A Multimodeling Approach for the Simulation of Energy Consumption in Manufacturing

Thorsten Pawletta[1*], Artur Schmidt[1], Peter Junglas[2]

[1]Research Group Computational Engineering & Automation, Wismar University of Applied Sciences, Philipp-Müller-Straße 14, 23966 Wismar, Germany; **thorsten.pawletta@hs-wismar.de*

[2]PHWT Vechta/Diepholz/Oldenburg, Rombergstraße 40, 49377 Vechta, Germany

SNE 27(2), 2017, 115 - 124, DOI: 10.11128/sne.27.tn.10377
Received: May 25, 2017,
Accepted: June 10, 2017 (Special Issue Review)
SNE - Simulation Notes Europe, ARGESIM Publisher Vienna
ISSN Print 2305-9974, Online 2306-0271, www.sne-journal.org

Abstract. In the design of manufacturing systems the consideration of resource usage, especially energy consumption, is getting more attention. However, the inclusion of all relevant physical processes in a unified modeling approach is a non-trivial task, if detailed analyses are required. The commonly used modeling approach for manufacturing systems is the discrete event modeling technique. However, models of physical processes are often continuous in nature and are modeled using ordinary differential equations or differential algebraic equations. Indeed, the investigation of such physical processes in manufacturing systems often demands a more specific consideration of process control operations, which are favorably modeled using state machines. To combine those different paradigms a multimodeling approach for manufacturing systems is proposed. The approach is illustrated by the example of a production line with an industrial furnace facility.

Introduction

The modeling and simulation of manufacturing systems has been a subject of study for several decades. According to [1], the typical modeling approach is discrete event modeling in this domain. This fact is reflected in the popular simulation tools applied in this field today. However, the situation has recently been changing, because of new aspects that have been taken into account and increasing requirements for accuracy.

One of these aspects is the time-dependent energy consumption of single-process operations, process chains or an overall production system, which becomes important in context with the increasing influence by renewable energy sources and the associated volatile energy availability and energy prices. Approaches for single-process operations, such as in [2], are focused on the energy consumption of single machine operations. They are often based on differential algebraic equations. However, approaches for investigating several machines coupled to a process chain use more abstract models mostly based on discrete event methods, such as in [3, 4]. Today most approaches related to energy processes in manufacturing are only focused on the simulation of energy consumption. In [5] it is emphasized that the energy consumption of a production line (PL) has to be considered in context with production planning and scheduling operations. In fact, the energy consumption then has to be examined together with all the other production performance indicators, such as through-put time, load factors, utilization etc. Hence, considerations regarding the model design and permissible model simplifications are important to master model complexity. For instance, it is necessary to determine how finely grained approximations for continuous energy consumption processes should be. In [6] a discrete-event approximation of those continuous behaviors is discussed, but depending on the research a more accurate approximation can be required.

In [7] a simulator coupling is proposed to execute manufacturing models with mixed discrete event and continuous process behaviour, what we call hybrid system dynamic. This approach is a customized solution and it shows well-known problems of simulator couplings.

A hybrid modeling approach based on the Discrete Event and Differential Equation System Specification (DEV&DESS) in [8] is discussed in [9]. It uses an in-line integration method that schedules the integration time as discrete events. Thus, continuous processes can be modeled using ordinary differential equations and are solved within a discrete event-oriented simulation environment. Both approaches are limited according to the modularity and clear separation between model specification and simulation execution.

This paper is a refined version of [10] and introduces a multimodeling approach for manufacturing systems to overcome those inadequacies. According to the theories in [11, 12], multimodeling means breaking a system into a network or hierarchy structure of individual models. The models may be specified by different dynamical behavior or are described using different methods [13, 14]. Hence, the overall multimodel is from the dynamical point of view often a hybrid model.

The approach is illustrated by the example of a component based PL with an industrial furnace facility that is refined using multimodeling in different layers. Beside the classical production performance indicators, it predicts the time-dependent energy consumption of its main consumer, the furnace facility. The prototypical example is implemented in the *MATLAB/Simulink* [10] environment using different modelling methods, such as entities, events, statecharts, ODEs and DAEs. Some parts of the implementation, pitfalls and simulation results will be presented to strengthen important parts of the approach.

1 Multimodeling Approach for Manufacturing Systems

In the past, methods of modeling and simulation were mainly used for the planning and optimization of the operation of manufacturing systems to determine production performance indicators, such as through-put time, facility utilizations, etc. The usage of discrete event modeling approaches is typical for those investigations. Figure 1 shows such a model of a simple PL implemented using *SimEvents* within *MATLAB/ Simulink*. The PL is reduced for simplicity to a minimal structure composed of: (i) a source component for generating the parts, called entities; (ii) a queue component with FIFO policy; (iii) a server component named furnace; (iv) a sink component for handled parts.

Statistical ports and signal scopes are omitted in the figure. Such a discrete event-based simulation model allows the prediction of the above mentioned performance indicators.

