A product trajectory-driven model builder for manufacturing systems

Patrick Charpentier, Andres Véjar

A Product Trajectory-Driven Model Builder for Manufacturing Systems

Patrick Charpentier * **Andrés Véjar** **

* *Research Centre for Automatic Control (CRAN), CNRS (UMR 7039), University of Lorraine, Campus Sciences, BP 70239, 54506 Vandœuvre Cedex, France (tel: + 33 (0)3 83 68 44 23 ; e-mail: patrick.charpentier@univ-lorraine.fr).*

** *Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland (e-mail: andres.vejar@ibspan.waw.pl).*

Abstract: This paper presents an original approach for automatic construction of a simulation model for complex manufacturing systems. The model generation is based on spatiotemporal product trajectories. The products therefore contribute directly to the control of the system. The formal generated model, a queuing network, is a permanent image of the real state of the system to be modelled; it can therefore be described as being auto-adaptive.

Keywords: Data-Driven Application, Modelling, Simulation, Product Trajectories.

1. INTRODUCTION AND GENERAL ISSUES

System modelling and identification is an important field of research in automation. The objective is to obtain a mathematical model of a system based on experimental data and available *a priori* knowledge. This model is intended to provide a faithful approximation of the behaviour of the physical system of interest, to estimate the physical parameters or design simulation, forecasting, monitoring or control algorithms.

The classical approach consists of formalizing *a priori* available knowledge, collecting experimental data, then estimating the structure, the parameters and uncertainties of a model, and finally validating (or invalidating) it (Garnier et al., 2007).

According to the knowledge available or extracted from the process to be identified, one speaks about parametric identification (the structure of the model is known) or about non-parametric identification (the structure of the model is unknown). The research presented in this paper is part of this complex dynamic systems modelling and identification problem; its behaviour cannot be represented accurately by mathematical equations. This kind of modelling is usually very time consuming and depends on the experience and skill of the modeller (Revel et al., 2004). An alternative to this approach is related to the use of machine learning, such as the popular neural networks. In general, these machines do not provide a structure of the model representing the system of interest, and therefore limit future use; see, for example, black-box models (Sjöberg et al., 1995). However, unlike models built by experts, it is possible to use these models online, in the sense that they remain connected to the real system, its inputs and outputs, and are constantly updated. The model can be described as being self-adaptive (Cheng et al., 2009).

This paper therefore presents our proposal to develop automatically a non-parametric model for complex systems (i.e. without the assistance of a human system). Our proposal is based on two complementary approaches:

- A top-down minimalist approach: the minimal knowledge is related to the nature of the system to be modelled. This knowledge will be used as input to build the model generator (e.g. the type of process implemented in the system to be modelled).
- An important bottom-up approach: a continuous collection of spatiotemporal data from the real system is used to feed the generator, and therefore to build, validate and maintain the model.

At first we present briefly the classical approaches for modelling complex systems. A zoom is performed on data collection and the design of the model structure. In a second step, data and knowledge needed to build a flow simulation model are presented. The next section is dedicated to our proposal of product-trajectories modelling by a directed and localized graph. We then describe the gradual construction of the model and its adaptation to changes in the system to be modelled. Some illustrations are presented in Section 4 to show the merits of our approach and of the model-builder. Section 5 concludes this work and presents some possible tracks of it.

2. CLASSICAL APPROACHES FOR MODELLING COMPLEX SYSTEMS

It is fairly standard in the literature (and sometimes in enterprises) to decompose system identification and modelling into phases to facilitate project management and to be able to validate progress step by step. Each of these phases can be supported by methods and tools intended to make more formal and effective progress. The

automation of all these tasks is now difficult to imagine. However, some of them are or can be automated to varying degrees. After finding the relevance of using a simulation tool in relation to a given problem, and contextual or *a priori* knowledge on the system to be modelled, the development of a simulation model requires the following steps:

(1) Data collection and pre-treatment
(2) Development of the model structure (based on a choice of the type of model used)
(3) Instantiation of the model from collected and processed data
(4) Verification and validation of the model.

