Preprints of the 19th World Congress
The International Federation of Automatic Control
Cape Town, South Africa. August 24-29, 2014

Component-Based Design of Simulation Models Utilizing Bond-Graph Theory

Petr Novák [*,**] Radek Šindelář [*]

[*] *Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems, Vienna University of Technology, 1040 Vienna, Austria (e-mail: {novak,sindelar}@ifs.tuwien.ac.at)*
[**] *Czech Technical University in Prague, 16636 Prague, Czech Republic*

Abstract: Simulation models are becoming efficient tools for real plant testing and control system fine-tuning. They are necessary for advanced process control. Since a design phase of simulation models is time-consuming and error-prone, this paper proposes a method for the systematic and semi-automated design of simulation models. It is intended mainly for signal-oriented simulators such as MATLAB-Simulink. The method presumes that large-scale industrial systems consist of subsystems, whose behavior is known. The task for the proposed approach is to create a simulation model from atomic simulation blocks approximating system components. The proposed solution utilizes the bond graph theory in a non-traditional way. Compared to the classical use of bond graphs, the proposed method does not create simulations from scratch, but it re-uses existing simulation components. It extends bond graphs with a concept of component implementations and annotations of their interfaces. Thus, models are assembled from atomic component implementations, whose selection is done by the proposed method for each node of the system topology.

1. INTRODUCTION

Current industrial systems are becoming complex and sophisticated. Designing and fine-tuning of control algorithms for such systems are typically based on simulation models. Compared to the use of a real system, they can perform experiments more fast, safely, and with lower costs. However, the design of simulation models is a time-consuming and error-prone task, as it is based on repeating manual work. In addition, maintenance of models requires complex work of experts that should be semi-automated.

This paper proposes a method for a semi-automated design of dynamic simulation models for industrial systems. The proposed method assumes that simulation models consist of simulation components representing real devices. The method assembles simulations from components, which are typically comprised in simulation libraries. When creating a model, the components from the library are instantiated and interconnected in the simulation model according to the structure of the real system. This approach is typically called "object-oriented modeling", more details can be found e.g. in Sinha et al. [2001]. The proposed method supports a problem of multiple implementations of each component, which differ in input and output interfaces. As the method is designed mainly for signal-oriented simulators, the problem of interconnecting simulation components to create a dynamic model is a nontrivial issue.

The recognition of the appropriate signal-wiring between components is supported by the bond-graph theory in the proposed method. In Layman's terms, bond graphs are engineering tools that are focused on describing physical systems mathematically. They involve devices, their connections, directed power transfers, and causality strokes. The causality assignment is an important feature of bond graphs, extending preceding power graphs to a much more powerful tool. In the presented method, bond graphs play an important role for selecting the right implementations of simulation components when more than one are available for each device, and also for determining how to interconnect these components. The presented method can be considered as an extension of the bond-graph theory for legacy simulation components. The components are considered as gray-boxes with known interfaces, but a possibly unknown implementation of the functionality itself.

The proposed method has been investigated in the frame of the development of the Simulation Integration Framework Šindelář and Novák [2011, 2012]. It is the framework supporting the integration of simulation models within industrial SCADA systems as well as other tools used in the industrial automation practice. Furthermore, it supports the design of simulation models, as integration and design are coupled issues and it is not efficient to solve them separately. This article extends the method presented in Novák and Šindelář [2011b], which was focused on the simulation models creation based on single-input and single-output components, and which did not handle the problem of junctions (i.e., component interconnections) satisfactorily.

The remainder of this paper is structured as follows. Sec. 2 summarizes related work. Sec. 3 introduces the bond-graph theory in a traditional way. Sec. 4 proposes a component-based approach for simulation model design that utilizes bond-graphs in a non-traditional way. Sec. 5 illustrates the proposed approach on a simple use-case. Finally, Sec. 6 concludes and proposes further work.

