

Chapter 5

Implementation of a Multiscale Core Model

The concept of a multiscale simulation for battery production systems is centered around a process chain model which acts as a coordinator for other models. Such process chain model determines the activities within a production system and derives inputs for detailed models of system elements. Consequently, this model is a key component of multiscale simulations.

For this reason, an adaptable multiscale core model was developed and implemented, which can be configured to any kind of process chain, contains essential sub-models, which can be extended by external simulation models. The intention behind this model is to provide a flexible software tool which contains the minimum functionality needed for the realization of specific multiscale simulation applications. To serve this purpose, the core model allows simulating the material flow of processing units and contains generic models for machines, product characteristics and processes, lighting, and workers. Furthermore, the model has interfaces for connecting external simulation models such as detailed process models, TBS system models, or a building model. As an example, a model for a compressed air generation system was implemented which is connected to the core model via a middleware software. This chapter presents the architecture of the core model (Sect. 5.1), its components (Sect. 5.2), the compressed air generation system model (Sect. 5.3), the synchronization concept for the connection of external models (Sect. 5.4), as well as the verification of the overall model (Sect. 5.5).

5.1 Core Model Architecture

The model architecture describes the components and their interactions within the core model. The main function of the core model is the simulation of the flow of

processing units through a virtual shop floor consisting of different machines¹. Processing units are of a certain product type which specifies the sequence of required production steps, each of which is allocated to one or several machines. Furthermore, processing units belong to jobs which are characterized by a job type and a quantity of final products. During production time, processing units search for suitable machines, move to these machines and are processed until the last production step is finished (see Sect. 4.3.2). Processing units and machines are modeled as agents with individual properties based on agent classes. This allows the replication of agents and a flexible definition of process chain layouts.

The core model also contains models for lighting and workers (agents) according to the multiscale model structure shown in Fig. 4.28. All system elements are assigned or dynamically linked to building zones. The core model aggregates all variables on the main level – and if necessary for each building zone –, provides data to interfaces for external models, and determines performance indicators. These indicators are visualized within a cockpit for result evaluation during simulation runs. The model also has an export function to make the results available after simulation. Figure 5.1 drafts the model architecture by showing the different sub-models and components with their inherent variables and information, as well as the interfaces for external models.

Based on this architecture, the core model is implemented within the software AnyLogic. AnyLogic is a hybrid simulation tool which enables combining DE, DS,

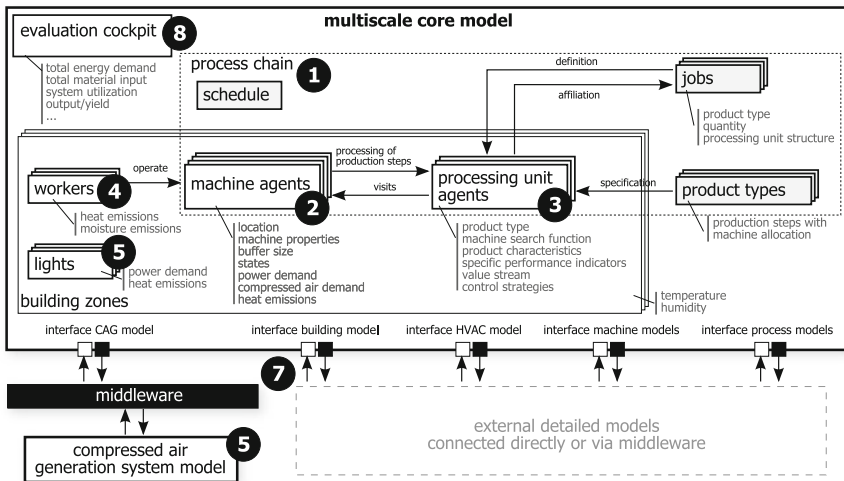


Fig. 5.1 Architecture of the multiscale core model with interfaces for external models

¹As stated in the concept derivation, machines can also be substituted by workstations which do not utilize any equipment.

