

L'apport des logiciels libres à la modélisation et la simulation des systèmes dynamiques

Contribution of open sources software to the modeling and simulation of dynamic systems

Thierry BASTOGNE

Université Henri Poincaré, Nancy 1,
Centre de Recherche en Automatique de Nancy (CRAN),
CNRS-INPL-UHP UMR 7039,
BP 239, F-54506 Vandœuvre-lès-Nancy Cedex, France,
Tél.: (33) 3 83 68 44 73 - Fax: (33) 3 83 68 44 62
Mél.: thierry.bastogne@cran.uhp-nancy.fr

Mots-clés : logiciels libres, modélisation, simulation, systèmes dynamiques, UML[©], Modelica[©].

Keywords : open source software, modeling, simulation, dynamic systems, UML[©], Modelica[©].

L'accroche. Cet article promeut l'utilisation de langages objet couplés à des compilateurs libres pour la modélisation et la simulation des systèmes dynamiques.

L'essentiel. La rentabilité du processus de modélisation d'un système dépend de facteurs comme l'indépendance de la démarche vis à vis des outils logiciels propriétaires, optimiser la ré-utilisation des modèles existants ainsi que la flexibilité et la portabilité des modèles développés. Une solution préconisée dans cet article est d'associer des langages de modélisation standard et orientés-objet à des logiciels libres de compilation pour la simulation. Le cas de deux langages objet, UML et Modelica, et d'un logiciel libre de compilation, Open Modelica, illustre ce point de vue. Il souligne aussi les efforts à développer en termes d'intégration et d'interopérabilité des différents logiciels libres de calcul scientifique.

L’essentiel. The profitability of the system modeling process depends on factors like the independence of the approach with respect to the closed source software, optimizing the re-use of existing models as well as the flexibility and portability of the developed models. A solution, recommended in this article, is to associate standard and object-oriented modeling languages with free compilers for their simulation. A study case, based on two object languages, UML and Modelica, and one open source simulation software, Open Modelica, illustrates this point of view. Efforts should now be focused on the improvement of open source software, for scientific computing particularly, in terms of integration and interoperability across software.

Résumé. Cet article aborde l’apport des logiciels libres à la modélisation et la simulation des systèmes dynamiques. L’indépendance de la méthode de modélisation vis à vis des outils logiciels, la ré-utilisation maximale des modèles existants, la flexibilité et la portabilité des modèles développés sont des critères primordiaux en modélisation et simulation. Un modèle est souvent décrit par un langage informatique et sa simulation est alors résolue par un logiciel de compilation associé au langage. L’utilisation d’un langage standard de type orienté-objet est un premier moyen de répondre à ces spécifications. Mais une totale maîtrise du processus de modélisation/simulation nécessite aussi d’utiliser des logiciels libres de simulation. Bien qu’antinomique à première vue, l’utilisation de langages standards n’est pas incompatible avec celle de compilateurs libres. Deux exemples de langage, UML pour la modélisation à haut niveau d’abstraction et Modelica pour la modélisation de bas niveau, ainsi que Open Modelica, compilateur libre du langage Modelica, illustrent ce point de vue. Toutefois, ce cas d’étude montre aussi certaines limites comme (1) une interface homme-machine généralement rudimentaire, (2) une absence d’outil de débogage efficace et (3) une absence quasi totale d’interopérabilité entre logiciels libres. Cela est particulièrement valable pour les logiciels de calcul scientifique comme R, MAXIMA, AXIOM, SCILAB, OCTAVE et OPEN MODELICA dans le but de disposer d’une plate-forme complète couvrant des disciplines variées comme le calcul statistique, le calcul formel, le calcul algébrique ou la modélisation sous forme de diagrammes objet.

Abstract. This article deals with the contribution of open source software to the modeling and simulation of dynamic systems. The independence of the modeling approach with respect to the simulation software, the maximum re-use of the existing models, the flexibility and portability of the developed models are essential specifications in modeling and simulation. A model is generally described by a programming language and its simulation is solved by a compilation software. The use of standard and object-oriented languages is a first recommendation. A second requirement is to use open source simulation software. Although paradoxical at first sight, the use of standard languages is not incompatible with that of open source compilers. Two languages, UML for high-level modeling and Modelica for low-level modeling, and one open source simulation

software, Open Modelica, illustrate this point of view. However, this study case also shows some limits like (1) a generally rudimentary man-machine interface, (2) the absence of effective tool of debugging and (3) the quasi complete absence of interworking between open source software. That is particularly important in scientific computing with open source software like R, MAXIMUM, AXIOM, SCILAB, OCTAVE and OPEN MODELICA in order to get a complete scientific platform covering various disciplines, *e.g.* statistical computing, symbolic computation, matrix algebra or object modeling diagrams.