2. RELATED WORK

A large variety of publications has been published in the area of bond graphs. A good introduction and motivation into bond graphs can be found in Blundell [1982]. The whole description of this method is summarized in Gawthrop and Bevan [2007], including various examples, system analogies, and practical issues. The authors of this paper learned about bond graphs from a text book Horáček [1999].

An approach combining bond graphs and object-oriented modeling to build simulation models is discussed in Borutzky [1999]. The paper summarizes three model description languages: MAST, VHDL-AMS, and SIDOPS. An application of VHDL-AMS is discussed in more details for example in Pêcheux et al. [2005], where the chemical domain is considered as a use-case.

The main contribution of Borutzky [1999] is focused on the application of bond graphs for Modelica-based models. Bond graph ports are realized by the "connector" class in Modelica language to specify the interface, which can be consequently connected via the operator "connect". Compared to Borutzky [1999], this paper is focused on signal-oriented simulators, whereas Modelica is equation-oriented. Furthermore, Borutzky [1999] does not discuss the problem of multiple implementations of specific components. The same author presented in Borutzky [2009] the whole bond graph method including specific steps to generate bond graphs for mechatronic systems, the sequential causality procedure introduced by Karnopp and Rosenberg, and finally some use-cases.

A modeling simulation package 20-SIM is presented in Broenink [1999]. It is an interactive tool for modeling and simulation of dynamic behavior of engineering systems. It has a modeling and a simulation part, it supports data sharing with other simulation packages, and it covers a wide range of techniques.

The generation of simulation models based on bond graphs is discussed in Beez et al. [2008]. Models are generated in the Modelica language and the real plant description utilizes AutomationML as a vendor-independent format for process and instrumentation description. Compared to this paper, we focus on signal-oriented simulators and we explicitly support various component implementations.

3. BOND GRAPHS

Bond-graphs are focused on describing power flows through systems, as "power is the universal currency of physical systems" Gawthrop and Bevan [2007]. Power is the rate of energy flow and mathematically, energy is the time-integral of power. Energy flows in from sources, it can be temporarily stored in specific components (such as capacitors or inductors in an electrical circuit), it is dissipated in some components (such as in resistors), and it produces desired effects in other components, see Blundell [1982].

Bond-graph theory is aimed at the unified and systematic way for mathematical description and modeling of physical systems. The bond-graph theory is based on the following three types of analogies, which are subsequently described in more details:

(1) Signal analogies;
(2) Component analogies;
(3) Connection analogies.

3.1 Signal analogies

Since physical systems are balancing energy by transferring power and tend to reach the highest entropy, the main impact on the system behavior have variables affecting the energy distribution within the system. As the rate of energy transfer is power, it is the power that has the fundamental role in bond graphs as well as in the method proposed in this paper.

In order to introduce a unified approach to diverse types of systems, the bond-graph theory defines two generic variables:

- Effort $e(t)$
- Flow $f(t)$

These variables are called "power variables" as their product is power:

$$p(t) = e(t)f(t)$$

Furthermore, the bond-graph theory utilizes two integrated variables, which are useful for component description:

- Integrated effort: $p(t) = p_0 + \int_{t_0}^{t} e(\tau)d\tau$
- Integrated flow: $q(t) = q_0 + \int_{t_0}^{t} f(\tau)d\tau$

Although mathematically it would be feasible to derivate these equations and to express flow (respectively effort) as a derivative of integrated flow (respectively integrated effort), these equations would not be causal in dynamic systems. The derivative operator needs to know future behavior, which is not numerically possible. These issues are reflected in the concept "causality", which is an important feature of bond graphs described later in Sec. 3.4.

3.2 Component Analogies

The above stated definition of generic signals is useful not only for the expression of signal relationships among systems of various physical nature, but also in order to define component interfaces and basic generic components. Component analogies are the second cornerstone, which is provided by the bond-graph theory.