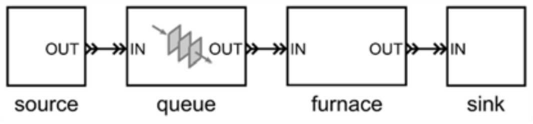

Figure 1: Simple discrete event-based production line model with a furnace component in *SimEvents*.

However, a precise determination of time-dependent energy consumptions of specific components or of an overall system requires a more detailed modeling of relevant components. In the case of the PL, the abstraction of the furnace facility as server in the sense of queuing theory is insufficient. Its abstraction has to be refined. According to [11, 14], states and events at this level of abstraction have to be refined to more accurate events and states at a next lower level. Such refinement leads to a model or network of models at a next lower level, which may be subject to refinement in subsequent steps. The resulting model of such a refined component is called a multimodel, consisting of interacting submodels. The multimodel of a component itself or the overall model often combines several modeling paradigms, and operates with different scales, or is a hybrid system, according to [13], if it includes both continuous-time and discrete-event behavior.

For a refinement of system components in discrete event-based manufacturing models we suggest a layer structure, as illustrated in Figure 2. Each layer represents a specific aspect of the component with well-defined interaction relations between the layers. This approach corresponds to the multilayered architecture, a common design pattern, used in software development [16]. Figure 2 suggests three general layers to refine a manufacturing system component, in which the models of different layers should specify the following characteristics.

- The *material flow* layer describes as highest abstraction the event-based flow of entities (i.e. parts) into and out of the component. It is the basic layer for connecting components to a production line or process chain model, such as illustrated in Figure 1.
  This layer may be refined in further steps using the entity-based modeling method to map internal material flows in more detail or to provide an interface to the other layers.

- The *process control* layer maps the local process control operations of the component. This is especially important for components with several internal manufacturing operations, where the parts are handled in several phases, which may be iterated according to an internal control program or other internal conditions.
- The process *physics* layer implements details of internal process operations that are relevant in the different manufacturing phases, such as energy flows, chemical reactions etc.

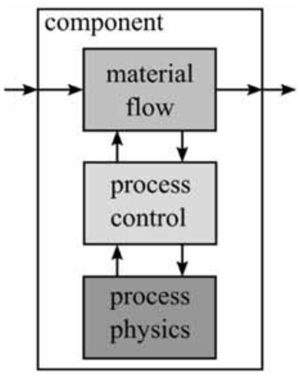

Figure 2: General layer structure to refine a manufacturing system component.

Depending on the specific characteristics of a component, the suggested layers can be arranged hierarchically or as a network of models to form a multimodel. Of course, each layer itself can be refined using models at a next lower level or a layer may be omitted. It is important to adapt the level of abstraction to the questions at hand and only include the processes that are needed to answer them.

Generally, each layer should provide its specific kind of information, which influences the interaction relations between the layers. The material flow layer is mainly concerned with logistic quantities such as waiting times and utilizations; the process control layer gives the order and timing of the different manufacturing phases, which can be useful in the context of other information. The kind of information provided by the physics layer can vary widely depending on the specific component characteristics and implemented details.

The various layers typically demand diverse modeling approaches and, depending on the level of abstraction, their scales are often different. For instance, in the considered example of the furnace component energy flows are accurately modeled based on physical laws, which are described by ODEs or DAEs.

2 Hybrid System Modeling and Simulation

The suggested multimodeling approach for manufacturing system modeling combines several modeling methods to describe different dynamical behaviour, called a hybrid system. Subsequently, we want to highlight some modelling methods and related simulation software.

2.1 Modeling methods

We will consider the modeling methods regarding the suggested layer structure in Figure 2. The material flow layer is often specified using a discrete-event modeling method, which defines abstract entities moving between stationary components and acting on them [8]. The entities are identified in manufacturing systems with workpieces or tools; temporary components move between production facilities.

A convenient method to describe the different manufacturing phases in a production facility, which is the concern of the process control layer, is that of state graphs [17]. The phases directly correspond to the states and the transitions describe the internal process logic. Alternatively, one could again use a process-based approach, wherein the entities denote abstract control tokens.

The actual manufacturing operations, here summarized under the term process physics, are often modeled based on natural or technical laws, e.g. from mechanics, thermodynamics or chemistry. This results in continuous models based on differential equations (ODEs). If the description contains algebraic constraints or the equations are constructed automatically using a physical modeling approach [18], then the mathematical model is enlarged to a system of differential algebraic equations (DAEs).

2.2 Related simulation tools

For the simulation of the logistic and process-oriented aspects of a production system several discrete event-based simulation environments exist and are in wide industrial use, such as *Arena* [19] and *Plant Simulation* [20]. Usually these programs lack algorithms such as ODE or DAE solvers to cope with continuous system specifications. This is why different simulators are coupled to solve such problems, such as in [7]. The introduced multimodel structure supports such simulator couplings, but it cannot rectify its general problems.