1 Introduction

En ingénierie des systèmes, la maîtrise des coûts, des délais, des risques et de la complexité, passe de plus en plus souvent par une étape de modélisation et de simulation des systèmes assistée par ordinateur [3, 4, 7]. Un modèle est généralement une description abstraite du système par une théorie ou par un langage symbolique donné. Il peut être mathématique, graphique ou algorithmique. Nous évoquons ici le cas des modèles à base de composants, modèles décrivant les composants internes du système et leurs modes d'interaction. La figure 1 présente un exemple de système hydraulique constitué d'une cuve de mélange binaire et de son instrumentation. Le modèle de composant de ce système est décrit à la figure 2, chacun des objets du modèle est associé à un des composants du système, les liens en trait plein représentent les échanges d'énergie alors que ceux en trait pointillé correspondent aux transmissions d'information [2]. La phase de développement de ces modèles est représentée par un cycle en V à la figure 3. On y retrouve une première phase d'analyse dans laquelle le système est décomposé en classes d'objets, puis une seconde phase de synthèse dans laquelle les objets modélisés sont recomposés pour constituer le modèle. La phase

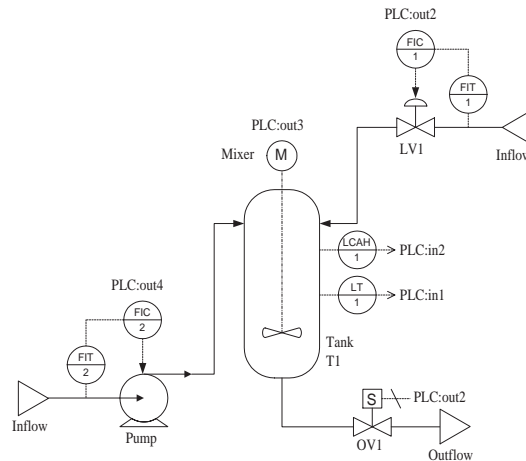


FIG. 1 – Système à modéliser

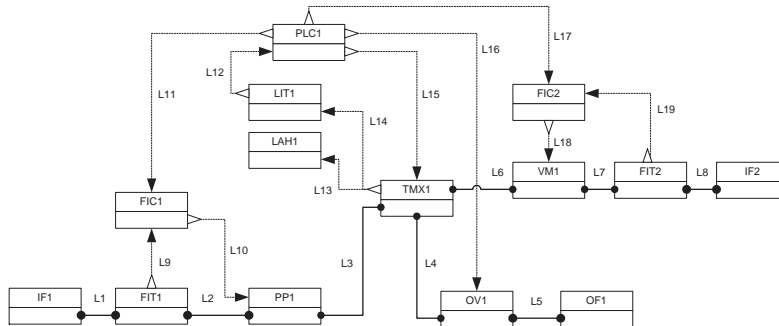


FIG. 2 – Modèle de composant du système

de modélisation est généralement une tâche longue, coûteuse et incertaine. Le rendement de cette étape et la robustesse du modèle vis à vis des erreurs de modélisation et des diverses incertitudes dépendent essentiellement des trois points suivants :

- la liberté de la démarche vis à vis des outils logiciels propriétaires, l’approche ne doit pas être remise en cause par la moindre modification du logiciel ;
- choisir les outils logiciels en fonction de la méthode et non l’inverse afin de pérenniser l’approche ;
- et capitaliser au mieux les résultats de la modélisation.

