Proposal of an Approach to Automate the Generation of a Transitic System's Observer and Decision Support using MDE

Mickaël Adam, Olivier Cardin, Pascal Berruet, Pierre Castagna

Proposal of an Approach to Automate the Generation of a Transitic System's Observer and Decision Support using Model Driven Engineering.

M. Adam*, O. Cardin*, P. Berruet**
P. Castagna*

** Institut de Recherche en Communications et Cybernétique de Nantes, Nantes France (e-mail: mickael.adam/olivier.cardin/pierre.castagna@univ-nantes.fr).*
*** Laboratoire en sciences et technologies de l'information, de la communication et de la connaissance, Lorient, France (e-mail: pascal.berruet@univ-ubs.fr).*

Abstract: Short term decision support for manufacturing systems is generally difficult because of the initial data needed by the calculations. Previous works suggest the use of a discrete event observer in order to retrieve these data from a virtual copy of the workshop, as up to date as possible at any time. This proposal offered many perspectives, but suffers from the difficulties to generate a decision support tool combining decision calculations and observation. Meanwhile, interesting developments were made in literature about automatic generation of logic control programs for those same manufacturing systems, especially using the Model Driven Engineering. This paper suggests the use of Model Driven Engineering to generate logic control programs, the observer and the decision support tool at the same time, based on the same data collected by the designer of the system. Thus, the last section presents the evolution needed in the initial data structure, as well as the conception flow suggested to automatize the generation.

1. INTRODUCTION

Because flexibility and performance are nowadays major qualities required for manufacturing systems, their complexity tends to rise. As a matter of fact, Decision Support Systems are used to make the decisions as efficiently as possible. It is possible to differentiate two types of decision support tools, the "off-line", for the long term decision support, and the "online" ones, for short term.

This paper is based on the hypothesis that a decision is needed to be computed when a disruption arrives. (Baillet *et al*., 1993) define a disruption like an unpredictable event, which troubles the objectives that have to be achieved. Five types of disruptions were defined, dealing with internal resources availability, supply issues, demand prediction, information and decision.

In addition, a list of typologies of short term decision is presented in (Pierreval, 1999). Among those, the "Affectation decision" deals with the fact that several resources sharing one or several operators are available to perform one task. The "Attribution decision" happens when one or several resources have the choice between several tasks to execute. The "Questioning decision" appears when a disruption which compromises the production arrives. Others types of decision are related, dealing with other types of disruptions or resources. The operator has to reorganize to include the perturbation.

When a disturbance happens on the manufacturing system, a decision needs to be taken. For example, when a new fabrication order is placed, the operator has to decide the parameters of the order in order to optimize its execution. The first problem he encounters is to consider the influence of his decision on the entire system (De Vin *et al.,* 2004), as this decision may change the behavior of the other order.

The second one is that this decision has to be taken quickly, because the system evolves in an uncorrected state during the decision computation. In order to help the operator, several decision support systems have been developed. One may cite for example Knowledge Based Systems (KBS), Fuzzy set, Analytical Hierarchy Processes (AHP), Multi-Attribute Utility Theory (MAUT), Case-Based Reasoning (CBR), Artificial Neural Networks (ANN), goal programming and online discrete event simulation.

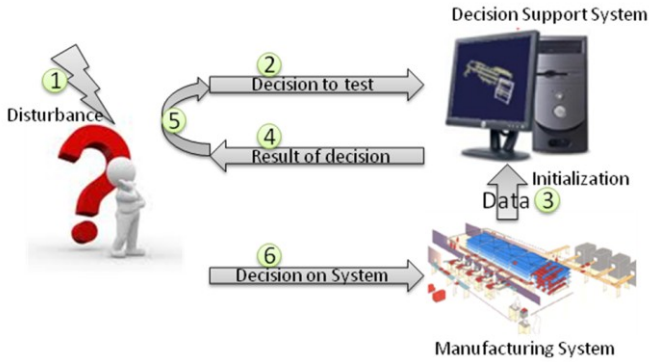

Fig. 1. Classical behavior of decision support systems

Generally, decision support systems have the same behavior (Fig. 1). Their role is generally to help the operator in charge of production activity control of the system to make a decision when a disturbance happens (1). A set of possible decisions (2) is then tested or calculated. This calculation is based on

the data coming from the real system about its state, usually used for initialization (3). Once the decision impact was calculated and the results returned to the operator (4), he has the possibility to discard these proposals in order to make a new calculation (5). Finally, when the decision seems good, it can be applied on the real system (6).