Instead, one should use a software environment that is capable of hybrid modeling and simulation. Such an environment, increasingly used for manufacturing system simulation, is *AnyLogic* [21]. It offers system dynamics, discrete event and agent-based methods. However, the mapping of complex ODEs to system dynamics diagrams is quickly confusing and physical modeling techniques, according to [18], are not supported. As a consequence, it may be only a useful choice for multimodeling problems with relatively simple continuous process physics.

Another widely used software supporting multimodeling, although less popular in the manufacturing simulation domain, is the Matlab/Simulink environment. Originally designed for the simulation of continuous systems using the signal flow paradigm, it can be extended using additional toolboxes and blocksets to include discrete, discrete event or physical modeling features: (i) The *SimEvents* blockset enables discrete event modeling based on the entity approach; (ii) *Stateflow* provides state chart modeling techniques; and (iii) *Simscape* expands the continuous tool chest with physical modeling features. This software environment provides the widest range of features for multimodeling today and it is already used and accepted in other engineering domains. Hence, it will be used in the following to illustrate and validate the suggested multimodeling approach by implementing some concrete examples.

An alternative choice could be to use a *Modelica* based solution [18] with the additional packages described in [22, 23] or *Ptolemy* [24] with the *OpenModelica* extension according to [25]. However, both *Modelica* and *Ptolemy* are even more unknown than *MATLAB/Simulink* in the manufacturing system community.

3 Basic Application to a Manufacturing System Component

To illustrate the introduced approach the furnace component of the PL model in Figure 1 will be analyzed for multimodeling and a set of models with different levels of abstraction will be designed.

3.1 Multimodeling of furnace component

Our objective of multimodeling is the refinement of the furnace component to investigate its time-related energy consumption. Industrial furnaces are widely used in metalworking processes and are one of the most extensive energy consumers in manufacturing systems. In addition, their internal operation is generally rather complex, making this an ideal example for refinement using different modeling approaches. The operation of such a furnace is patterned in the following after the descriptions in [9, 26].

When parts arrive at the furnace, they are collected until a given batch size is reached. A complete batch then enters the furnace, is processed and leaves the furnace. Then, the batch will probably be resolved for processing the parts by other facilities. The refinement of such internal processes of a component is part of the material flow layer, as demonstrated in Figure 2.

Moreover, the heat treatment itself consists of several phases that are implemented by a local control; these are mapped and refined at the process control layer. Figure 3 shows an example of such a control with six operation phases.

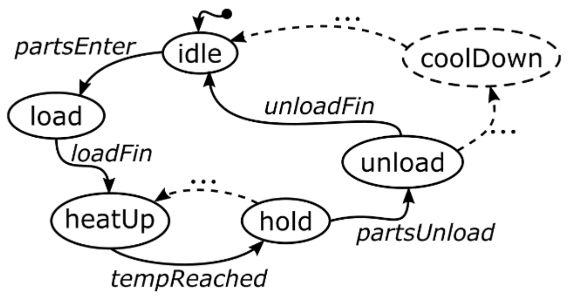

Figure 3: State graph describing a local control of furnace operations.

The idle phase spans the times before parts have entered and after all parts have left the furnace. During the load phase parts enter the furnace. In the heat-up phase the furnace is heated until a given temperature is reached; this is then held constant during the hold phase. According to the requirements, the heat-up and hold phases can be iterated several times with different temperatures. Finally, the parts leave the oven in the unload phase. An additional cool-down phase may be included either to make sure that the parts leave the furnace with a moderate temperature or to describe a shutdown of the furnace.

Since the focus of our example study lies on the energy consumption, the heat flows in the furnace have to be considered in more detail, and must be refined at the process physics layer (Fig. 2). The energy source is the power supply of the actual heater. From here the heat flows mainly through convection and radiation processes to the parts and to the internal structures and the casing of the furnace. During the heat-up and hold phases losses are mainly due to conduction through the casing into the environment, while in the load or unload phases additional losses are caused by the open doors. Because of the complicated geometry, the physical details, especially of the convection processes, are also rather complicated. However, for the estimation of the total heat flows common approximative methods usually give quite accurate results. Concrete mathematical models for specifying the process physics will be considered afterwards in context with their prototypical implementation.

Based on the previous considerations, Figure 4 shows the multimodel structure with defined interfaces for the refinement of our furnace component based on the layers introduced in Figure 2. In this abstraction it consists of three interacting models: (i) MF for the material flow; (ii) PC for the process control; and (iii) PP for the process physics. The labels (B) and (C) are not of interest at this point.

The models have to communicate in several ways:

- MF receives parts from the external input port <1> and sends to the PC the number of parts that have entered the MF.
- When the batch size is reached, the PC starts the PP, which models the different manufacturing phases during the operation of the furnace.
- During the operation of the furnace the PC signals the current manufacturing phase to the PP, which adapts the internal heat flows accordingly. This can mean changing the supplied heat between the heating and holding phases or increasing the losses due to the doors being open while loading or unloading.
- In return the PP sends the current temperature values of the furnace and the parts to the PC, which uses them to determine whether the heat-up phase or the optional final cool-down phase of the furnace is complete (Fig. 3).
- When the manufacturing phase unload (Fig. 3) is finished, the PC sends a *leaving* signal to the MF, which accordingly forwards the processed parts to its external output port <2>.