The theory defines the following one-port components:

(1) *Source of effort* (SE) is an ideal source of effort. In electrical systems, it is an ideal voltage source.
(2) *Source of flow* (SF) is an ideal source of flow. In electrical systems, it is an ideal current source.
(3) *Resistor* (R) is a component, which relates effort and flow by a static function, which can be non-linear.
(4) *Capacitor* (C) is a component accumulating energy and having a static function between effort and integrated flow. This function can be non-linear.
(5) *Inductor* (I) is a component accumulating energy and having a static function between flow and integrated effort.

All these components are called one-port components. It means that each component couples one pair of effort and flow. One of these variables is input and the second one is output. Only in the case of sources, it is done by definition, which one is output; in the other cases, it is determined by the bond graph. The bond-graph theory also supports components having more than one ports. Examples of 2-port components are a transformer (TF) and a gyrator (GY), for simplicity reasons we restrict on 1-port components.

3.3 Connection Analogies

Having the generic components, the simulation model scheme should be created by interconnecting these components. The connections are called bonds in the language of bond graphs and in case of power transfers, it is a pair of power variables. The third analogy addresses the problem of connecting devices in series or in parallel.

The typical approach used for electrical circuit analysis is based on the Kirchhoff's first and second laws. In complex systems, it is hard to recognize and to automate which of them should be used. The bond-graph theory introduces an abstraction of connection types:

(1) *0-junction* is a junction having the equal value of effort on all bonds and the sum of the (directed) flows is zero:
$$e_1 = e_2 = ... = e_n$$
$$f_1 + f_2 + ... + f_n = 0$$
(2) *1-junction* is a junction having the equal flow for all bonds and the sum of effort equal to zero:
$$e_1 + e_2 + ... + e_n = 0$$
$$f_1 = f_2 = ... = f_n$$

The type of a junction to use depends on the type of the physical system as follows.

- *Non-mechanical systems*: In most of the systems, a 1-junction is a serial connection of components, whereas a 0-junction represents a parallel connection.
- *Mechanical systems*: In case of mechanical systems, the assignment is vice-versa. 1-junctions are parallel connections, whereas 0-junctions are serial ones.

3.4 Creating Bond Graphs

A bond graph is a graph containing components, junctions, connections, directions of power flows, and causality strokes. We have already discussed the fundamental issues related to components, junctions and connections. In the further text, we will focus on power direction, causality and the whole method of creating bond graphs.

The power direction defines the positive direction of power through each bond. This direction is not crucial in terms of the mathematical description, but it is an important issue for understanding the sign convention, i.e., what a positive or a negative value means for each bond. The theory recommends specific rules for assigning the direction. For example, the power should be directed out of sources, it should be directed into 1-port components C, I, or R, and last but not least, in 0 and 1-junctions the power should go out via at least one arc. These rule examples show that the theory fits well for the machine-based processing.

An important aspect of bond graphs is the causality, declaring which of the variable pair flow and effort is the given variable and which one is calculated in each connected component or junction. A source of effort has effort as a given output, whereas the flow depends on the remainder of the system. The second kind of a source, the source of flow, has a causality vice-versa. As it has been already mentioned, one of the requirements on simulation schemes is to calculate integrated variables by discrete summing and not calculating the flow or effort as a derivative of the integrated flow, respectively the integrated effort. Another requirement on the causality arises in the area of junctions. In case of 0-junctions, the common effort is given by exactly one component, whereas in case of 1-junctions, the common flow is given by exactly one component. The causality is denoted by adding a short bar to the end of a bond, Blundell [1982], which is called a causality stroke. For determining the causality, the bond-graph theory proposes a recipe defining the order: (i) Causality of bonds connected to sources, (ii) integral causality of component types I and C (if it is possible), (iii) causality of other remaining nodes in a way that requirements on node causality are satisfied. In case of resistors, the causality is arbitrary. If there is a causality collision, either a differential causality of I and C should be used, or it is solved by modifying the level of abstraction for system description.

Last but not least, the bond-graph theory defines several types of bond graph reductions. Due to the limited space, this problem is not addressed in this paper.