These methods generally need accurate data directly coming from the real system to initialize the computation. For example, (El-Bouri *et al.,* 2006) use an Artificial Neural Network as decision support system, applied on a job-shop to select the best dispatching rule to be applied on each machine. Without speaking of the learning phase, where lots of data collected from the real system are used, this method needs a vector at the entrance of the network which is made of several key data, representing the actual state of the system as accurately as possible.

In the same idea, online simulation was defined by (Davis *et al.*, 1998) as being a discrete event simulation constantly connected to the real system, and initialized by the state of this system when supervisor wants to simulate a short term decision. This method results in a good prediction ability as the modeling power of discrete event simulation is relatively high.

The issue is that these data are not always available, either because they are not measured or not measurable. Not measured data often deal with the position of physical objects on the system. Solutions do exist, like for example the use of long focused Radio Frequency Identification (RFID) (Hotz *et al.,* 2006), but they have the disadvantage of cost and do not always enable a sufficient precision. Furthermore, these solutions do not give a solution for not measurable data, like for example the data included in an electronic tag on a transporter.

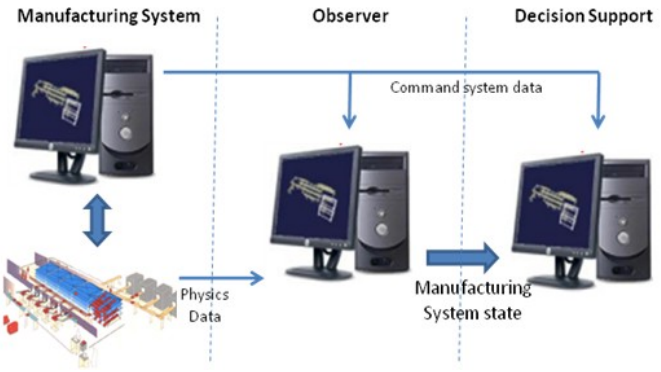

Fig. 2. Decision support proposal from (Cardin, 2007) based on online simulation

(Cardin, 2007) dealt with this kind of issues in the case of online simulation. He suggested the use of a module which would be able to reconstruct the missing data. This module is always connected to the manufacturing system, scans known states and data from the real system and reconstructs in real time the missing data the online simulator needs. This module, called an observer of the discrete event system, does the link between the real system and the online simulation (Fig. 2) as being a virtual copy of the real system, preformatted for the initialization of online simulator as being developed with a discrete event simulation formalism itself.

This approach was validated on a real manufacturing system. This validation showed good prediction abilities from the online simulator, with the reactivity needed thanks to the observer. However some problems were still raised during development phase.

The main problem for the use of such an observer is the complexity of the development phase. This is the main obstacle of its use in a wide industrial context. Indeed, the difficulty is to perform the link between logic control of the manufacturing system and the numerous observer's data needed. Same kind of problem was already raised by (Gonzalez *et al.,* 1998).

Furthermore, the design of the observer requires being an expert of several formalisms, as the designer has to be able to link the control of the real system with its modeling of the observer and the online simulator.

All these problems dramatically increase the time needed for the development of online simulation in a production environment. As a matter of fact, this paper presents a development solution meant to solve these issues, by automating some steps of the development, especially those in relationship with the real system.

To be able to benchmark the solution, this work is based on (Cardin, 2007), who uses an observer made with the discrete event simulator Arena. Hence, the observer developed here will also be made of discrete event simulation. First part of this paper presents in detail the observer of (Cardin, 2007). Then a state of the art of automatic generation of simulation models is presented, followed by a state of the art on logic control automatic generation. Finally we will discuss about the best way to use the generation approaches presented before to generate our observer.

2. THE OBSERVER

The observer is essential element in the decision support system established by (Cardin, 2007), enabling this system to have access to any data of the real system. In this case, it was used on the particular case of online simulation for the production activity control of a manufacturing system.

