A formal meta-modeling approach for the design of automated manufacturing systems

Laurent Piétrac[1], Bruno Denis[2] and Jean-Jacques Lesage[2]

[1] LAI, INSA de Lyon, Bat 303
20, avenue Albert Einstein F-69621 VILLEURBANNE CEDEX - FRANCE
Laurent.Pietrac@lai.insa-lyon.fr
[2] LURPA, ENS de Cachan
61, avevue du Président Wilson F-94235 CACHAN CEDEX - FRANCE
{Bruno.Denis,Jean-Jacques.Lesage}@lurpa.ens-cachan.fr

Abstract. The design of Automated Manufacturing Systems (AMS) consists in the construction of many models. The quality of the designed system depends on the quality of the produced models, and also on the quality of the languages and methods used during the design. Some recent works aim to improve these languages and these methods by using meta-modeling. The purpose of this paper is to make an inventory of the requirements in meta-modeling and to propose a language of meta-modeling enabling to meet these requirements.

1 Introduction

The AMS are increasingly complex systems, whose design requires increasingly diversified techniques. Moreover, an enhanced quality and an increased safety in operation are required for these systems. These requirements are reflected on the process of AMS design, whose various stages are punctuated by the contruction of models.

The development of models requires a certain number of concepts, the most important of which we are recalling here (in bold). Each model is built starting from a modeling **language**, which is sometimes accompanied by a **method of construction**. Then the model can be used as a vector of communication; it has to respect **the syntax and the semantics** of the language. In addition to a method of construction, a language can thus be associated with methods of **model checking**. In the same way, it is necessary **to validate a model** of the system compared to the expected properties of this system. Moreover, some models must allow the simulation of the dynamic behavior of the system: the language is then associated to a **model player**.

A model, whatever it is, cannot take into account all the aspects of a system itself. Thus, the design of an AMS requires the construction of several models. These models require various languages. To ensure the consistency and the complementarity between these various models, different languages must be integrated within integrated design methods. When they use one kind of model, these methods involve the concepts of construction, validation, checking and

player. When they use several kinds of model, these methods involve the concepts of **import and export between models**, and interpretation of a model to define properties.

2 State of the art

To structure the design of the AMS a lot of life-cycle oriented approaches were proposed by many authors. These approaches aim to specify the various activities required by the complete design of a AMS. For each activity, the expected results are specified, but without imposing any method of work. Then, these approaches are independent from the languages and methods used. But neither consistency between models, nor their internal consistency are approached.

With the need of models dealing with several requirements, CIM-OSA proposes a generic framework of modeling [1]. This framework of modeling proposes a classification along three axes of the models to build: the axis of generation, the axis of derivation and the axis of particularization. While referring to this cube, designers can check that the models built cover different and complementary aspects. CIM-OSA proposes models associated with each box within the cube, however users are free to use their own languages. But neither internal consistency with each model, nor the bonds between them are approached.

These two frameworks make it possible to structure the design of the AMS, but they do not directly contribute to modeling. In order to better define the languages used and the methods integrating these languages, a new approach appeared: the modeling of built models. These " models of models " are often called meta-models. In our opinion, only this new approach makes it possible to make these languages and methods more rigorous and more evolutionary. The construction of meta-models obliges us to structure our vision of the modelled language or method. In addition, the meta-models can be modified according to the practices or according to the new user's needs, while obliging the meta-modelisator to think about the consistency of the produced meta-model.

Most usually, the languages of meta-modeling used are the data models. The languages used are NIAM [2], or entity-relationship [3]. These data-oriented approaches allow a precise and clear expression of syntax and semantics of the models. However, they do not make it possible to take the dynamic behavior aspect of the models into account.