Les outils logiciels libres offrent une réponse aux deux premiers points de cette liste. Il existe aujourd’hui plusieurs logiciels libres de calcul scientifique comme R^1 pour le calcul statistique, MAXIMA² et AXIOM³ pour le calcul symbolique, SCILAB⁴ et OCTAVE⁵ pour le calcul algébrique, SCICOS⁶ pour l’édition de schémas-blocs ainsi que OPEN MODELICA⁷ pour la modélisation objet des systèmes physiques. En plus d’être des logiciels libres, tous sont également des gratuits. Pour ce qui est du dernier point, la capitalisation efficace de la connaissance implique plusieurs conditions :

- optimiser la ré-utilisation des modèles existants ;
- la modularité ou flexibilité des modèles, c’est-à-dire pouvoir modifier ou mettre à jour une partie du modèle sans systématiquement avoir à remettre en cause le reste ;
- et vérifier la portabilité des modèles, c’est à dire pouvoir exécuter le modèle sur d’autres plates-formes de simulation.

¹<http://www.r-project.org/>

²<http://maxima.sourceforge.net/>

³<http://savannah.nongnu.org/projects/axiom>

⁴<http://www.scilab.org>

⁵<http://www.octave.org/>

⁶<http://www.scicos.org/>

⁷<http://www.ida.liu.se/~pelab/modelica/OpenModelica.html>

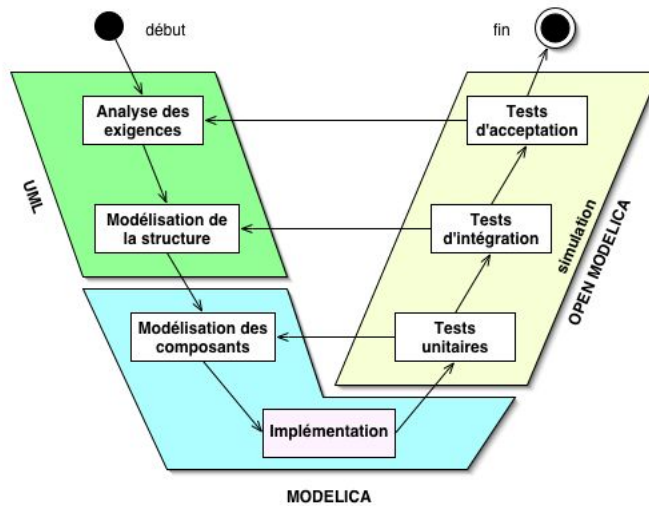


FIG. 3 – Cycle de modélisation/simulation en V

Capitaliser les modèles dans des bibliothèques est un moyen couramment utilisé par les logiciels de modélisation pour permettre à l'utilisateur de les réutiliser ultérieurement. Cependant, un objet doit aussi posséder des interfaces de communication standards pour être re-connecté facilement dans un autre modèle. Concernant la seconde spécification, les notions de classes et d'héritage liées au paradigme objet offrent un début de solution. A cela il faut ajouter la possibilité d'une modélisation non causale ou implicite des systèmes, c'est-à-dire la possibilité de déclarer des équations de comportement sans avoir à définir au préalable si une variable est une variable d'entrée ou de sortie. En effet ce caractère entrée-sortie dépend pleinement de chaque application. Enfin, pour ce qui est du dernier point, une solution consiste à utiliser des langages standards ou quasi-standards. En effet, il est tout à fait envisageable de développer des modèles sur la base de langages standards à l'aide de logiciels libres de compilation. Un exemple bien connu est GCC⁸, logiciel libre GNU, pour la compilation d'algorithmes écrits en langage C, C++ ou Objective-C. Les langages UML et Modelica évoqués dans les paragraphes suivants sont deux illustrations de solution pour la modélisation de systèmes dynamiques.

2 Modélisation structurelle avec UML

Une première étape en modélisation à base de composants consiste à déterminer la structure statique du système, c'est-à-dire identifier les classes d'objets et leurs modes d'interaction. La manière d'aborder cette décomposition est multiple.