The observer, meant to be able to give a complete picture of the manufacturing system at any time, can be split into three aspects:

1. It must be able to retrieve the data from the operating system.
2. It must be able to reconstruct all needed unknown data.
3. It must be able to inform (i.e. transmit data) the decision support system as quickly as possible.

The choice of technology used to program the observer is determined by these three aspects.

2.1 Data Collection Aspect

To be able to retrieve data from the real system, the observer needs to be constantly connected to the system. (Cardin, 2007) states that the observer should not influence the control logic, which is generally constituted of the only variables needed for control. This paper stands on the same hypothesis, and considers such manufacturing systems, where the control logic is implemented on Programmable logic controller (PLC), linked through the OPC standard to the observer.

2.2 Reconstruction Aspect

An example of data reconstruction is the case of a conveyor system with two sensors, one at the entrance and one at the exit. If the exact products' positions on the conveyor are needed, the control logic only detects the position in front of the sensors. The role of the observer is thus to determine where the product is when it is not in front of a sensor.

Of course, deviation may happen. Due to an incorrect evaluation of some of these parameters, there might be a sensible difference between the real position of the products on the conveyor and its estimation in the observer. As a matter of fact, the observer must be able to synchronize its evolution with the evolution of the real system. Figure 3 presents the concept of synchronization applied to the case of the conveyor: At t=t1, the observer estimates the product in front of sensor1 whereas the real one is not arrived yet. Then the observer stops the transporter in that position until the real sensor s1 detects the real product. On the opposite, at t=t2, the real product is in front of sensor s2 whereas the observer displays the product far before sensor s2. The real system is ahead the observer. The tracking will instantaneously put the estimated product in the right position. By further analysis, a detection of systematic synchronization for the same reason could be used to reveal either a problem on the modeling of the observer or a deviance of the real system's behavior.

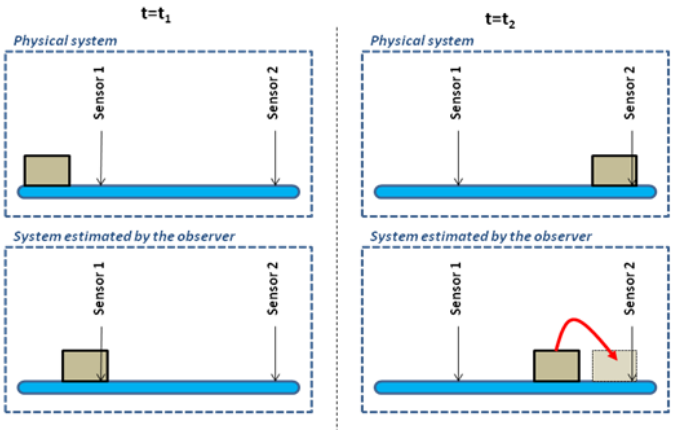

Fig. 3. Synchronization issue: the example of a conveyor

2.3 State Restitution Aspect

The state of the observer is designed to be, at any time, the actual state of the real system. Obviously, some differences exist, as the modeling is necessarily not perfect. However, synchronization mechanisms keep the difference between the observer and the real system at an acceptable level.

The link with the decision support system is different according to the system. For example, ANN only need an entrance vector, whereas online simulations need the state of every simulation primitives of the observer.

An important feature of decision systems is the delay between the request for a decision and the result of the computation. In the context of online decision, this delay is obviously short, as the system keeps on evolving during the computation. Thus, the results of this computation might not be valid any more. To permit this reactivity, the data structure of the observer must be close to the data structure of the decision support system, which is generally sensibly different from the data structure of the control, main source of data for the observer. The time objective leads to the conclusion that the conversion between the data structure of control and the data structure of the decision support system should be included in the design of the observer: even if it makes the design of the observer more complex, the data transfer to the decision support system is made a lot easier and faster.

2.4 Problematic

The main problem with the observer to be a realist solution is that it is too much complex to develop: the data link between the real system and the observer is difficult to establish due to the large amount of data to take into account on the real system. Furthermore, to be efficient, the designer has to be competent at a time in discrete event simulation to design the observer, industrial network to establish the link with the control and logic control programming, in order to be able to identify the data of the control with the data of the observer.