AB, and SD simulation within one model. The software is based on the programming language Java.²

5.2 Core Model Components

As shown in Fig. 5.1, the main components of the core model are sub-models for jobs, product types, processing units, machine agents, workers, lights, building zones, the evaluation cockpit, and schedules. This section describes these components.

5.2.1 Jobs and Product Types

Jobs and product types are objects used for defining and controlling the sequence and routing of processing units. Jobs correspond to requests for the production of a quantity of a product type. As an example, a job is created for the production of 1000 battery cells of a certain type or for the assembly of 200 specific modules. Depending on the product type, different processing units are required to simulate the routing of a job as well as to track product characteristics along the process chain. For this reason, job objects define the type and quantity of processing units which are required to simulate the production of a desired quantity of final products.³ In the process chain model, a job is characterized by a job type, the desired quantity of final products and the associated product type(s). Three different job types have been predefined. The first is a generic job type which can be used to imitate the production of one product type in a certain quantity. This job creates a number of processing units corresponding to the desired quantity. Processing units can be treated as single product units or as batches.⁴ This allows using the generic job type if identical product units should be produced. The second job type addresses the production of battery cells. A job defines two processing units, each of which represents an electrode batch. When both batches are completed, the job defines processing units according to the desired quantity of cells. A job is completed when the last related processing unit (cell) is finished. It is necessary to specify the product types for anodes, cathodes, and cells. The third job type is usable for the combined assembly of modules and systems. The desired quantity of systems has to be specified to derive the required number of modules.⁵ As soon as enough modules are finished to complete a system, a processing unit of a system may start production. Table 5.1 summarizes the characteristics of the three job types.




²Further information about AnyLogic can be found on www.anylogic.com.

³As an example, the production of 100 cells requires two processing units representing the electrode batches and 100 processing units for representing each cell.

⁴That means that the number of processing units is different for single unit or batch production. Example: If 100 product units should be produced in batches of 20 units, required are five processing units. A generic job would have to be defined with a quantity of five.

⁵The number of modules per system has to be specified in the system's product type definition.

Table 5.1 Characterization of job types regarding quantity, product types (anodes (a), cathodes (c), cells (ce), modules (m), and systems (s)) and related processing units

Job type	Quantity (Q)	Product type (PT)	Processing units
Generic	Q	PT	
Cell	Q_{cell}	$PT_{anode}, PT_{cathode}, PT_{cell}$	
System	Q_{system}	PT_{module}, PT_{system}	

The definition of product types allows the differentiation of electrodes, cells, modules, and systems. Moreover, a product type can specify different variants of these products such as electrodes of different materials or cells with different numbers of layers. In this model, product types specify processing units regarding their production step sequences, the allocation of production steps to machines, as well as processing times and material inputs per production step.

In summary, the job and product type objects make it possible to configure the model to simulate a specific process chain. It can be used to simulate different production stages – according the defined system boundaries – separately or within a combined process chain.


5.2.2 Processing Unit Agents

Processing units are modeled as agents based on a processing unit agent class. On process chain level, processing units are displayed by simplified interactive icons which indicate the ID and product type. Moreover, a dynamic circle indicates the progress during processing. Within each agent, the icon contains further information about the current state, accumulated material cost and direct embodied energy (DEE), as well as an illustration of the current product state. Figure 5.2 shows an exemplary icon of an electrode processing unit within the first mixing process, still in raw material state.

This agent class contains a state chart for representing the states of a processing unit during production, parameters for defining the product specifications, the related job and assigned product type, variables for product characteristics, input materials, and direct embodied energy as well as an algorithm for selecting the next suitable machine. The state chart of a processing unit indicates if a processing unit is already started, searches for a next machine, moves to a machine, is at a machine, or is completed after the last production step. These states allow to control and monitor the behavior of processing units within the process chain. When a processing unit is created and initialized, it searches for the first machine. For this task, the algorithm is implemented according to the shortest throughput time control strategy.⁶ This

⁶Other control strategies and selection criteria are possible (see Sect. 4.3.2).