Summarizing the whole method, the theory proposes to perform the following steps to create the bond-graph:

(1) Generation of nodes representing components and junctions;
(2) Generation of arcs representing bonds;
(3) Assignment of the power direction;
(4) Exclusion of a reference junction;
(5) Reduction of the graph;
(6) Assignment of the causality strokes;
(7) Writing down mathematical equations manually.

4. COMPONENT-BASED METHOD FOR SIMULATION MODEL DESIGN

Bond graphs are suitable for machine-based processing as they offer simple lists of recommended steps and rules, which are easy to implement. Nevertheless, their typical use suffers from (i) manual use, and (ii) the method is by its nature analytic and works with an internal description of the behavior of the components. It does not face complex units such as pump stations seamlessly. This section motivates and explains the proposed method addressing the above mentioned shortcomings, which is the main contribution of this paper.

4.1 Motivation for the New Method

The basic motivation for the research can be summarized in the following requirements that should be fulfilled:

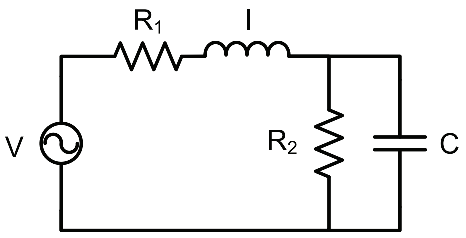

Fig. 1. Exemplary electrical circuit including a voltage source, resistors R_1 and R_2, and both accumulators of energy – a capacitor C and an inductor I.

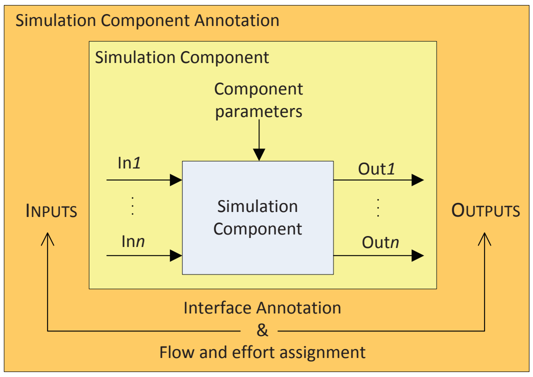

Fig. 2. A simulation component and its interfaces. The interface annotation includes inputs and outputs; and it maps inputs and outputs to generic signals flow and effort.

(1) *Support for a component-based approach*
Simulation models of large-scale industrial systems typically consist of simulation components representing sub-parts of the plant. The goal is to handle each component as a whole and to work just with its interface, no matter how the block is internally implemented.

(2) *Automate the bond-graph method*
The bond-graph method has been already implemented many times in diverse tools, nevertheless, these implementations work with a typical approach based on the analytic description of devices (such as capacitor, inductor, etc.). The proposed method is intended to support a component-based approach and to enable re-use of engineering artifacts. The idea of the presented generation of simulations is to non-experts be able to create simulations as well.

In order to illustrate the methods on a practical example, we will use the electrical circuit depicted in Fig. 1, which is utilized to demonstrate the proposed method.

4.2 The Proposed Method – Design of Extended Bond Graphs with Component Implementations

The proposed method adopts the assumption of standard bond graphs that systems consist of subsystems. We call models of the sub-systems "simulation components" and simulation models of industrial plants are created by combining and interconnecting these components. Fig. 2 depicts the view on a simulation component adopted in this paper. Each simulation component has input and output variables. For example, a resistor is a one-port component having one pair of flow and pressure, representing electrical current, respectively voltage in the real system.

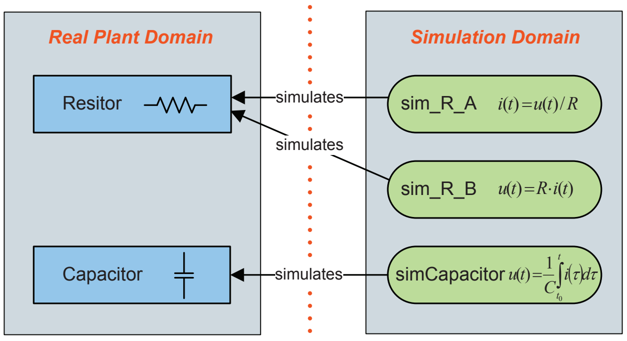

Fig. 3. Example of mappings between real devices and simulation components that approximate their behavior. In the case of the resistor, two component implementations are causal.