3. AUTOMATIC GENERATION TECHNIQUES

3.1 Automatic generation of discrete-event simulation models

Automatic generation of simulation models is generally performed in two steps: data organization and then call a generation routine with these data as arguments.

To organize the data, several possibilities were explored. (Gong *et al.*, 1990) use text files to represent a manufacturing system with Automated Guided Vehicles for transfers. In (Young, 2000), the use of relational database enables to generate a Flow Shop simulation for Arena and ProModel. (El Haouzi *et al.*, 2007) uses jointly Microsoft Excel and a UML representation to represent a Demand Flow Technology (DFT)manufacturing system.

Along the years, there has been a clear evolution on the method used to order the data, linked to the development of new formalisms. This evolution was guided by the need for a better abstraction of the data organization for manufacturing systems of growing complexity.

Almost every work on automatic simulation models generation states that the use of libraries in the generation routine is a good solution (Kang, 1997), (Young *et al.,* 2000), (El Haouzi *et al.,* 2007), (Mueller *et al.,* 2007), (De Miguel *et al.,* 2000). Libraries are models of atomic elements of the manufacturing systems, accepting arguments. These atomic models are instantiated, adapted (through the arguments) and

linked with each other according to the data presented before. The terminology of libraries or template may differ in some papers (Young *et al.*, 2000), but the principle stays the same. The use of libraries permits to facilitate the generation step, but also permits to capitalize the atomic models, usually called components.

Generation routine is generally homemade. Several techniques are used to interpret the data: for example parsers to transform XML or text files, or models transformation techniques coming from Model Driven Engineering (MDE) theory for UML descriptions. The MDE is a methodoly for software development which permit to create and transform domain models tanks to tools, concepts and languages.

3.2 Automatic generation of logic control programs

The singularity of discrete-event observer is its permanent connection to the real system, and in particular to its command. As a matter of fact, next section also reviews techniques and methodologies to generate control models for manufacturing systems.

Most papers focus on automatic generation of PLC code, and especially IEC 61131-3 code generation. (Vogel-Heuser *et al.,* 2005) deals with Structured Text (ST), (Chiron, 2007), (Estévez *et al.,* 2007) generate Function Block Diagrams (FBD). (Devinder *et al.,* 2009) proposes Program Organization Units (POU). The flow proposed in (Lallican *et al.,* 2007) generates Sequential Function Chart (SFC). All the methods are based on the same previously mentioned tools: parser and transformation models.

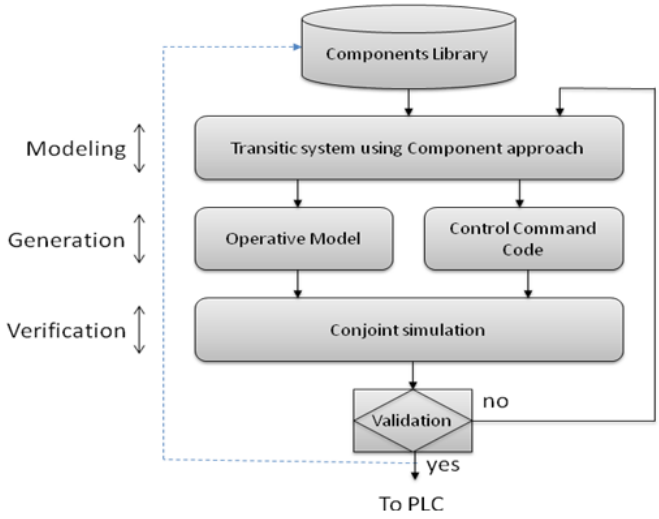

Fig. 4. Component approach from (Lallican *et al.*, 2007)

(Lallican *et al.*, 2007) and (berruet *et al.*, 2007) work use a component based approach to generate PLC code for transitic system. The approach (Fig. 4) is split into three parts.

On the first step, the model of a transitic system is built using a specific language and the component concept. The second step consists in generating both an operative model and the control logic code, IEC61131-3 compliant. The operative model enables on the third step to check the generated PLC code. If the verification is validated, the code can be implemented on PLC; otherwise the model from the first step has to be modified.