Fig. 5.2 Icon of an electrode processing unit (here: processing unit nr. 1 of product type 2)



New machine found?: ●
Number of states: 5
Current state: 0
Material cost [€]: 17.354
DEE [kWh]: 691.617

visible on upper level not visible on upper level

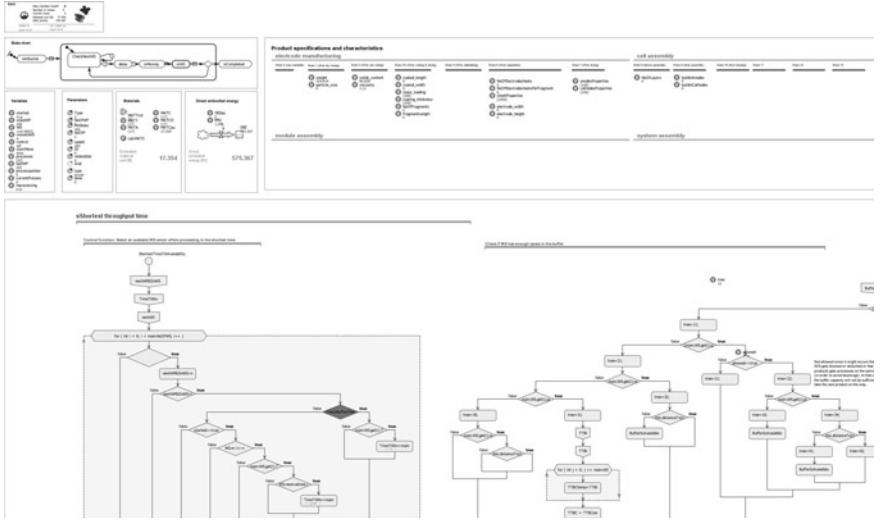


Fig. 5.3 Screenshot (for illustration only) of processing unit agent class content: *top left* icon, state chart, parameters and variables; *top right* product specification and characteristics per state; *bottom* algorithm (action chart) for machine selection

algorithm requires information from machine agents such as the location and the time until availability.⁷ After a machine is selected, the processing unit agent moves to the machine. When the processing is finished, the machine agent sends a signal to the processing unit which searches for the next machine. If no available machine can be found, the processing unit blocks the current machine. In this case, the search algorithm stays active until a machine becomes available. This logic corresponds to the processing unit flow illustrated in Fig. 4.9. Figure 5.3 presents a screenshot of the processing unit agent.

The flow of processing units is visualized by a value stream map. For each production step, the model creates a data field which is filled with related performance indicators such as processing time and waiting time. This allows analyzing if, where, and for how long a processing unit had to wait prior to processing. Also it is possible to determine the overall time of production for each processing unit. Each agent con-

⁷If external machine models are used, the remaining processing time might not be known or predetermined but a result of the simulation. In this case, the algorithm considers a very long time until availability. As a result, a machine is usually not selected unless no other machine is available.

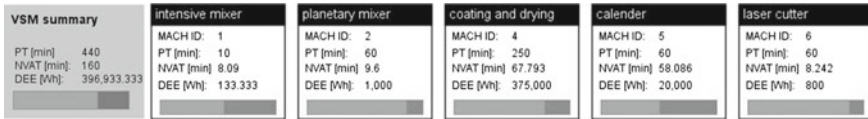


Fig. 5.4 Screenshot of an exemplary VSM representation within a processing unit agent

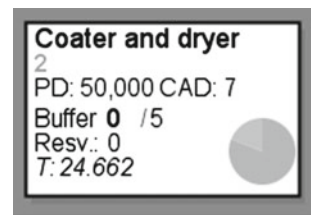
tains a VSM representation showing the history of processing. Figure 5.4 presents a screenshot of an exemplary VSM representation of an processing unit for an electrode batch. The figure shows the sequence of processes for an electrode job with related processing data as well as a summary of derived indicators such as lead time. The bars indicate the productive and non-productive time shares. Furthermore, the representation shows the direct energy demands from processing at each machine.

5.2.3 Machine Agents

Machines are also modeled as agents based on a machine agent class. On process chain level, machines are displayed by interactive icons which indicate machine name, number, current state (by the color of the border), the current demands, machine temperature, the number of processing units within the buffer, and the time shares per state. Figure 5.5 shows an exemplary screenshot of a machine icon.

