck A, Valentin. EC. 2008. Design guidelines for simulation building blocks. In *Proceedings of the 40th Conference on Winter Simulation*.

Wolstenholme EF. 2003. Towards the definition and use of a core set of archetypal structures in system dynamics. *System Dynamics Review*, **19**(1): 7-26.

Yasarcan H. 2010. Improving understanding, learning, and performances of novices in dynamic managerial simulation games. *Complexity*, **15**(4): 31-42.