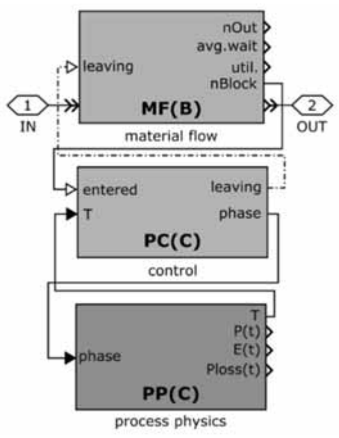

Figure 4: Basic multimodel structure with defined interfaces for the refinement of the furnace component.

Each model has its specific set of parameters and output quantities: The MF defines the batch size and logistic properties such as average waiting time and machine utilization; the PC gets the heating program and outputs the time in the different manufacturing phases. The PP needs a lot of physical parameters for the calculation of the heat flows and provides the power requirements during the process phases as well as the total energy consumption.

3.2 Design of a model library for the furnace component

Based on the basic multimodel structure in Figure 4, a set of models for each layer has been implemented and organized in a library (Fig. 5). The several models use different modeling methods or have varying levels of detail. They are labeled with two letters denoting the layer and a third letter in brackets giving the level of detail in ascending order, with (A) being the simplest model:

- The basic material flow model MF(A) uses only a simple server, while MF(B) explicitly contains an input tray, where the batch is compiled.
- The process control subsystem PC(A) uses a simple entity-based model, implemented in *SimEvents*, to describe one pass through the four basic process phases and ignoring idle and cool-down phases. PC(B) enables a repetition of heat-up and hold phases according to its heat program parameter, implemented as an entity-based model in *SimEvents*. Additionally, PC(C) adds a cool-down phase, but because of the more complex control logic it is implemented using the state machine approach with *Stateflow*.

- The process physics model PP(A) uses only the internal oven temperature and a simple formula for the global losses, while PP(B) adds the temperature of the parts, the heat transfer between oven and parts and additional losses during the load and unload phases. Both use standard *Simulink* blocks to implement the corresponding differential equations. Finally, PP(C) employs physical modeling, which is modeled using *Simscape* for the same physical processes as PP(B). This makes the physical model structure more transparent and easier for engineers to expand.

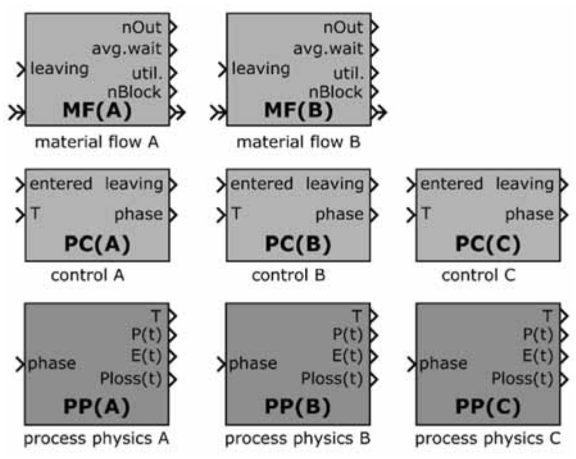

Figure 5: Library with models for composing several multimodels with different levels of detail for the furnace component.

4 Some Modeling & Implementation Details

In the following, three multimodel variants for the furnace will be described in more detail: (i) a very basic multimodel named *ovenBAA* – the capital letters stand for the composition of MF(B), PC(A) and PP(A) models; (ii) a medium complex multimodel *ovenBCA*; and (iii) the most complex multimodel variant *ovenBCC*. While this section is devoted to some implementation details of the single models used in the three multimodels, the next one will discuss some simulation results. The implementation was carried out using the *MATLAB/Simulink* environment and related tools.

4.1 Material flow model

All of the examples considered here use the extended model MF(B) for mapping the internal material flow.

The entity-based model structure of MF(B) (Fig. 6) consists of two simple servers: the first for the *input tray*; and the second (*N-Server*) for the furnace proper. Both hold incoming entities (i.e. parts) up to the given batch size, until their succeeding gates open to transfer the entities to the next stage. The intermediate *Gate* guarantees that a full batch is always delivered to the furnace; the final *Release Gate* is triggered – by the model at the process control layer (Figs. 7, 8) – at the end of the unload phase.

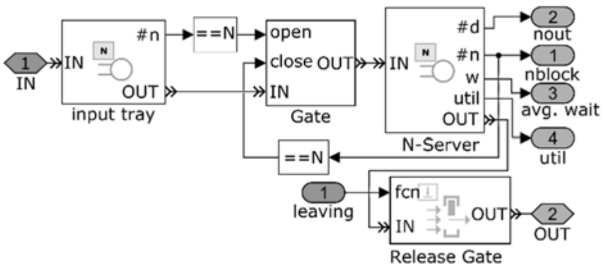

Figure 6: Discrete event-based model of MF(B) using *SimEvents*.