Steps (1) and (2) are the subjects of the two following subsections. We return to steps (3) and (4) during the presentation of our proposal.

2.1 Data Collection

This important phase is generally time consuming for complex systems modelling (Fowler and Rose, 2004), and may represent up to 50% of the total time taken when building models using conventional approaches (Robertson and Perera, 2002). Depending on the nature of the system under study, collection of data and/or information may be technically achieved in different ways:

(1) By direct measurements performed by human operators or *ad hoc* instruments
(2) In an Enterprise Information System (EIS), where these data and/or information are pre-treated and recorded
(3) By estimations based on theory or similar phenomena already studied.

Data processing is of course dependent on the nature of the data, and also on the mode of collection. Information from an EIS is already formatted (average processing times, for example), while data from the shop floor must be filtered and processed to be used in a simulation model. Finally, this raises issues of the reliability, of the representativeness of the data and/or information used as model parameters, and therefore their maintenance. The most reliable and most representative data are undoubtedly those from the real system in real time obtained by *ad hoc* instrumentation, but they have the disadvantages already presented.

2.2 Design of the Model Structure

Different approaches for partial automation of the development phase of the model structure can be found in the literature. They can be grouped into two families.

The approach using templates (Guru and Savory, 2004) for a progressive construction of the model, on the basis of a generic set of bricks to be parameterized, is probably the one most used in manufacturing systems. One speaks about "generic template-based simulation" or GTS. An important advantage of the GTS approach is linked to the concept of model re-use (Brown and Powers, 2000). Some authors also talk about composable simulations (Page and Opper, 1999; Kasputis and Ng, 2000), where the modelling task can be reduced to a composition of simulation blocks. Conceptual modelling (Zhou et al., 2006), including model-driven engineering, is an alternative to the template approach, but requires significant mental effort (no integral automation but there is methodological support).

3. PROPOSAL

Approaches that generate the model structure by a direct feed from an enterprise resource planning system are rarer (e.g. in the case presented by Koh et al. (2006)). One speaks about "Data-Driven Simulation" (DDS), (Cao et al., 2004). In this approach the model and its structure are automatically generated by a data flow from the system instrumentation or from information systems. The maturity of the concept is due to Darema (2004), who introduced Dynamic Data-Driven Application Systems (DDDAS). Simulation applications must be able to accept new data at runtime and change the measurement process dynamically. For example, Yang (2008) uses the DDS to construct an inventory model. The principles of DDS seem very interesting because they allow the derivation of on-line simulation models, and these models have adaptive structures and adaptive parameters. This approach can also reduce the validation phase of the model, because it is a priori an image of the actual behaviour of the system of interest. The nature of the models depends on the nature of the system of interest (natural or artificial systems, for example). Data-driven modelling and simulation can be compared with the non-parametric identification approaches in automation.

Our goal is to obtain a simulation model flow quickly and easily, representative of the system and its operation at any time, while retaining the possibility of changing the structure and/or parameters for later use. Product localization in time (spatio-temporal data) will constitute for us the minimalist type of input data for the system modelling and identification. More formally, we consider that the generator data inputs consist of a set of location points and each single point is specified by (id, r, t), where:

- id is the object (or product) identifier
- r is the position, generally in the plane
- t is the time.

Our proposal aims to provide a product trajectory-driven model builder for manufacturing systems.

3.1 Data and Knowledge to Build the Model

The construction of a model of a system of interest is performed by a model builder, mainly driven by spatio-temporal trajectories of all products circulating in the system to be modelled. The knowledge necessary for the generation of the model is represented in Fig. 1 through the various links between the model builder, the system to be modelled and the generated model.