An algebraic modeling was used to improve the "internal" semantics (dynamic point of view) of the Sequential Function Chart (SFC) model [4]. The principle of this approach is to use algebraic equations to characterize the state of each step according to the conditions of activation and desactivation of this step. This modeling allows to formalize the dynamic behavior of the SFC. However, the specification of the dynamic behavior can be understood only if the specification of the static aspect is provided. The absence of description of the data, in this meta-model, makes us think that it can come only in complement to another meta-model. For the specification of the static and dynamic aspects, the interest of the use of two distinct meta-models is shown in [5]. However, we

think that this can lead to the construction of meta-models that do not take into account all the semantics of the modelled concepts. Therefore it can involve important losses or semantic errors.

The most advanced works on the construction of models are those by Soeki [6] and by Gee [7]. These works are complementary. Indeed, Soeki is interested in the specification of the steps of construction of a model, with an abstract aspect. On the other hand, Gee specifies the concrete aspect (visual) of language, without imposing precise steps in the method of construction.

Concerning multi-models methods, they have been studied at the LURPA Laboratory since 1990 [8]. In particular, Kiefer [9] made it possible to define the static point of view of the bonds existing between symbols of integrated languages within the same method.

3 The choice of the Z language

Unlike the approaches presented above, our objective was to be able to specify the syntax as well as the semantics of the languages of the AMS design within a meta-model, and also the methods associated with these languages. Moreover we aim to take into account all these aspects with only one meta-language, to avoid any problem of integration between several meta-languages. We also wanted to validate our meta-models. These constraints led us towards a formal meta-language.

The Z Language [10] was selected for these capacities to model data and modifications of the state of the modelled system. Moreover this language has had a significant development and is the subject of a large literature. A lot of softwares to write Z specifications exist: for example we used Z-EVES [11] to check and validate our meta-models.

The first benefits of the use of Z for meta-modeling were presented in [12]. To validate our approach in a more exhaustive way, we applied it on two other examples: an example of a single language for AMS design and an example of a multi-language method.

4 Meta-modeling of a single language

The selected language is a language which allows us to use many aspects of our approach. It requires the meta-modeling of its syntax, of its semantics and also of its associated player and method of construction of models: the Generalized Petri Net was chosen [13].

To build a meta-model of a language, there are three stages (figure 2). The first stage consists in building the meta-model of the language using its textual definition. Then, this meta-model must be checked, that is the second stage. Lastly, third stage, the meta-model must be validated compared to the need. These three stages are presented here.

4.1 Definition

A Generalized Petri Net (GPN) is a 5-tuple, PN = (P, T, F, W, M_o) where:
P = $\{p_1, p_2, \ldots, p_n\}$ is a finite set of places,
T = $\{t_1, t_2, \ldots, t_n\}$ is a finite set of transitions,
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,
$W : F \to \{1, 2, 3, \ldots\}$ is a weight function,
$M_0 : P \to \{1, 2, 3, \ldots\}$ is the initial marking,
$P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

4.2 A part of the meta-model

First of all, the meta-model describes all used types:

$[PLACE, TRANSITION]$

$USE ::= build \mid play$

The PN schema describes used the variables in a Petri net model:

PN

$P : \mathbb{F}\, PLACE$
$T : \mathbb{F}\, TRANSITION$
$arcTP : TRANSITION \leftrightarrow PLACE$
$arcPT : PLACE \leftrightarrow TRANSITION$
$WarcTP : TRANSITION \times PLACE \nrightarrow \mathbb{N}_1$
$WarcPT : PLACE \times TRANSITION \nrightarrow \mathbb{N}_1$
$M0 : PLACE \nrightarrow \mathbb{N}$
$M : PLACE \nrightarrow \mathbb{N}$
$enabled : \mathbb{F}\, TRANSITION$
$usePN : USE$

$\mathrm{dom}(arcTP) \subseteq T$
$\mathrm{ran}(arcTP) \subseteq P$
$\mathrm{dom}(arcPT) \subseteq P$
$\mathrm{ran}(arcPT) \subseteq T$
$\mathrm{dom}(WarcTP) = arcTP$
$\mathrm{dom}(WarcPT) = arcPT$
$\mathrm{dom}(M0) = P$
$\mathrm{dom}(M) = P$