4.2 Process control models

The process control model starts when the number of parts that have entered at the material flow layer is equal to the batch size. In the process control model PC(A) (Fig. 7), implemented using *SimEvents*, this leads to the creation of a control entity that passes through a line of servers denoting the different phases. Except for one, all servers simply have a fixed processing time; only the heat-up phase is different. Here the entity is held in a server until the current oven temperature, which is computed in the model at the physics layer (Figs. 9, 10), has reached the given temperature T_{set}. After the unload phase, the control entity is destroyed and the wake-up signal (*trigger*) is generated, which opens the *Release Gate* in the model at the material flow layer (Fig. 6).

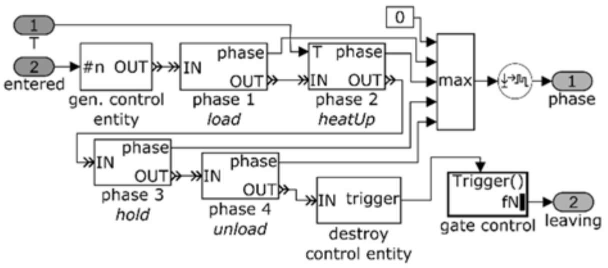

Figure 7: Discrete event-based model of PC(A) using *SimEvents*.

The much more elaborate process control model PC(C) (Fig. 8) incorporates all six phases, pictured in Figure 2, as well as possible repetition of the heat-up and hold phases. According to the theory in [8], it is a hybrid DEV&DESS model. It is modeled based on the state machine approach, but it includes continuous state event handling and has been mainly implemented using *Stateflow*. The two auxiliary subsystems (*schedTEvent*, *checkTemperature*) create events when the waiting time has changed or the heat-up or cool-down temperature has been reached. The model PC(C) is essentially an adapted version of a model described in [9].

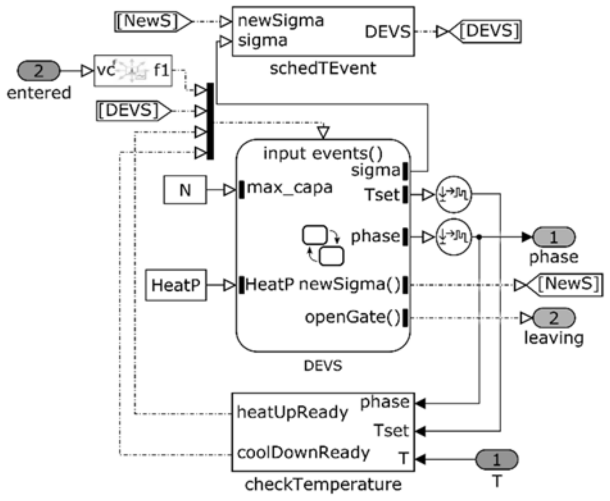

Figure 8: Discrete event-based model with continuous state event handling of PC(C) mainly using *Stateflow*.

4.3 Process physics models

The basic process physics model PP(A) (Fig. 9) uses a simple power balance to compute the change of the oven temperature T_0:

$$C_0\dot{T}_0 = P_{heat} - P_{loss} \tag{1}$$

where C_0 is the total heat capacity of the oven. The power loss is computed with Newton's simple law of cooling:

$$P_{loss} = k_a(T_0 - T_e) \tag{2}$$

where T_e is the temperature of the surroundings and k_a a constant that subsumes all convective and conductive processes. The heating power is assumed to be constant during the heat-up phase and to match the losses during the hold phase:

$$P_{heat} = \begin{cases} P_H & heat-up \\ k_a(T_0 - T_e) & hold \end{cases} \tag{3}$$

For the computation of the total power demand of the furnace, a constant power P_0 is added, which subsumes all non-heating processes, such as a base load or the power electronics.

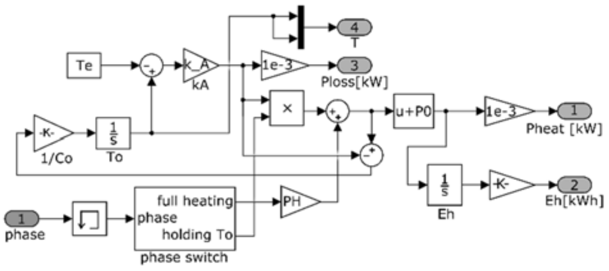

Figure 9: ODE-based model of PP(A) using Simulink.

The model PP(C) (Fig. 10) uses physical modeling based on *Simscape* to incorporate much more physical details, such as convection and radiation from the furnace – during load and unload phases – to the environment. The governing differential algebraic equations are not built up explicitly here; instead, the physical components such as heat capacities and various kinds of heat flow are represented directly, which makes the physical structure of the model much clearer.