3.3 Overview

From this review of automatic models generation, we noticed that the techniques include more and more UML based modeling. The use of libraries is also an interesting point as it provides reusability.

Moreover, (Lallican *et al.*, 2007) approach introduces the component approach to be as close as possible to the control. This specificity might permit to solve the observer generation problem. Thus, the presented approach is adapted from Lallican's, and we first delineate it.

To model transitic systems, (Lallican *et al.*, 2007) has proposed a formal language following MDE requirements. This language enables to describe system from a library of components. Components model a part of the system (e.g. conveyors, sensors, jack, PLC). Aggregation provides a bottom up approach. Components are composed of views, which classify pieces of information. The operative view physically characterizes the component, the control logic view expresses the logic of the component, the constraint view describes security/safety behavior constraint and the topological view characterizes locations and areas of the component. After the model edition, different model transformations enable to generate the logic control and add it to a target model. This step is performed using the ATL language. The logic control can be imported on an IEC 61131-3 editor and can be simulated with the operative model before on-site implementation. The proposal also provides tools for this.

4. THE PROPOSED FLOW

The main goal is to solve the observer implementation issues, essentially due to the amount of engineering needed for development and to the link between the real system, the observer, and the decision support which is rather difficult to establish. To do so, the main idea of this work is to generate automatically the observer, in parallel with the control logic and a part of the decision support. However, an observer as defined previously is not useful without an associated decision support tool. As a matter of fact, the automatic generation should also deal with this decision support tool.

As presented in the state of the art, the work of (Lallican *et al.*, 2007) seems very interesting in this orientation thanks to the developments that were shown, and thus will be used in this paper: the method presented already generates in parallel the control logic and a simulator. Furthermore, the genericity of the tools that were used, like UML and ATL, makes it easily adaptable. The method is based on MDE, which is an ensemble of concepts and tools, which permit to model and to transform models.

The approach of automatic generation of the observer using Lallican's methodology presents several benefits. First, the component approach enables reusability of atomic models, very convenient for the development of similar applications. Second benefit is the opportunity to link the data of the

observer with the data of the logic control during the creation of the logic control programs.

However, the meta-models transformations used by (Lallican *et al.*, 2007) have to be adapted. Indeed, some changes on the meta-models themselves, and on the transformations have to be done in order to integrate additional information about the observer and the decision support system. This additional information depends on the features that the observer and the decision support tool have to provide. Once again, the component approach is very convenient thanks to the concept of views. These views contain all the information needed for a specific aspect of an element. For example, warning the operator about a significant difference between the real system's state and the associated prediction is simply performed by adding a new view including warning time delays information in the meta-model.

After dealing with the meta-models, second aspect of this work is to define the correct transformations. To do so, it is relevant to define the target meta-models, one for the observer and one for the decision support tool. On (Lallican *et al.*, 2007) work, the target meta-model is IEC 61131-3 compliant. The new target meta-models are oriented to discrete-event simulation, with the objective to demonstrate the feasibility of the approach with online simulation decision support.

The development phase is relatively complex. To ease the understanding, two types of users, defined by (Chiron, 2007), are considered: Modelers and Designers. A modeler develops models, components, libraries, meta-model transformations and an Integrated Development Environment (IDE), who enables the generation of the observer and of the decision supports. The designer instantiate the modeler's work in order to model his specific transitic system, and then call the transformations programmed by the modeler in order to obtain the desired model.

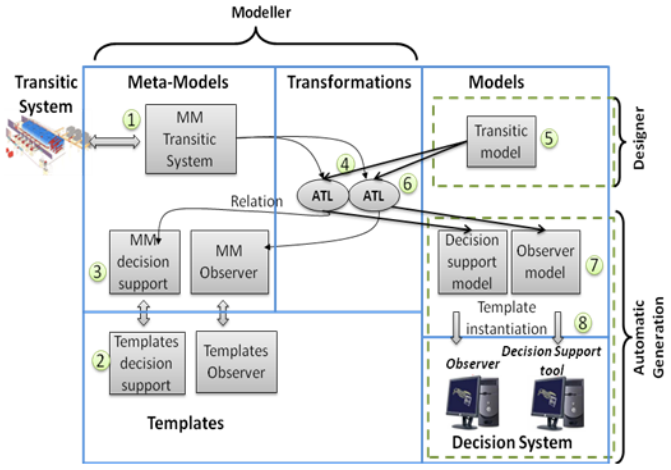

Fig. 5. Proposed flow of automatic generation

The general flow is represented in (Fig. 5). First (1), the modeler has to define the types of components which will be found on the transitic system. These components take the same specifications, views, hierarchy and typology (Fig. 6) than the components which were defined by (Lallican *et al.*, 2007). Every system has to be represented by components in this approach. The next step is to create the Meta-models of these components of the transitic system.