⁸<http://www.gnu.org/software/gcc/gcc.html>

Une première question concerne la nature de la décomposition. Elle peut être matérielle, temporelle ou fonctionnelle. Une seconde question traite de la direction de la décomposition. Elle peut être descendante, montante ou mixte. Une analyse descendante démarre de la définition générale du système et consiste à le décomposer en éléments de plus en plus petits et précis. Cette procédure s'arrête lorsque tous les éléments sont supposés être connus. Cette approche générale, souvent considérée comme étant la plus systématique et la plus logique, permet à l'ingénieur de préserver les aspects génériques du système. Néanmoins, les éléments sont rarement tous connus et la réalité n'est pas toujours aussi cohérente que souhaitée par ce mode de décomposition. Cette solution est plutôt utilisée dans les études de conception. En rétro-ingénierie, il est souvent plus économique de commencer par considérer les éléments terminaux de l'installation, c'est à dire d'effectuer une analyse montante. Cette approche consiste à agréger les éléments en sous-systèmes de plus en plus importants jusqu'à obtenir le système complet. Mais, il est encore souvent plus pratique de combiner les deux approches via une décomposition mixte [1]. Enfin, une dernière question consiste à déterminer le niveau d'abstraction du modèle : modélisation microscopique ou macroscopique ? Ce choix dépend essentiellement de l'application finale du modèle : prédiction, fiabilité, sûreté de fonctionnement, simulation, etc.

Pour traiter ces différentes questions, le langage UML (Unified Modeling Language) offre, dans son panel de diagrammes, quelques représentations utiles à cette décomposition [5]. UML est né de la fusion des méthodes objet (OMT, Booch et OOSE) et a été normalisé par l'OMG en 1997. UML n'est pas à proprement dit une méthode de modélisation mais plutôt une bibliothèque de diagrammes objets. Dans cette collection, les diagrammes de classe et les diagrammes d'activité sont deux outils permettant de traiter la décomposition du système sous des aspects matériel, temporel ou fonctionnel. Beaucoup de logiciels libres pour l'édition de graphiques ou de schémas, comme DIA⁹ logiciel libre sous licence GPL, possèdent désormais des bibliothèques d'édition de diagrammes en langage UML.

Les diagrammes de classes expriment la structure statique d'un système, en termes de classes et de relations entre ces classes. La figure 4 présente un exemple de diagramme de classe d'une unité de stockage composée de n cuves de classe **C**, version spéciale d'une classe de cuve générique : **CG**. Les classes sont ici représentées par des rectangles compartimentés contenant leur identité, leurs attributs (paramètres et variables) et leur comportement exprimé sous la forme d'équations décrites en langage Modelica. La composition est une relation non symétrique entre deux classes dans laquelle la classe dominante est appelée classe composite ou classe conteneur et l'autre classe est la classe composante. La composition se représente dans les diagrammes par un losange de couleur noire placé du côté de la classe composite. La généralisation est une relation entre une classe plus générale et une classe plus spécifique. La relation de généralisation se représente au moyen d'une flèche orientée de la classe plus spécialisée vers la

⁹<http://www.gnome.org/projects/dia/>

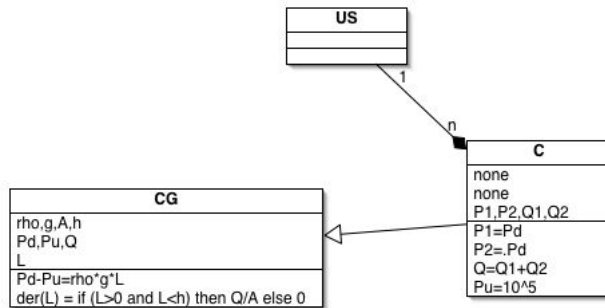


FIG. 4 – Diagramme de classe d’une unité de stockage : classe **US**

classe plus générale.

Un diagramme d’activité est un graphe dirigé représentant le déroulement d’un processus sous la forme d’étapes ordonnées séquentiellement. Le début et la fin du processus sont représentés respectivement par un point noir et un point noir encerclé, les activités par des rectangles arrondis et les transitions par des connexions unidirectionnelles. La figure 5 présente un exemple de diagramme d’activité d’un cycle de production d’une unité de mélange. Ce cycle est décomposé en quatre phases : repos, remplissage, mélange et vidange.

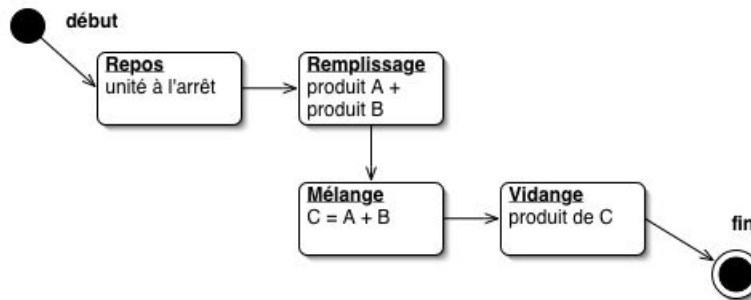


FIG. 5 – Diagramme d’activité d’une unité de mélange

3 Modélisation à base de composant avec le langage Modelica

Une fois la structure statique du modèle de composant déterminée, il reste à définir le comportement de chaque classe d’objet. Le langage Modelica, né en 1998 par un effort d’unification de langages objets existants : Alan, Dymola, NMF, ObjectMath, Omola, SIDOPS+, Smile [6], peut être utilisé dans ce but. La publication et la spécification de ce langage sont réalisées par l’association