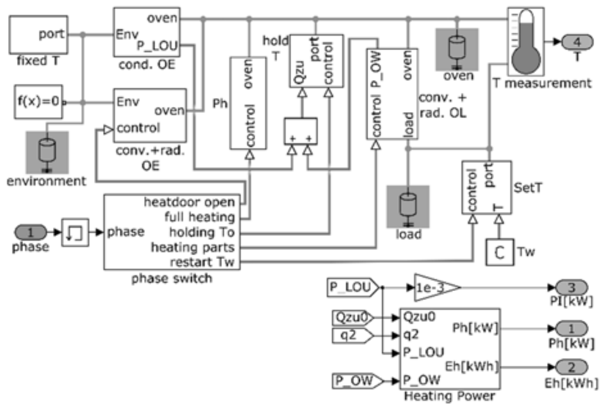

Figure 10: Physical model (DAE) of PP(C) using *Simscape*.

4.4 Implementation pitfalls

The extension of *MATLAB/Simulink* by several packages (blocksets) is necessary to make a multi-paradigm model possible, but it also leads to small inconsistencies, which have to be overcome. A minor nuisance here is the large number of different ways used to express values and signals in the packages used: Simple scalar values can be time-based in *Simulink*, event-based in *SimEvents* or physical-valued in *Simscape*. An event can be defined by a sample time hit, a rapid value change (*edge*), a special trigger signal or a function call.

To clearly define a common interface for the interacting models, their external values are fixed in the following way: The temperature and phase values are time-based; the number of parts is event-based; and the *leaving* signal is a function call. If the internal implementation of a block generates differing types, then one has to use one of the many converter blocks to get the proper kind of signal, e.g. the round *Event to Timed Signal blocks* that can be seen in Figures 7 and 8.

This problem is especially annoying in the physical modeling environment. The large number of necessary converters and reference points clutters the model and destroys the clear physical structure. Hiding them in subsystems is an obvious way to regain an ordered visible representation of the underlying physics model.

5 Exemplary Simulation Results

Using the introduced approach, a large number of simulation studies is possible to calculate multifaceted performance indicators. We will restrict our exemplary consideration to the energy consumption of the furnace that is the main consumer in our simple production line. Moreover, we will investigate the costs of refinement relating to the simulation run time by comparing different multimodels, which use different modeling paradigms or are based on a different level of detail.

5.1 Results relating to the energy aspect

In the exemplary experiment 24 parts were processed using a batch size of six. After the heat-up phase their temperatures were held first to 400 °C for 40 minutes, then to 800 °C for 30 minutes. Figure 11 shows some simulation results related to the energy consumption, calculated using our most complex multimodel variant *ovenBCC*: the temperatures of the furnace and parts, the power consumption of the furnace, the accumulated used energy and the current internal manufacturing phase, each as functions over time.

The results are similar to those presented in [9], but *ovenBCC* incorporates more details, especially of the physical model layer. The other two example models *ovenBAA* and *ovenBCA* basically reproduce the results from [9], since they are based on the same physical model assumptions. For comparison, Table 1 shows the total energy use E_1 at the exit of the last part and E_2 at the end of the simulation; Figure 12 displays the power consumption for the three models.

Model	E_1 [kWh]	E_2 [kWh]
ovenBAA	71.3	108.8
ovenBCA	69.0	106.6
ovenBCC	82.6	117.6

Table 1: Comparison of the total energy consumption.

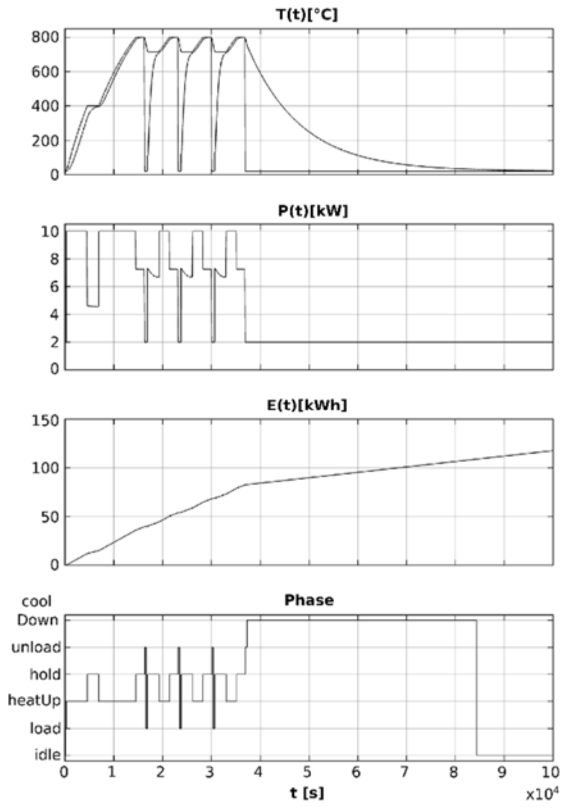

Figure 11: Simulation results of model variant ovenBCC.

The higher energy needs in variant *ovenBCC* are due to the heating of the parts, which enter the oven with the low temperature of the environment – an effect that has not been taken into account in the other models. The small difference between the two simpler multimodel variants results from the different timing of the phases, as can be seen from Figure 12.