The minimalist nature of the information used is linked to a priori knowledge about the system of interest (e.g. it is a manufacturing system). This knowledge will be enriched over time by gradually constructed knowledge that is either operational and behavioural knowledge or singular knowledge (Bourne, 1997). For our target application, these types of knowledge are:

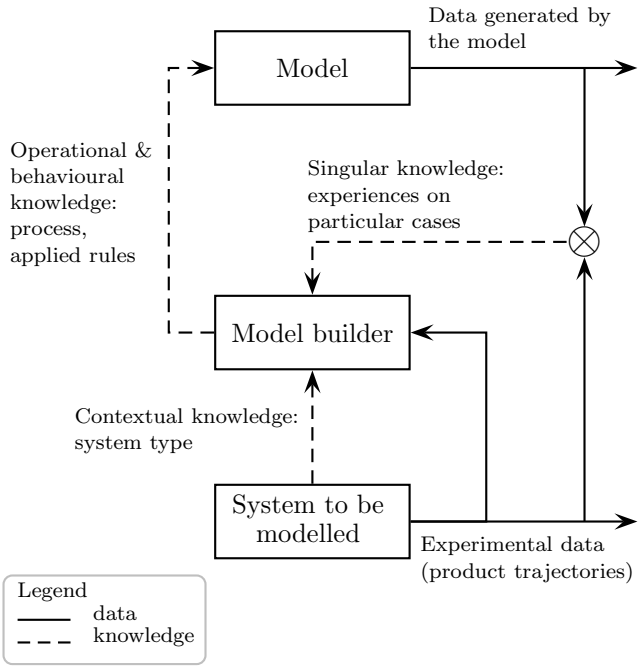

Fig. 1. Data and knowledge flow from/to the model builder

- Contextual or a priori knowledge: this type of knowledge is related to the nature of the complex system studied. In the particular case of the application described in this paper, this kind of knowledge has led us to choose a queuing network model. This type of model is considered representative of phenomena that can occur in the system to be modelled. The basic structural elements of the generated model will therefore be servers and the links between them.
- Operational and behavioural knowledge: unlike contextual knowledge, this knowledge is constructed by a generator based on the spatio-temporal data collected. It represents the structural modelling of the network graph (its typology, the number of servers and links, and the servers' positions). In our particular application, the manufacturing process and bill of materials of the products can also be built using collected data. The model parameterization, via the extraction of information such as the number of nodes (in t) and their relative positions, the time of service (by product and server) and waiting time (by product and server), complement this structural knowledge.
- Singular knowledge (derived from the feedback gained during trials): this knowledge allows a "fit" of the generated model based on specific spatio-temporal characteristics of the system. In our application case, this knowledge type may change the model parameters (spatial dimensions of queues at servers), and also the model structure (order of magnitude of the time constants of the system for a dynamic evolution of the graph; appearance and disappearance of servers).

3.2 Product Trajectory Modelling by a Directed and Localized Graph

During their passage through the manufacturing system, objects follow trajectories in the plane (x, y). The trajectory of the product can be represented by a directed graph where:

- The operations (composition, decomposition, transformation) are represented by nodes
- The transports (or transitions between operations) are represented by directed arcs, where the orientation reflects the precedence relations between operations. Root nodes of the graph represent the entry of raw materials to the system, and the terminal node of the graph represents the final product obtained at the output of the system.

The observation of trajectories over time allows us to determine the speed $\dot{r}_i(t)$ of the object. At each point were $\dot{r}_i(t) = 0$, it is possible to locate specific points to build the model, *e.g.* the buffer operations (or servers) that are associated with the nodes of the graph during the product development. (for more details see Véjar and Charpentier, 2012). These buffer operations (and their locations) will be taken into account if the products stop in the same geographical area with a certain recurrence. The recurrence and the geographical area of the product stops are typically two generator parameters related to singular knowledge of the system (see the previous paragraph).