The dynamic characteristics of a machine are determined decentralized within a machine agent. On the inside, each machine agent consists of a flow chart for modeling the processing unit agent flow through the machine, a state chart for modeling of possible states and their transitions, parameters for specifying machine properties (e.g. location, buffer size, etc.)⁸, and variables for describing the machine characteristics (e.g. power demand, compressed air demand, and heat emissions). Elements of the AnyLogic enterprise library are used to model the processing unit flow through a machine. These elements allow to control the routing of processing unit entities entering a machine to a buffer (queue element) which is closed (by a hold element) in case the machine is still ramping-up. After ramp-up, a processing unit is routed to the actual process (delay element) and – after the processing time – the processing unit

Fig. 5.5 Representation of a machine agent on process chain level



⁸Machine parameters can be externally configured in spreadsheets and imported to the model.

leaves the machine if a new machine (or a buffer) is available for further processing. If a processing unit does not find a next machine, the processing unit blocks the current machine. If a machine failure occurs, the state chart switches to the failure state and it is assumed that the machine behaves as in idle state. The processing of the current processing unit is resumed after the failure is resolved.⁹

The flow chart communicates with the state chart which describes possible machine states according to the definition of the generic machine model. The state chart further determines the demands for electric power and compressed air. Furthermore, the state chart allows to determine the time shares of different operational states such as ramp-up time, idle time, and processing time. This also enables to derive the utilization of each machine. The power demand is input to the calculations of machine temperature and heat emissions which also require certain machine parameters (mass, surface area, etc.) and the current building zone temperature.

In addition, machine agents contain an algorithm for determining the time until a machine is available for processing of a next processing unit. This algorithm considers various factors such as the current state and buffer level, remaining processing time, future processing times of buffered processing units, and remaining movement times of processing units which are currently moving to a machine. Figure 5.6 shows a screenshot excerpt of the machine agent class content.

Machine agents can further contain process models and a set of associated process parameters. Process models describe how a process modifies particular product characteristics of processing units. Depending on the specific process, process models can be realized by functions which calculate the resulting product characteristics based on specifications and existing characteristics of a processing unit. Alternatively, if the processing results vary over time or different sequential activities can be distinguished, it is also possible to use timed state charts to define the execution of functions in more detail.¹⁰

Finally, machine agents are responsible for the connection of external machine models. For each machine, a machine agent has to be placed within the process chain. If an external machine model is used, the machine agent coordinates the execution of the external model. For this purpose, the state chart contains a section which controls the interaction with external models. This is important since the external model has to behave according to the defined states. Thus, the state chart of the machine agent is used to send and receive coordinating signals. The machine agent also receives values for power and compressed air demand from the external model. Table 5.2 lists the variables for communication with an external model.

This set of variables may be extended for specific purposes. For example, if an external machine model simulates the detailed thermal behavior of machine compo-

⁹Different failure behavior could be implemented.

¹⁰This might be relevant for processes with long processing times. As an example, the process characteristics during coating and drying may vary over time if the machine temperature increases or the coating thickness increases due to variations in the coating device. The characteristics of the coating layer may vary over time and may differ for different fragments of the layer. In this case, a state chart may repeatedly trigger the process model function.

Fig. 5.7 Representation of a worker agent on process chain level



5.2.4 Workers

Within the core model, it is assumed that each machine needs at least one worker for operation. Workers are modeled as agents.¹¹ As soon as a processing unit has selected a machine which is currently turned off, this machine agent requests the required number of workers. On process chain level, worker agents are represented by an interactive icon, which is shown in Fig. 5.7. Each worker agent contains a state chart with the states standing, walking, light work, and hard work. During walking, a worker moves along a defined path to the target location. The current location of a worker is mapped with the building zone areas and the heat and moisture emissions are allocated to the current zone. In general, modeling of worker agents allows either to determine the minimum number of workers or if a specific number of workers if sufficient for the operation of all machines. Furthermore, it allows to determine the heat and moisture emissions to each building zone. This aspect may be relevant for dry rooms which accept only a certain internal moisture intake.