It is interesting to note that the total energy results only differ by 10 % - 15 %. If this level of accuracy is sufficient for the question at hand, e.g. for a global assessment of a complex production line, then one can use one of the simple physics models. However, Figure 12 shows that for a detailed examination of the power needs during the individual phases one has to stick to the complexities of multimodel variant *ovenBCC*.

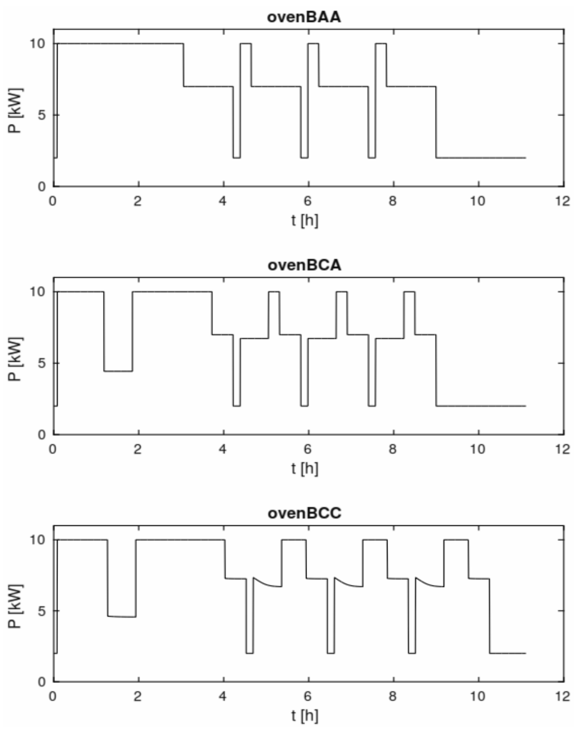

Figure 12: Comparison of the power results.

5.2 Comparison of run times

How much do we pay for the additional level of detail in the complex multimodel? For a comparison of run times, the number of parts was raised to 600 and three runs were performed for each multimodel. To get rid of loading and compile times, the means of only the last two results were taken, leading to the values in Table 2.

Model	Run Time [s]
ovenBAA	25.1
ovenBBA	27.7
ovenBCA	20.7
ovenBCB	39.3
ovenBCC	49.8

Table 2: Comparison of run times.

The results show that the complex physical model using *Simscape* in multimodel *ovenBCC* (line 5) costs a factor of two in execution time, which can be reduced in this case to a factor of 1.6 by replacing DAE-based physical modeling with standard ODE methods using Simulink (*ovenBCB*, line 4). However, we see essential differences in execution time for multimodels with the same simple process physics model PP(A) (line 1-3), which are surprising.

The multimodels *ovenBBA* (line 2) and *ovenBCA* (line 3) use process control models of nearly the same complexity, but in *ovenBBA* implemented with *SimEvents* and in *ovenBCA* with *Stateflow*. Moreover, the multimodel *ovenBCA* (line 3) uses the more complex process control model PC(C), than *ovenBAA* (line 1) with the PC(A) model that was implemented with *SimEvents*. Apparently, the implementation of *SimEvents* has some potential for optimization.

6 Conclusion

The introduced multimodel approach supports the component-oriented refinement of manufacturing system models, using various modeling methods and models with different levels of abstraction. Generally, it suggests a refinement of components based on the three logical layers material flow, process control and process physics.

The first layer maps the internal material flow. It delivers an external input and output interface for connecting with other components in a manufacturing system model and an internal interface for communication with the second layer. The dynamic behavior at this layer is generally discrete event-based. In the second layer local process control operations are modeled. This layer provides event-based control inputs for the other two layers, but it could be necessary to handle state events of continuous values as well. In the third layer specific process operations should be mapped. The dynamic behavior at this layer can vary significantly and particularly depends on the level of abstraction. Of course, depending on the characteristics of a system component and the intended level of abstraction, layers may themselves be omitted or refined using several interacting models.

The paper illustrated the approach using the example of a production line with an industrial furnace as the main component. According to the layers, models with different levels of abstraction have been implemented for the furnace, which could be aggregated to a multimodel. Then, the energy consumption of the furnace was investigated with three different multimodels and the simulation run times were measured. Hence, accuracy effects and computing costs could be compared.

The approach opens many new ways for investigating manufacturing systems, but the complexity is increasing. Hence, in the next step it should be combined with metamodeling techniques such as those considered in [27].

Acknowledgments. Thorsten Pawletta and Artur Schmidt thank the German Research Foundation for funding this research (no. PA 631/2-2). Peter Junglas is grateful for the hospitality extended to him by Tom Schramm at the Department of Geomatics, HCU Hamburg, during his work on this project.

References

[1] Lefeber E, Rooda JE. Modeling and Analysis of Manufacturing Systems. In: Fishwick PA. editor. *Handbook of Dynamic System Modeling*, Boca Raton: Cpaman & Hall/CRC, 2007, p. 34_1-34_20.

[2] Eisele C. *Simulation-based optimization of the electrical consumption of metal-cutting machine tools (German).* [dissertation], TU Darmstadt, Germany, 2014.