The *Play* schema gives the evolution of state variable of the PN schema when the transition named *Fired*? is fired. This schema is an example of a specification of a player:

$$
\begin{array}{|l}
\hline
\textit{Play} \\
\Delta PN \\
Fired? : TRANSITION \\
\hline
usePN = play \\
Fired? \in enabled \\
P' = P \\
T' = T \\
arcTP' = arcTP \\
arcPT' = arcPT \\
WarcTP' = WarcTP \\
WarcPT' = WarcPT \\
M0' = M0 \\
\forall\, p : PLACE \mid p \in P \bullet \\
\quad (p \in arcPT^{\sim}(\!|\, \{Fired?\} \,|\!) \cap arcTP(\!|\, \{Fired?\} \,|\!) \\
\quad \land\ M'(p) = M(p) - WarcPT(p \mapsto Fired?) \\
\qquad + WarcTP(Fired? \mapsto p)) \\
\qquad\qquad \lor \\
\quad (p \in arcPT^{\sim}(\!|\, \{Fired?\} \,|\!) \setminus arcTP(\!|\, \{Fired?\} \,|\!) \\
\quad \land\ M'(p) = M(p) - WarcPT(p \mapsto Fired?)) \\
\qquad\qquad \lor \\
\quad (p \in arcTP(\!|\, \{Fired?\} \,|\!) \setminus arcPT^{\sim}(\!|\, \{Fired?\} \,|\!) \\
\quad \land\ M'(p) = M(p) + WarcTP(Fired? \mapsto p)) \\
\qquad\qquad \lor \\
\quad (p \notin arcPT^{\sim}(\!|\, \{Fired?\} \,|\!) \cup arcTP(\!|\, \{Fired?\} \,|\!) \\
\quad \land\ M'(p) = M(p)) \\
enabled' = \{t : TRANSITION \mid (\forall\, p : PLACE \bullet \\
\quad (p, t) \in arcPT' \land WarcPT'(p, t) \leq M'(p)) \\
\qquad \lor\ t \notin \mathrm{ran}(arcPT') \bullet t\} \\
usePN' = usePN \\
\hline
\end{array}
$$

4.3 The checking of the meta-model

To check a specification in Z language, two properties must be checked (in addition to the respect of types in the operation schemas):

- the specified state has no contradiction. This can be established by showing that the contraint part of the state schema is satisfiable. This is usually achieved by proving an initialization theorem: we show that an initial state, at least, exists:

  $$\exists\, PN' \bullet PNinitial$$

- The operations are total, i.e. that they are always defined.

For example, the pre-condition of the operation schema *Play* is obtained here:

$$\mathrm{pre}\ Play \mathrel{\widehat{=}} usePN = play \land Fired? \in enabled$$

Thus, this operation is not total. It is necessary to improve these schemas, and for this reason we will create new diagrams.

$$Report ::= okay \mid place\_in\_use \mid place\_not\_in\_use \mid transition\_in\_use \mid transition\_not\_in\_use \mid bad\_operation \mid transition\_not\_in\_enabled \mid P\_and\_T\_empties$$

Success

$$r! : Report$$

$$r! = okay$$

The new player operation is done by the *Simulation* schema:

ErrorFired

$$\Delta PN$$
$$Fired? : TRANSITION$$
$$r! : Report$$

$$usePN = play$$
$$Fired? \notin enabled$$
$$r! = transition\_not\_in\_enabled$$

PlayOrStopPlayInBuild

$$\Xi PN$$
$$r! : Report$$

$$usePN = build$$
$$r! = bad\_operation$$

$$Simulation \mathrel{\widehat=} (play \land Success) \lor ErrorFired \lor PlayOrStopPlayInBuild$$