Modelica¹⁰. Modelica n'est pas un langage standard au sens ISO/ANSI/IEEE mais un langage non propriétaire à l'image de C++ mais dédié à la modélisation des systèmes physiques. Il possède tous les avantages des langages objets. En outre, il permet la modélisation non causale ou implicite des systèmes. De plus, il possède des instructions lui permettant de traiter simplement la modélisation d'un grand nombre de systèmes à dynamique hybride. Enfin, autre avantage, il autorise l'inclusion de routines écrites en C et Fortran77. L'écriture d'un modèle de composant à l'aide du langage Modelica peut être réalisée à partir de n'importe quel éditeur de texte. La figure 6 montre l'exemple d'un gratuitiel, FME¹¹, dédié à l'édition des modèles développés en Modelica. L'algorithme édité à la figure 6 est un exemple basique de modèle d'une classe d'objet associée à une cuve de vidange. On y retrouve des instructions de base comme (**der**) pour la dérivée temporelle d'une variable et les instructions conditionnelles (**if** . . . **then** . . . **else**) pour le traitement des discontinuités dans les équations différentielles.

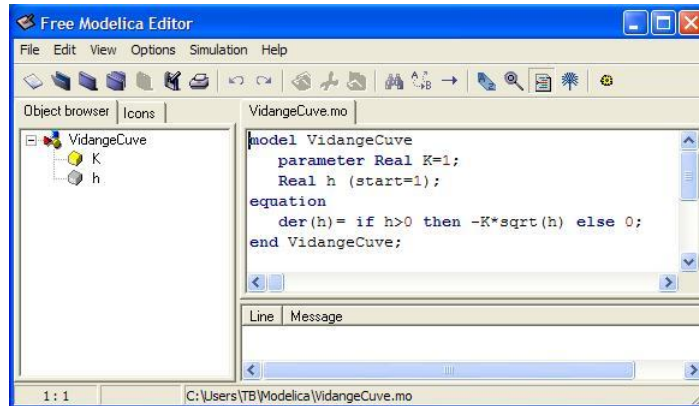


FIG. 6 – FME : Free Modelica Editor

Mais un tel modèle est inutile si on ne dispose pas des outils logiciels pour le simuler. Jusqu'en 2004 les seuls compilateurs du langage Modelica étaient commerciaux. Depuis 2004, il existe un début d'alternative fondée sur un compilateur libre, Open Modelica[©], développé à l'Université de Linköping en Suède. La figure 7 montre les fenêtres de commande et de tracé des courbes de cet outil. La première fenêtre, figure 7(a) permet de lancer les instructions de chargement du modèle, de simulation et de tracé des résultats. La figure 7(b) illustre un exemple de courbe de simulation dans le cas de la cuve de vidange.

L'interface graphique de ce compilateur est certes sommaire et ne vaut pas encore celle des produits commerciaux dans ce domaine mais la jeunesse de ce compilateur explique cette différence. L'évolution rapide, si l'on en juge sur le nombre de versions développées en un an, permet d'espérer une amélioration rapide et significative de son ergonomie et de ses capacités. On