[3] Weinert N. *An approach for the planning and operation of energy-efficient production systems (German).* [dissertation], TU Berlin, Germany, 2010.

[4] Larek R. *Resource-efficient design of manufacturing process chains by simulation and numerical optimization (German).* [dissertation], Univ. Bremen, Germany, 2012.

[5] Schrems S. *A method for the model-based integration of machine-specific energy demand in production planning (German).* [dissertation], TU Darmstadt, Germany, 2014.

[6] Larek R, Brinksmeier E, Meyer D, Pawletta T, Hagendorf O. *A discrete-event simulation approach to predict energy consumption in machining processes.* Production Engineering Research and Development, Berling: Springer Pub., 2011, p. 575-579.

[7] Peter T, Wenzel S. Simulation-based Planning and Evaluation of Energy Efficiency for Production Systems in Car Manufacturing (in German). *Simulation in Production and Logistics, Fraunhofer Pub.*, Stuttgart, Germany, 2015, p. 535-544.

[8] Zeigler BP, Kim TG, Prähofer H. *Theory of Modelling and Simulation.* SanDiego: Acad. Press, 2nd ed., 2000, 510 p.

[9] Schmidt A, Pawletta T. Hybrid modeling of manufacturing process chains and their energy consumption in a discrete event simulation environment (German). In: Wittmann, J. editor. Proc. *ASIM 2014-22. Symp. Simulationstechnik*, Sep 2014: Berlin, Germany, p. 109-116.

[10] Schmidt A, Pawletta T, Junglas P. A layered structure for modeling manufacturing processes with the inclusion of energy consumption. In: Proc. *ASIM Symp. STS/GMMS*, Mar. 2016: Lippstadt, Germany, p. 155-164.

[11] Fishwick PA. *Simulation Model Design and Execution: Building Digital Worlds.* Upper Saddle River: Prentice-Hall Inc., 1995, 432 p.

[12] Park M, Fishwick PA, Lee J. Multimodeling. In: Fishwick P.A. editor. *Handbook of Dynamic System Modeling*, Boca Raton: Cpaman & Hall/CRC, 2007, p. 14_1-14_28.

[13] Mosterman PJ. Hybrid Dynamic Systems: Modeling and Execution. In: Fishwick PA. editor. *Handbook of Dynamic System Modeling*, Boca Raton: Cpaman & Hall/CRC, 2007, p. 15_1-15_26.

[14] Shephard MS, Seol ES, Dale BF. Towards a Multimodel Hierarchy to Support Multiscale Simulation. In: Fishwick PA. editor. *Handbook of Dynamic System Modeling*, Boca Raton: Cpaman & Hall/CRC, 2007, p. 12_1-12_17.

[15] The MathWorks. *Simulink: Simulation and Model-Based Design.* Online: www.mathworks.de/products/simulink/ (called 2015-11-11).

[16] Buschmann F, Meunier R, Rohnert H, Sommerlad P, Stal M. *Pattern-Oriented Software Architecture: A System of Patterns.* vol. 1. New York: John Wiley & Sons, 1996, 476 p.

[17] Harel D. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8, 1987, p. 231-274.

[18] Fritzson PA. *Principles of Object-Oriented Modeling and Simulation with Modelica 3.3.* New York: Wiley & Sons, 2015, 1256 p.

[19] Kelton WD, Sadowski R, Zupick N. *Simulation with Arena.* 6th ed. New York: McGraw-Hill, 2014, 656 p.

[20] Bangsow S. *Tecnomatix Plant Simulation: Modeling and Programming by Means of Examples.* Springer: Berlin, 2016, 713 p.

[21] Grigoryev I. *AnyLogic 7 in Three Days*. 2nd ed. Noth Charleston: CreateSpace Independent Pub., 2015, 256 p.

[22] Elmqvist H, Gaucher F, Mattsson SE, Dupont F. State Machines in Modelica. In: Otter M, Zimmer D, editors. *Proc. 9th Int. Modelica Conf.*, Sep 2012: Munich, Germany, p. 37-46.

[23] Sanz V, Urquia A, Cellier FE, Dormido S. System modeling using the Parallel DEVS formalism and the Modelica language. *Simulation Modelling Practice and Theory*, 18(7), 2010, p. 998–1018.

[24] Ptolemaeus C. *System Design, Modeling, and Simulation using Ptolemy II.* Ptolemy.org, 2014, 690 p.

[25] Mirzaei M. *Integration of OpenModelica into the Multiparadigm Modeling Environment of Ptolemy II.* Master Thesis, Univ. of Linköping, Sweden, 2014.

[26] Heuer V, Löser K. Energy optimization of thermal-chemical vacuum processes and aggregates in large-scale productions (German). *Elektrowärme International*, 69, 2011, p. 261-268.

[27] Pawletta T, Schmidt A, Zeigler BP, Durak U. Extended Variability Modeling Using System Entity Structure Ontology within MATLAB/Simulink. In: *Proc. SCS Int. SpringSim/ANSS*, Apr. 2016: Pasadena/CA, USA, 2016, p. 62-69.