Each type of product is associated with a directed graph of this type; the entire system can then be defined as the union of all graphs of products. The structure of the complete graph allows us to highlight links between space and time; a position can be specified for each node of the graph (the servers). This graph structure is alive: it will evolve over time to be an image of the system of interest. Figure 2 illustrate this concept with an example of a graph. We intentionally omitted the elements concerning the determination of the different times (transfer, waiting, and service time) for each type of product but they can be consulted at (Véjar and Charpentier, 2012; Véjar, 2011).

3.3 Progressive Construction of the Model and Adaptation

The construction by the generator of the complete graph representing the model of the system as a whole requires several phases. In a simplified manner:

- the progressive construction phase of the model and the updating of data are based on the contextual knowledge and the spatio-temporal data flows;
- the model adaptation is performed thanks to the spatiotemporal data flows, but also with the singular knowledge coming from the feedback from experience.

Algorithm 1: Progressive construction of the model

Data: a location data stream
$(id_0, r_0, t_0)(id_1, r_1, t_1)(id_2, r_2, t_2)\cdots$

Result: a global servers list and a global products list

Set the global servers list S to an empty list
Set the global products list P to an empty list

foreach *location point* (id, r, t) *from the data stream* **do**
  `UpdateGlobals` (id, r, t)
end

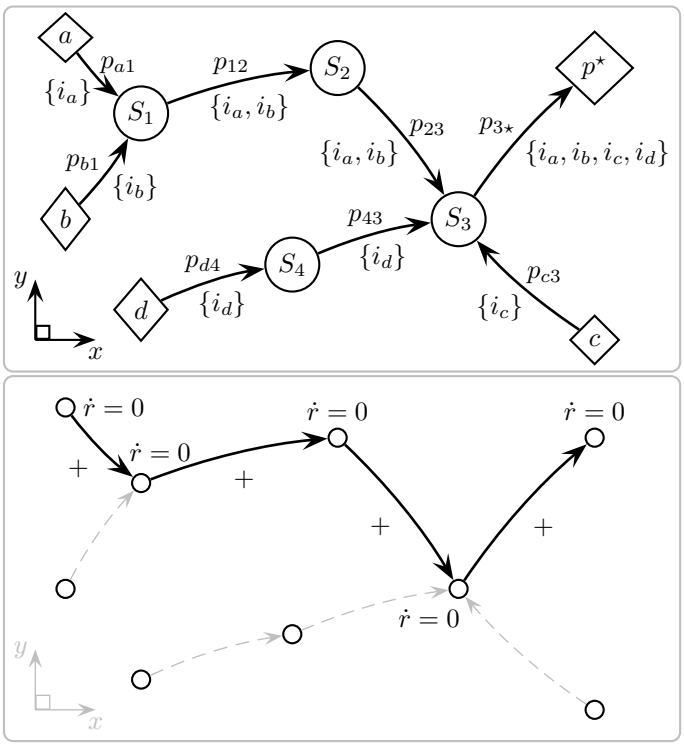

Fig. 2. Example of the directed graph of a product $p^\star$. Top: the diamonds represent the birth or disappearance of objects and/or products. The circles represent operations. Labels associated with arrows represent the product and all associated product identifiers at each level of the process of transformation. Bottom: speed of component a

The generated model aggregates the operational and behavioural knowledge (and parameters) built by the generator. Algorithm 1 and Function UpdateGlobals presents the progressive construction of the model. It is based on the analysis of spatio-temporal product trajectories where the states of motion or stop are differentiated at two time-instants: current time (t) and previous time ($t - dt$).

Updating model data occurs after the copy of the product is definitely out of the system. Two situations can occur: the copy of the product belongs to a product type already referenced; or the product type is new. The discrimination between these two cases is made on the basis of a similarity measure of the spatio-temporal distance between the trajectories of the types of products already known and the output copy of the product. If the product is a new one (high distance), a new type is created, and its characteristics recorded. Otherwise, the data collected during its passage through the module "progressively building a model" are merged with the information already present. They therefore contribute to the evolution of the model parameters. This product data update participates in the adaptation of the model. This adaptation is complemented by a simple mechanism to observe the transient or sustainable aspect of the model elements (products, and therefore servers and related information). It deals with the adaptation of the model to changes over the long term of the information collected.