Considering the resistor as an example of a simulation component, we can model this elementary electrical engineering device by implementing one or both of the equations:

$$u(t) = Ri(t)$$
$$i(t) = u(t)/R$$

In the case of a resistor, such an example can seem to be a simple mathematical anagram at first. However in the case of complex and non-linear systems, the situation is much more difficult. In many cases, such dualities do not exist (e.g., due to non-linearities such as dead-zones), or the mathematical description is so complex that it is not efficient to work with it directly. Simulation experts are expected to use the whole simulation components without knowing their specific implementation details in the proposed approach (i.e., simulation models are considered as gray-boxes with known meaning of input and output variables).

The proposed method assumes that each simulation component can be modeled by one or more "simulation component implementations". Fig. 3 depicts the use-case, where each resistor can be simulated by either a resistor component implementation having effort as input and flow as output, or vice-versa. The goal of the proposed method is thus not only to instantiate components, but also to select the right implementation for each system topology node. Using the terminology of bond graphs, the proposed approach extends the concept "simulation component" with a set of one or more "simulation component implementations", which differ from each other in their interfaces.

The proposed method extends the original bond-graph method with the two following issues:

(1) Support for various component implementations;
(2) Improved causality assignment.

In more details, support for component implementations means that for each component (such as R or C), a set of available implementations is assigned. According to interfaces of component implementations, requirements on causality stroke positions from the component implementation point of view are extracted automatically. If

Algorithm 1 Generation of the Extended Bond Graph with Component Implementations

```
BondGraph $bondGraph$ = new BondGraph();
$bondGraph$.generateJunctions();
$bondGraph$.generateComponents();
$bondGraph$.addComponentImplementations();
$bondGraph$.generateBonds();
$bondGraph$.excludeReferenceNode($referenceNode$);
Bond $bond$ = $bondGraph$.getStartingBond();
$bond$.setExplored(true);
BGStack $bondsToAssign$ = new BGStack();
$bondsToAssign$.push($bond$);
while $bond$ != null do
    enum $result$ ={assigned, notAssignable};
    result $retVal$ = $bond$.assignCausality();
    if $retVal == result$.assigned then
        $bond$.setExplored(true);
        for each $bondNext$ in $bond$.getNext() do
            if $bondNext$.getExplored() == false then
                $bondsToAssign$.add($bondNext$);
            end if
        end for
    else if $retVal == result$.notAssignable then
        $bond$ = $bondsToAssign$.popSubstituableBond();
        if $bond$ == null then
            return(null);
        end if
    end if
end while
return($bondGraph$);
```

simulation experts need to avoid differential causality, they do not implement acausal implementations of capacitors and inductors and the method does not take them into account at all.

If some components have more than one implementations, the selection of the right implementation for each component is driven by the compatibility of interfaces. To recognize the compatibility, the causality algorithm is extended in order to support various component implementations.

The proposed method can be expressed by the pseudo-code summarized in Alg. 1. First of all, into a new extended bond graph are generated junctions in the same way as in the typical use of bond graphs. Therefore, it must be decided whether the system is mechanical or not, which is important for considering parallel and serial connections. Although components are generated in a similar way as in typical bond graphs, each of those components is enhanced with 1 to n component implementations. Consequently, bonds are generated by adding edges into the graph representation. The extended causality assignment algorithm systematically explores the graph (see the `while` loop in Alg. 1) and tries to assign causality by implementing a depth-first search algorithm. When more than one possibilities of assignments are possible for each bond, the selection is done randomly and it is marked in the extended stack, which is used for driving the search algorithm. This situation occurs in case of junctions with at least three bonds, or components having more than one implementations. If the algorithm explores a bond for which the causality assignment is not possible (i.e., a collision of causality is reached), the search algorithm makes backtracking to the last randomly assigned bond, its assignment is changed and the systematical exploration continues from this corrected bond as a typical depth-first search algorithm.