The operation defined by *Simulation* is total :

$$\begin{aligned}
&\text{pre } Simulation \\
&\Leftrightarrow (\text{pre } play \land \text{pre } Success) \\
&\quad \lor \text{pre } ErrorFired \lor \text{pre } StartPlayOrPlayInPlay \\
&\Leftrightarrow (usePN = play \land Fired? \in enabled \text{pre } true) \lor \\
&\quad (usePN = play \land Fired? \notin enabled) \\
&\quad \lor usePN = build \\
&\Leftrightarrow usePN = play \lor usePN = build \\
&\Leftrightarrow true
\end{aligned}$$

4.4 Validation

To check the operation schemas, we used the Z-EVES toolbox [11]. The PN instance, described figure 1, was designed to highlight all the possible cases of marking which allow to fire a transition.

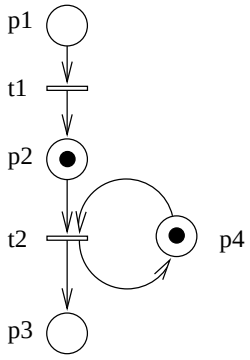

Fig. 1. An example of PN model

In order to simplify the validation, we modified the initial state to describe the PN model presented directly:

PNinitial

PN'
$t1, t2 : TRANSITION$
$p1, p2, p3, p4 : PLACE$

$P' = \{p1, p2, p3, p4\}$
$T' = \{t1, t2\}$
$arcTP' = \{(t1, p2), (t2, p3), (t2, p4)\}$
$arcPT' = \{(p1, t1), (p2, t2), (p4, t2)\}$
$WarcTP' = \{((t1, p2), 1), ((t2, p3), 1), ((t2, p4), 1)\}$
$WarcPT' = \{((p1, t1), 1), ((p2, t2), 1), ((p4, t2), 1)\}$
$M0' = \{(p1, 0), (p2, 1), (p3, 0), (p4, 1)\}$
$M' = \{(p1, 0), (p2, 1), (p3, 0), (p4, 1)\}$
$enabled' = \{\}$
$usePN' = build$

We checked the *Play* schema with the following new schema:

$Test2 \mathrel{\widehat{=}} PNinitial \fatsemi StartPlay \fatsemi Play$

We checked this schema with the following value:

$Fired? := t2$

The result obtained was the result expected:

- when *StartPlay* has been performed, we get $enabled' = \{t2\}$;
- when *Play* has been performed, we get :
  - $enabled' = \{\}$ and
  - $M' = \{(p1, 0), (p2, 0), (p3, 1), (p4, 1)\}$.

This small example, and many others, allow us to validate our meta-model compared to our need, and this by the simple instanciation of the definite sets.

5 Meta-modeling of a multi-langages method

The first stage of the meta-modeling of a multi-language method consists in meta-modeling each language which composes it. For each language, as for the example of the previous session, it is necessary to build the meta-model of the language and of the associated methods, then to check and validate these meta-models. The meta-models of each language are then integrated within the meta-model of the method. This also requires to add all the relations allowing this integration. That can also require some modification of certain elements of the original meta-models. The meta-model obtained must also be checked and validated.

In [14], the multi-language method tested is a method which integrates a particular class of Petri net and differential equations. Thus, this test allowed us to show that our approach works for methods of design of dynamic hybrid systems.