Function UpdateGlobals(id, r, t)

Data: a location datum (id, r, t), global servers and products lists (S, P)

Result: updated globals lists (S, P)

Check if exists product p in P associated to identifier id

if *does not exists p associated to id* **then**
  Create server s at (position, time) $= (r, t)$
  Create product p associated with id and server s
  Add p to the global products list P
else
  Select product p in P associated with id
  Select server s associated with p
  if *product p was* stopped *at time $(t - dt)$* **then**
    if *server s* position *is r* **then**
      Add $(+dt)$ to stopped_time of s
      Set time of server s to t
    else p is in movement, update last server
      Select s' from list S with the same position of s
      if *p was waiting for service at s'* **then**
        Set service_time of s as the time passed from time_last_output of s' to $(t - dt)$
      else server s' was free when p arrived
        Set service_time of s to stopped_time
      end
      Set time_last_output of s' to $(t - dt)$
      Add s to servers_list of p
      Create server s at (position, time) $= (r, t)$
      Associate server s with p
    end
  else product p was in movement
    if *server s* position *is r* **then**
      Set product p as stopped
      if *does not exists s' from S at* position *r* **then**
        Add s to the global list of servers S
      end
    else product p is in movement
      Set position of server s to r
    end
    Set time of server s to t
  end
end

Each type of product is associated with a "validity function". This function reflects quantitatively the recurrence of the appearance of this product type over time. When this validity function reaches zero, it means that the product type has not appeared in the system for a long time (the time window is again a parameter of the generator): the product must disappear from the model as well as the servers related only to it. This function is incremented if, during the execution of this module, a copy of this type of product was present on the system, else it is decremented. The execution frequency of this module is lower than the frequency of event occurrences in the flow, so as not to unnecessarily slow the execution of the whole.

4. ILLUSTRATIONS

To illustrate the operation of the generator and validate the approach, we propose now two types of results. The

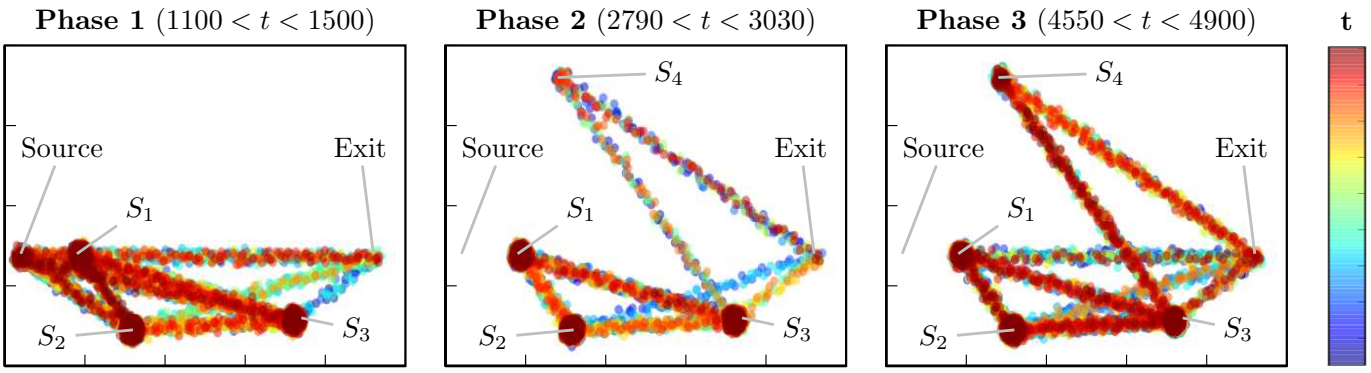

Fig. 4. Model adaptation. Phase 1: The system consists of three servers. Phase 2: Emergence of a new server. Phase 3: The new server is validated.