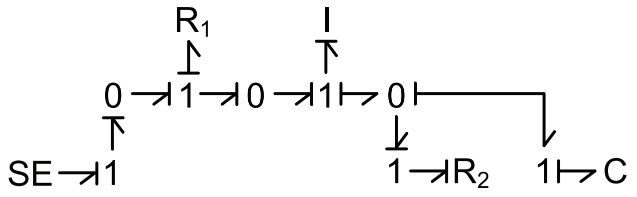

Fig. 4. Bond graph of the electrical circuit, including components, junctions, directions of power, and causality strokes.

The results of this method can be classified as follows:

(1) *A solution exists.* In this case, it is possible to create one or more simulation models for the industrial plant.
(2) *A solution satisfying all conditions does not exist.* The used simulation component implementation sets are proved not to be able to model the specific system. Further component implementations must be added.

Finally, it is necessary to generate the simulation model in a simulator. When generating models, simulation components are instantiated according to the found component implementations. Junctions are generated as sums of variables representing the used power sign conventions. The basic principles of the model generator have been published in Novák and Šindelář [2011a].

5. USE-CASE

The Algorithm 1 is followed in the further text and the extended bond graph for the electrical system introduced in Fig. 1 is generated. The first step is the generation of 0-junctions and 1-junctions, components, and bonds. Till this point, the process does not differ from typical bond-graphs, where it would proceed with assignment of causality strokes according to causality assignment rules for each component. On the contrary, the proposed method cannot assign causality directly. Causality requirements are not defined generally, these pieces of information depend on component implementations. Therefore, it is necessary to load available implementations for each component. In this step, just interface annotations are required, the simulation components themselves are necessary later for the simulation model generation.

In case of the electrical circuit, we assume that the following simulation component implementations are available in a simulation library:

(1) Source of effort $SE = (\text{Flow}_{\text{in}}; \text{Effort}_{\text{out}})$
(2) Resistor $R_A = (\text{Effort}_{\text{in}}; \text{Flow}_{\text{out}})$
(3) Resistor $R_B = (\text{Flow}_{\text{in}}; \text{Effort}_{\text{out}})$
(4) Capacitor $C = (\text{Flow}_{\text{in}}; \text{Effort}_{\text{out}})$
(5) Inductor $I = (\text{Effort}_{\text{in}}; \text{Flow}_{\text{out}})$

The assignment of causality strokes is based on the interface description of available components in the presented method. The extended bond graph for the electrical system is depicted in Fig. 4, which includes components, junctions, bonds, power directions, and assigned causality strokes. Although each component has mapped component implementations behind it, this issue is not reflected in the diagram graphically. Both resistors R_1 and R_2 map their possible implementations R_A and R_B. Since this assignment has been found, the system can be simulated with component implementations available in the library. According to the result of the causality assignment algorithm, we can see that the resistor R_1 should be modeled by the component implementation R_B having flow as input and effort as output, and the resistor and R_2 should be modeled by the component implementation R_A having effort as input and flow as output. Other components have only one available implementations, thus the selections are trivial cases.

The last step of the method is the generation of a simulation model for a signal-oriented simulator. We have previously selected the appropriate component implementations and thus the main intelligence now is to technically implement 0-junctions and 1-junctions. These parts are implemented by "sum" blocks in MATLAB-Simulink, considering the power direction sign convention.

6. CONCLUSION AND FUTURE WORK

Simulation models are useful tools for testing of control algorithms, training of human operators, or estimation of unmeasured states. Unfortunately, their design phase is time-consuming and error-prone. This paper contributes to improve and to semi-automate the design phase of simulation models for large-scale industrial systems.