5.2.5 Building Zones and Lighting

The core model allows defining building zones within the global shop floor coordinate system. Zones are characterized by area, temperature, lighting condition, and inside heat and moisture emissions. Heat emissions are caused by machines, workers, and lighting. For this reason, machines are assigned to building zones according to their position within the shop floor and processing units and workers are linked to zones based on their current location. The core model aggregates these emissions for each building zone and provides the internal loads to an interface for the connection of a building model. In return, a building model is expected to provide the inside temperature – and, depending on the specific building model – humidity of each zone. For this purpose, the definition of building zones in the process chain model has to match the zone definition in the external building model.

Each building zone has light sources which can be operated according to different schedules and control strategies. In the current model, the lights are turned on when a worker enters a zone and turned off if the last worker leaves a zone.¹² The control of lighting for each zone is realized by a state chart which assigns a state-based power

¹¹The model contains a pool of workers which are all able to operate any type of machine. It would also be possible to specify the skills and characteristics of each worker agent in more detail.

¹²It is assumed that all lights within a zone are turned on and off simultaneously.

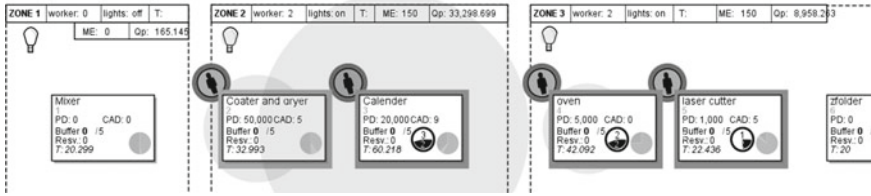


Fig. 5.8 Screenshot of core model level with zones, machines, workers, and processing units

demand for lighting. This power demand depends on the required lighting intensity, efficiency, zone area, and type of light source. The power demand equals the heat emission to the building zone. Figure 5.8 presents a partial screenshot of the process chain level showing multiple machines and workers within three building zones. The characterizing variables of each zone are presented at the top of each zone. The circles around machines indicate the current intensity of heat emissions.

5.2.6 Shift Schedules

Mostly, production facilities operate only during defined shift hours according to schedules. Considering these schedules in the simulation is important since machines and lights may be turned off and HVAC operation switched into energy saving mode during non-operation hours. Furthermore, the output of final products per time period strongly depends on the shift schedules. AnyLogic allows simulating dates and times based on a calendar. This enables defining schedules and deriving control signals for machines, TBS, and machine operators. In the implemented model, it is required to define a daily start time and a shift duration. However, it is also possible to define other schedules considering breaks and weekends.

5.2.7 Evaluation and Visualization

Simulation runs generate various data and values for defined performance indicators. The AnyLogic model allows browsing through all agents and model components to inspect specific variables, plots, and charts. Furthermore, on core model level, the evaluation cockpit provides an overview about the key indicators of each simulation run and allows to visualize and export time series of indicators. Plotting various system variables over a longer period of time enables the analysis of effects acting on a larger time scale such as weather impacts. Also it supports the observation of the required initiation time or so-called warm-up period of the simulation

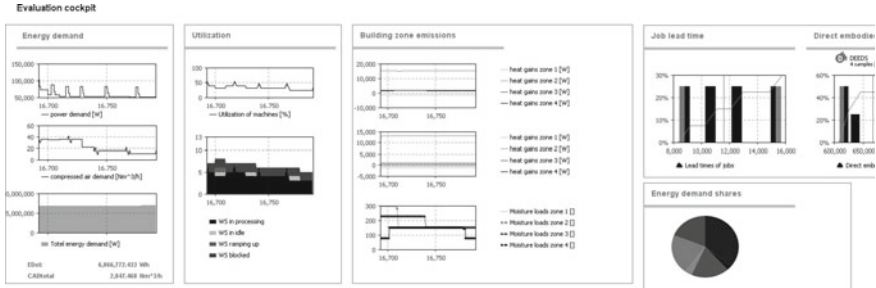


Fig. 5.9 Screenshot (for illustration only) of the visualization of performance indicators within the model cockpit (energy demands, process chain utilization, building zone characteristics, lead times per job, time shares of machine operation, etc.)