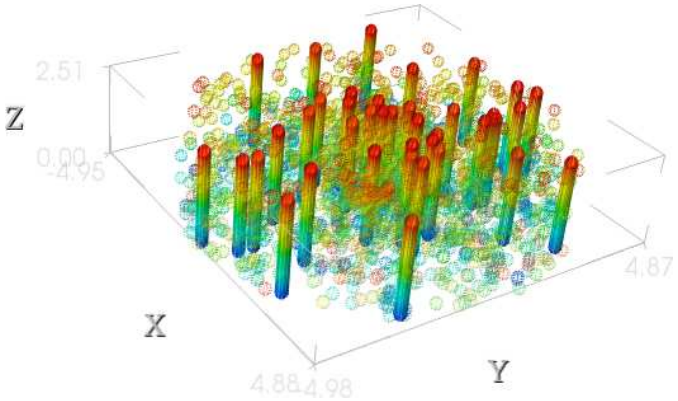

Fig. 3. Spatio-temporal view of a manufacturing system with 40 machines.

first is an illustration of the spatio-temporal analysis. Fig. 3 shows a representation in which x and y represent the two-dimensional space of a system of interest, and the z-axis represents time. The space areas where the products pass the most time are represented by the peaks along the z-axis. These high-density areas correspond naturally to the identified positions of the servers. This image is a snapshot reflecting the structure of the model at a particular moment.

To illustrate the structural adjustment of the model, in Fig. 4 we present three different states of the same system for three different periods. In the scenario, a new product type different from those existing previously appears. Those figures shows the system status before, during, and after the introduction of this new product. Time is represented by the colours on this figure. One can see the gradual emergence of a new server (top of Fig. 4, Phase 2), not initially present in the system. A video of this adaptation would be more representative. Other situations (such as machine disappearance, changes of locations, and new products) could be represented. The phenomena observed and measured all confirm the adaptability of the model to changes in the real system.

5. CONCLUSIONS

Our proposal constitutes an original alternative for the modelling of complex and evolving systems. We here focus on manufacturing systems because we have some expertise in these systems and they are certainly easier to instrument than other types. The proposed approach mainly uses spatio-temporal product trajectories (bottom-up approach) and some a priori or singular knowledge. The construction of the model is formal and is based on directed and localized graphs. It presupposes that objects' movements are observed and generate (directly from their own instrumentation, or indirectly through instrumentation of the environment) information on their trajectories. As such, they contribute to the control of their environment: the system is partially controlled by the products through the model the products help to generate. This model is of the "grey box" type. Unlike models based on neural networks or other technologies, it is, for example, possible to access the graphs and to change or reconfigure them for tests such as "what-if". This model also has the great advantage of being synchronized with reality: it continually reflects this reality and follows the evolution of the real system. We have validated our approach with several test cases and presented the results visually.

At this stage of development, different complementary approaches to continuing this work can be considered. The first is to check the feasibility of identifying system products, either through an identifier associated with them, or through their spatio-temporal trajectories. This could be some signature for each type of product. The second approach, based on the spatio-temporal information of each product item, could involve the determination of the priority rules implemented on the system of interest. We could then construct additional knowledge that could be included in the model. Finally, the application to non-manufacturing cases (including complex systems such as human actions) would probably be a very interesting challenge.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the financial support of the CPER 2007-2013 "Structuration du Pôle de Compétitivité Fibres Grand'Est" (Competitiveness Fibre Cluster), through local (Conseil Général des Vosges), regional (Région Lorraine), national (DRRT and FNADT) and European (FEDER) funds.

The study is cofounded by the European Union from resources of the European Social Fund. Project PO KL "Information Technologies: Research and their interdis-

ciplinary applications", Agreement UDA-POKL.04.01.01-00-051/10-00.

REFERENCES

Bourne, C. (1997). Catégorisation et formalisation des connaissances industrielles. In J.M. Fouet (ed.), *Connaissances et Savoir-faire en entreprise*, 179–197. Hermès, Paris.