The proposed method assembles simulation models from simulation component implementations approximating real devices. Having the simulation components, they must be instantiated according to the real plant structure and interconnected according to the real plant device connections. Each component can have more than one implementations, i.e., the mapping between real devices and component implementations can be 1:n. All simulation components are annotated in terms of their interfaces and mappings of signals to flow and effort signals introduced by the bond-graph theory. The suitable simulation component implementations are selected based on the compatibility of interfaces, which is decided via the automatically created extended bond graph of the system.

The vision behind this approach is to enable design and use of simulations for non-experts as well. However, the proposed method does not address system and component parameters in the current implementation.

Future work will be focused on integrating this method into the simulation model designer, which will be the part of the Simulation Integration Framework. Since current implementations of this algorithm have been tested for one-port components, we will focus on extensions and testing for multi-port components. In addition, penalty values for each component implementation will be added so that the search algorithm for causalities and appropriate component implementations to be informed and faster. The sum of these penalty values will evaluate the "cost" of the whole model.

ACKNOWLEDGEMENTS

The authors would like to thank their partners from the Christian Doppler Laboratory for Software Engineering Integration for Flexible Automation Systems for the discussion and feedbacks. This work was supported by the Christian Doppler Forschungsgesellschaft, the Federal Ministry of Economy, Family and Youth, and the National Foundation for Research, Technology and Development - Austria.

This work was supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS12/188/OHK3/3T/13.

REFERENCES

S. Beez, A. Fay, and N. Thornhill. Automatic generation of bond graph models of process plants. In *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2008)*, pages 1294–1301, 2008.

A. J. Blundell. *Bond Graphs for Modelling Engineering Systems*. Ellis Horwood Limited, Chichester, England, 1982. ISBN 0-85312-510-4.

W. Borutzky. Bond graph modeling from an object oriented modeling point of view. *Simulation Practice and Theory*, 7(56):439 – 461, 1999. ISSN 0928-4869.

W. Borutzky. Bond graph modelling and simulation of multidisciplinary systems – an introduction. *Simulation Modelling Practice and Theory*, 17(1):3 – 21, 2009. ISSN 1569-190X.

J. F. Broenink. 20-sim software for hierarchical bond-graph/block-diagram models. *Simulation Practice and Theory*, 7(5 - 6):481 – 492, 1999. ISSN 0928-4869.

P. J. Gawthrop and G. P. Bevan. Bond-graph modeling. *IEEE Control Systems Magazine*, 27(2):24–45, April 2007.

P. Horáček. *Systémy a modely [In Czech language]*. Czech Technical University in Prague, 1999.

P. Novák and R. Šindelář. Integrated design of simulation models for passive houses. In *CEUR workshop proceedings*, volume 821, pages 13–18, 2011a.

P. Novák and R. Šindelář. Applications of ontologies for assembling simulation models of industrial systems. In *On the Move to Meaningful Internet Systems: OTM 2011 Workshops*, pages 148–157, Hersonissos, 2011b. Springer, Dordrecht. ISBN 978-3-642-25125-2.

F. Pêcheux, B. Allard, C. Lallement, A. Vachoux, and H. Morel. Modeling and Simulation of Multi-Discipline Systems using Bond Graphs and VHDL-AMS. In *Proceedings of the International Conference on Bond Graph Modeling and Simulation (ICBGM)*, 2005.

R. Sinha, V.-C. Liang, C. J. J. Paredis, and P. K. Khosla. Modeling and simulation methods for design of engineering systems. *Journal of Computing and Information Science in Engineering*, 1(1):84–91, March 2001. ISSN 1530-9827.

R. Šindelář and P. Novák. Framework for simulation integration. In *Proceedings of the 18th IFAC World Congress*, volume 18, pages 3569–3574, Bologna, 2011. IFAC. ISBN 978-3-902661-93-7.

R. Šindelář and P. Novák. Simulation integration framework. In *10th IEEE International Conference on Industrial Informatics (INDIN 2012)*, pages 80–85, 2012.