¹⁰<http://www.modelica.org>

¹¹<http://www.et.web.mek.dtu.dk/FME/download.html>

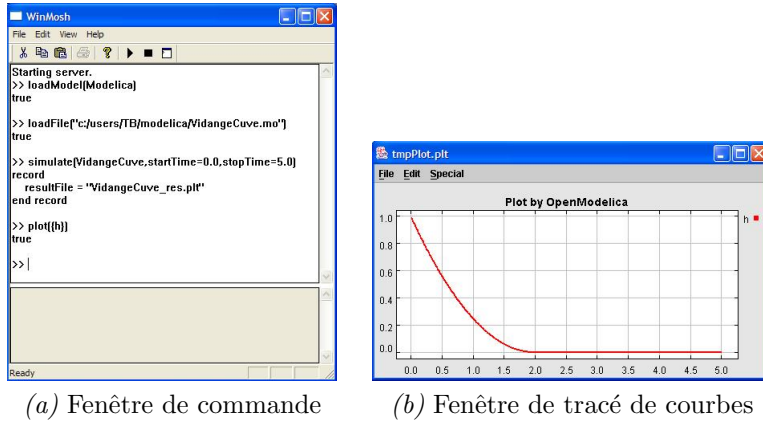


FIG. 7 – Interfaces graphiques de Open Modelica v. 1.2

peut espérer également l'apparition d'autres compilateurs libres comme celui en développement à l'INRIA au sein des projets successifs SYMPA¹² et METALAU¹³ pour son intégration dans SCICOS.

4 Conclusion

La modélisation et la simulation des systèmes sont deux disciplines de plus en plus utilisées en ingénierie des systèmes pour la maîtrise des coûts, des délais, des risques et de la complexité. Très liées, ces deux disciplines sont, à tort, souvent confondues. En mathématique, la modélisation peut être vue comme la mise en équation du problème et la simulation comme la résolution de ces équations; en informatique, la modélisation correspond plutôt à la programmation et la simulation à la compilation. Dans le cas des systèmes dynamiques complexes, pour des critères d'indépendance, de ré-utilisation, de flexibilité et de portabilité, il est intéressant d'utiliser des langages standards ou quasi-standards pour la partie modélisation et des logiciels libres de compilation pour la simulation. Bien qu'antinomique à première vue, l'utilisation de langages standards n'est pas incompatible avec celle de compilateurs libres mais plutôt complémentaire. En effet, le paradigme objet contribue aux aspects de ré-utilisation et de flexibilité des modèles, l'utilisation de langages standards accroît leur portabilité et l'utilisation de logiciels libres de simulation participe à l'indépendance de l'approche vis à vis des outils logiciels propriétaires. Deux exemples de langage, UML pour la modélisation à haut niveau d'abstraction et Modelica pour la modélisation de bas niveau, ainsi que Open Modelica, compilateur libre du langage Modelica, illustrent ce point de vue. Toutefois, l'utilisation de ce compilateur libre montre des limites qui sont aussi celles de la plupart des logiciels libres, à savoir : (1)

¹²<http://www.telecom.gouv.fr/rntl/FichesA/Simpa.htm>

¹³<http://www.inria.fr/rapportsactivite/RA2002/metalau/module10.html>

une interface homme-machine généralement rudimentaire, (2) une absence d'outil de débogage efficace et (3) une absence quasi totale d'interopérabilité entre logiciels libres. Ce dernier point est particulièrement valable pour les logiciels de calcul scientifique comme R, MAXIMA, AXIOM, SCILAB, OCTAVE et OPEN MODELICA dans le but de disposer d'une plate-forme complète couvrant des disciplines variées comme le calcul statistique, le calcul formel, le calcul matriciel ou la modélisation sous forme de diagrammes objet.

5 Les auteurs

Thierry Bastogne est maître de conférences en Automatique et Traitement du Signal à l'Université Henri Poincaré, Nancy 1, depuis 1998. Il effectue ses recherches au CRAN (Centre de Recherche en Automatique de Nancy) sur la modélisation expérimentale des systèmes dynamiques.

Références

- [1] K. J. Aström and B. Wittenmark. *Computer-Controlled Systems - Theory and Design*, chapter Design : An Overview, pages 224–241. Prentice Hall, 1997. Third Edition.
- [2] T. Bastogne. A multiport object-oriented diagram for batch system modeling. Methodology and implementation. *Simulation Practice and Theory*, 12(6) :425–449, October 2004.
- [3] F. E. Cellier. *Continuous System Modeling*. Springer-Verlag, 1991.
- [4] L. Ljung and S.T. Glad. On global identifiability for arbitrary model parameterizations. *Automatica*, 30 :265–276, 1994.
- [5] P.-A. Muller and N. Gaertner. *Modélisation objet avec UML*. Eyrolles, 2002.
- [6] M. Tiller. *Introduction to Physical Modeling With Modelica*. Kluwer International Series in Engineering and Computer Science, 2001.
- [7] B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of Modeling and Simulation - Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, 1999. Second Edition.