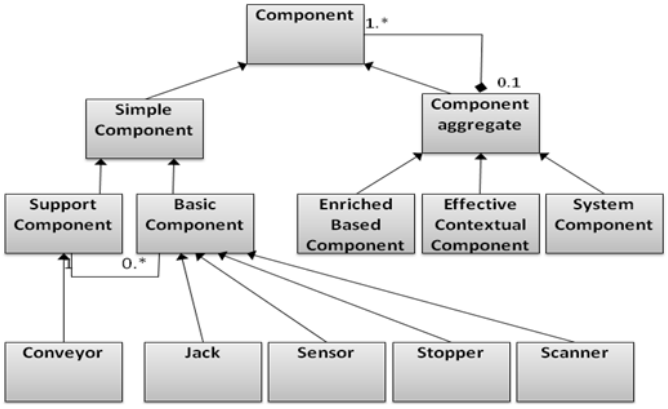

Fig. 6. Component typology meta-model

When this is done, the modeler creates the templates in the final language (i.e. SIMAN Arena in this case) of the observer, and the templates needed to build the decision support tool (2). Obviously, these templates must match the meta-models of the transitic system. Next meta-models corresponding to these templates have to be created (3).

At this step, transformations can be programmed in ATL (4). A part of the Integrated Development Environment (IDE) is next automatically generated thanks to MDE tools. The modeler integrates previous transformations on this IDE, permitting to generate the observer model and decision support model corresponding to their respective meta-models. All these meta-models are placed in a library, on the IDE, so that it can be used by the designer.

The IDE permits the designer to edit a transitic model corresponding to the transitic Meta- model (5) representing the real system. He sets up all components, their views, and the relationships between the components and the control logic. ATL transformations are applied on this model (6) to generate decision support model and observer model (7). Both these models are imported by template instantiation (8) created before by the modeler.

It has to be noticed that this approach does not generate a complete decision support tools ready to use, but only a basis mostly able to do the link with the observer. This facilitates the creation of a fully functional decision support tool. The conception and the validation of the tools are the last task of the designer.

5. CONCLUSION

Short term decision support for manufacturing systems is generally difficult because of the initial data needed by the calculations. (Cardin, 2007) suggested to use a discrete event observer in order to retrieve these data from a virtual copy of the workshop, as up to date as possible at any time. This proposal offered many perspectives, but suffers from the difficulties to generate a decision support tool combining decision calculations and observation, especially about the

necessary connection with the data of the manufacturing system.

Meanwhile, interesting developments were made in literature about automatic generation of logic control programs for those same manufacturing systems. The literature review exposed in the second section shows that works using MDE (Berruet *et al.*, 2007) (Lallican *et al.*, 2007) can be particularly interesting for the generation of the decision support tool.

Last section of this paper presents the adaptation of the initial data structure of (Lallican *et al.*, 2007) needed for implementing the observer, and the suggested conception flow.

Future work deals with the implementation of the framework, creation of new components, views, template, models and transformations with the associated IDE. The flow will then be tested on a real transitic system to validate the approach and the framework.

REFERENCES

Baillet, Couvreur, Cauvin, (1993). Disturbance classification for reactive scheduling Production research. In: *Proceedings of the 12th International Conference on Production Research*, Lappeenranta, Finland, pp.16-20.

Berruet, P., J.L. Lallican, A. Rossi, J.L. Philippe, (2007). "Generation of control for conveying systems based on component approach", *IEEE SMC 2007*, pp.1408-1414, Montréal.