Brown, N. and Powers, S. (2000). Simulation in a box: a generic reusable maintenance model. In *Proceedings of the 32nd conference on Winter simulation*, WSC '00, 1050–1056. Society for Computer Simulation International, San Diego, CA, USA.

Cao, B., Farr, R., Byrne, M., and Tannock, J. (2004). Data-driven Simulation of the Extended Enterprise. *18th International Conference on Production Research*.

Cheng, B.H.C., de Lemos, R., Giese, H., Inverardi, P., and Magee, J. (2009). Software Engineering for Self-Adaptive Systems. *Lecture Notes in Computer Science*, 5525. doi:10.1007/978-3-642-02161-9.

Darema, F. (2004). Dynamic data driven applications systems: A new paradigm for application simulations and measurements. In M. Bubak, G. Albada, P. Sloot, and J. Dongarra (eds.), *Computational Science - ICCS 2004*, volume 3038 of *Lecture Notes in Computer Science*, 662–669. Springer Berlin Heidelberg. doi:10.1007/978-3-540-24688-6_86.

Fowler, J.W. and Rose, O. (2004). Grand challenges in modeling and simulation of complex manufacturing systems. *SIMULATION*, 80(9), 469–476. doi:10.1177/0037549704044324.

Garnier, H., Gilson, M., Bastogne, T., and Richard, A. (2007). Identification de modèles paramétriques à temps continu à partir de données expérimentales. *Techniques de l'Ingenieur*, S7 140.

Guru, A. and Savory, P. (2004). A template-based conceptual modeling infrastructure for simulation of physical security systems. In *Proceedings of the 36th conference on Winter simulation*, WSC '04, 866–873. Winter Simulation Conference.

Kasputis, S. and Ng, H.C. (2000). Model composability: formulating a research thrust: composable simulations. In *Proceedings of the 32nd conference on Winter simulation*, WSC '00, 1577–1584. Society for Computer Simulation International, San Diego, CA, USA.

Koh, S., Gunasekaran, A., and Saad, S. (2006). Parts verification for multi-level-dependent demand manufacturing systems: a recognition and classification structure. *The International Journal of Advanced Manufacturing Technology*, 31, 305–315. 10.1007/s00170-005-0184-9.

Page, E.H. and Opper, J.M. (1999). Observations on the complexity of composable simulation. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future - Volume 1*, WSC '99, 553–560. ACM, New York, NY, USA. doi:10.1145/324138.324433.

Revel, L., Habchi, G., and Maire, J. (2004). Analyse du processus d'élaboration d'un projet de simulation. In *5ème Conf. Int. Francophone de MOdélisation et SIMulation*, volume 1, 293–301. Nantes, France.

Robertson, N. and Perera, T. (2002). Automated data collection for simulation? *Simulation Practice and Theory*, 9(6), 349–364. doi:10.1016/S0928-4869(01)00055-6.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P.Y., Hjalmarsson, H., and Juditsky, A. (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12), 1691–1724. doi:10.1016/0005-1098(95)00120-8.

Véjar, A. (2011). *Génération de modèles de simulation adaptatifs, pilotée par les trajectoires produits*. Thése de Doctorat, Université Henri Poincaré - Nancy I.

Véjar, A. and Charpentier, P. (2012). Generation of an adaptive simulation driven by product trajectories. *Journal of Intelligent Manufacturing*, 23, 2667–2679. doi:10.1007/s10845-011-0504-x.

Yang, M. (2008). Using data driven simulation to build inventory model. In *Proceedings of the 40th Conference on Winter Simulation*, WSC '08, 2595–2599. Winter Simulation Conference.

Zhou, M., Zhang, Q., and Chen, Z. (2006). What can be done to automate conceptual simulation modeling? In *Proceedings of the 38th conference on Winter simulation*, WSC '06, 809–814. Winter Simulation Conference.