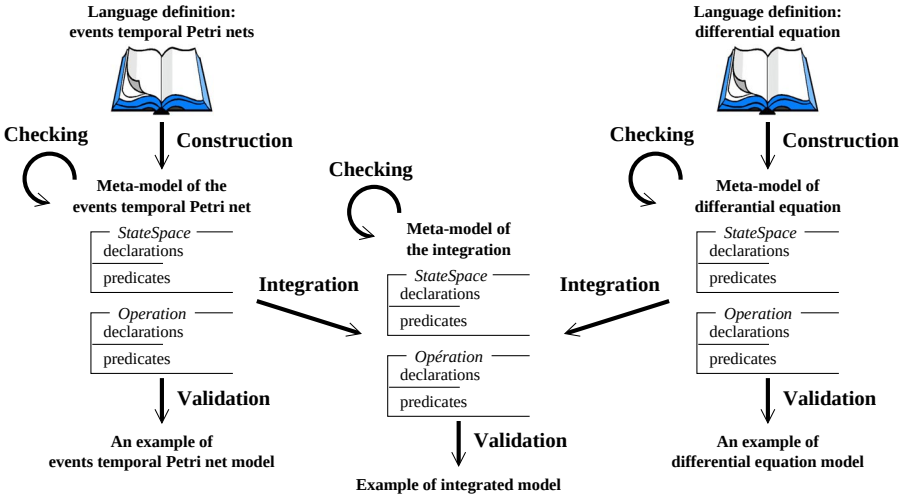

Fig. 2. Stages for meta-modeling a method

In this paper, it is not possible to develop the meta-language of such a method. The interested reader can refer to [14].

6 Conclusion

Our work has shown the feasibility and interest of the formal meta-modeling of the activity of modeling for automated manufacturing systems. the use of the Z language makes it possible to specify the syntax, the semantics, the dynamic behavior of the languages, as well as the methods associated with these languages, all in the same meta-model. This integration of all the aspects of the modeling activity guarantees the consistency of the meta-models. The use of a formal language as meta-language makes it possible to validate and check the meta-models, thus ensuring their rigor and their respect of the needs.

A meta-model of a language or a method can be used as a requirement for software editors. Then, while we specify a language of a method using meta-modeling, it could be interesting at the same time to "run" the current specification. That should help the designers to refine their specification. Our current works thus aim to study the interest of the formal B language to get a "playable" specification.

References

1. A. Consortium, ed., *Open system architecture for CIM, Research reports Esprit project 688.* AMICE Consortium - Springer verlag, 1989.
2. P. Lhoste, *Contribution au génie automatique : concepts, modèles, méthodes et outils.* Habilitation à diriger les recherches, Nancy I University, France, 1994.
3. F. Couffin, S. Lampérière, and J.-M. Faure, "Contribution to the grafcet formalisation. a static meta-model proposition," *European Journal of Automation*, vol. 31, no. 4, pp. 645–667, 1997.
4. E. Bon-Bierel, "Méta-modèles du grafcet," Master's thesis, Nancy I University/ENS de Cachan, France, 1994.
5. E. Bon-Bierel, *Contribution à l'intégration des modèles de systèmes de production manufacturière par méta-modélisation.* PhD thesis, Nancy I University, France, 1998.
6. M. Saeki, "A meta-model for method integration," *Information and software technology*, vol. 39, pp. 925–932, 1998.
7. D. M. Gee, "Formal specification of visual languages." http://computing.unn.ac.uk/ davidg/papers/fspec.ps.Z, 1995.
8. B. Denis, J.-J. Lesage, and G. Timon, "Towards a theory of integrated modelling," *Journal of Design Sciences and Technology*, vol. 2, pp. 87–96, Oct. 1993.
9. F. Kiefer, *Contribution à l'ingénierie intégrée des systèmes de production : formalisation des mécanismes d'intégration entre modèles et applications sur site industriel.* PhD thesis, ENS de Cachan, France, 1996.
10. J. M. Spivey, *The Z notation.* Prentice-Hall Europe, 2nd ed., 1992.
11. M. Saaltink, "The z/eves system," tech. rep., ORA Canada, Ottawa, Ontario, Canada, 1995.
12. L. Pietrac, B. Denis, and J.-J. Lesage, "Formalization of the design of control systems," in *Sixth International Symposium on Robotics and Manufacturing (ISRAM'96), Second World Automation Congress (WAC'96)*, (Montpellier, France), 27–30 may 1996.

13. T. Murata, "Petri nets : Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, 1989.
14. L. Piétrac, *Apport de la méta-modélisation formelle pour la conception des systèmes automatisés de production.* PhD thesis, ENS de Cachan, France, 1999.