Cardin, O.(2007). *Contribution of online simulation to production activity control decision support - application to a flexible manufacture system*. Phd Thesis, Université de Nantes, Nantes

Chiron, F. (2007). *Contribution à la flexibilité Contribution à la flexibilité et à la rapidité de conception des systèmes et à la rapidité de conception des systèmes automatisés avec l'utilisation d'UML.* PhD Thesis, Université Blaise Pascal de Clermont-Ferrand.

Davis, Wayne J. (1998). Online simulation: Need and evolving research requirements. In: *Handbook of Simulation, de Jerry Banks*, pp.465-516.

De Miguel, M., T. Lambolais, S. Piekarec, S. Betgé-Brezetz, J. Péquery, W. Emmerich and S. Tai (2001). Automatic Generation of Simulation Models for the Evaluation of Perfomance and Reliability of Architectures Specified in UML. In: *EDO 2000, LNCS 1999*, pp. 83-101, 2001. *Springer-Verlag Berlin Heidelberg*

De Vin, L.J., H.C. Amos, Ng and J. Oscarsson (2004). Simulation-Based Decision Support for Manufacturing System Life Cycle Management. In: *Journal of Advanced Manufacturing Systems*, vol. **3**, pp.115-128

Devinder, Thada, C.M. Park, S.C. Park, G.N. Wang (2009). Auto-generation of IEC Standard PLC Code Using t-MPSG. In: *International Journal of Control, Automation and Systems*, Vol**. 7**, N. 2, pp.165-174.

Drake, Glenn R., and Jeffrey S. Smith. (1996). Simulation system for real-time planning, scheduling, and control. *28th conference on Winter simulation*. Coronado In: *ACM Press*, pp.1083 - 1090.

El-Bouri, A. and Shah P., (2006), A neural network for dispatching rule selection in a job shop, *International Journal of Advanced Manufacturing Technology*, Vol. **31**, N. 3-4, pp.342-349.

El Haouzi, H., R. Pannequin And A . Thomas (2007). Génération automatique de plateformes de simulation pour des systèmes organisés en flux tirés, *7e Congrès International de Génie Industriel*, Trois Rivières, Canada

Estévez, E., M. Marcos, et al. (2007). Automatic generation of PLC automation projects from component-based models. In: *The International Journal of Advanced Manufacturing Technology,* Vol. **35**, N. **5-6**, pp.527-540.

Gong, D-C., L.F. McGinnis. (1990). An AGVS Simulation Code Generator for Manufacturing Applications. *22nd Winter Simulation Conference*. *IEEE Press* Piscataway, NJ, USA, pp.676-682.

Gonzalez, Fernando G., et Wayne J. Davis. (1998). Initializing on-line simulations from the state of a distributed system. in: *30th Winter simulation*. Washington, D.C. In: *IEEE Computer Society Press*, pp.507-514.

Hotz, I., A. Hanisch, and T. Schulze. (2006). Simulation-based early warning systems as a practical approach for the automotive industry. *37th conference on Winter simulation.* Monterey, California, pp.1962-1970.

Kang, S., (1997). Knowledge based automatic simulation model generation system, In: *IEE Proc-Circuits Devices Syst*., Vol. **144**, No 2, pp.88-96.

Lallican, J.L, P. Berruet, A. Rossi, J-L. Philippe, (2007). A component-based approach for conveying systems control design, *IFAC ICINCO,* Angers, May 9-12, pp.329-336.

Mueller, R., C. Alexopoulos, L.F. McGinnis, (2007). Automatic Generation of Simulation Models for Semiconductor Manufactoring. *39th Winter Simulation Conference,* IEEE Press Piscataway, NJ, USA, pp.648-657

Pierreval, H., (1999). Proposition de typologie des décisions en temps réel agissant sur les flux des systèmes de production, *MOSIM*, Annecy.

Vogel-Heuser, B., D. Witsch, et al. (2005). Automatic Code Generation from a UML model to IEC 61131-3 and system configuration tools. *International Conference on Control and Automation,* pp. 1034-1039.

Young, J.S., Albert T. Jones, Richard A. Wysh. (2000). Automatic Generation of simulation models from neutral libraries: an example. *Winter Simulation Conference Proceeding*, U.S.A, vol.**2**, pp.1558-1567.