model.¹³ Examples of indicators in the cockpit are the electrical power demand of all machines, the compressed air demand of all machines and the resulting compressor power demand, the process chain utilization, as well as the aggregated heat and moisture emissions per building zone. In addition, a pie chart illustrates the shares of demanding systems on the overall energy demand. Furthermore, the cockpit contains histograms of the lead times and embodied energy of jobs as well as a display showing the current output of final products. Figure 5.9 shows a screenshot of the evaluation cockpit.

5.3 Compressed Air Generation System

The model for a compressed air generation system is implemented in the software Matlab Simulink based on the previously described concept model. The model consists of the three blocks tank, controller, and compressors. The tank block contains the functions for the calculation of the system pressure *cap*. Inputs to the block are the compressed air demand from the process chain, pressure losses due to leaks, and the air supply from the compressor. The output of the block is the system pressure which is an input to the controller block. The controller determines the air supply required to maintain a system pressure within the accepted interval. It generates signals for switching the compressor on and off. In the compressor block, Stateflow elements are used to model the compressor states off, on, and idle as well as to determine the air supply, the state-based power demand and the count of switching cycles. Figure 5.10 shows a screenshot of the implemented model.

¹³The warm-up period refers to the time from the start of a simulation run until the model represents the normal or balanced behavior of the real system. For example, in the simulation of a series production, the model is warmed-up if all machines are operating and buffers are filled. This period has to be considered in the result evaluation.

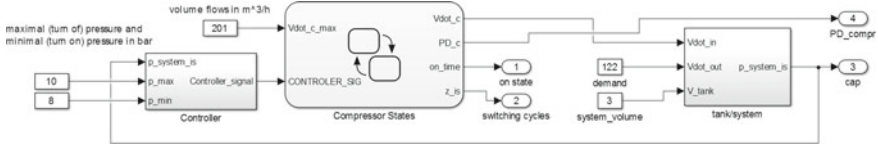
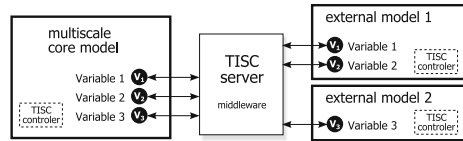


Fig. 5.10 Screenshot of compressed air generation system model in Simulink

Fig. 5.11 Exemplary structure for model coupling via TISC middleware



5.4 Coupling and Synchronization of Models

The core model provides two different types of interfaces for the coupling of external models. First, static numeric models can be connected by a direct interface.¹⁴ The process chain model calls the external model and waits for the results of the external model in order to continue the simulation based on these results. For example, this interface can be used to utilize static process models which calculate resulting product characteristics based on empirical data samples. Such model can be called by a machine agent during the processing state.

Second, external simulation models can be coupled with the core model by using a middleware software. The TISC software suite is used which provides different packages for establishing co-simulations.¹⁵ The core element is the TISC Center which connects different client models. The TISC Center is responsible for the co-simulation configuration, synchronization, and data exchange between all models.¹⁶ Figure 5.11 illustrates the connection of external models to the core model using TISC. At model start-up, AnyLogic has to load the required Java classes and connect to the TISC server. Moreover, all exchanged variables have to be defined at model start up. In the AnyLogic model, a reoccurring event triggers the synchronization by sending values to the TISC server, waiting for a synchronization signal, and writing the received values to the corresponding variables in AnyLogic. Figure 5.12 shows a screenshot of the TISC server control window.

¹⁴This coupling approach was used in Schönemann et al. (2016).

¹⁵TISC is offered by TLK-Thermo GmbH. Information can be found on www.tlk-thermo.com.

¹⁶The TISC suite provides interfaces to various software tools such as Matlab, Modelica (Dymola), Ansys, and others. However, TISC did not provide a standardized interface for AnyLogic. For this reason, the TLK-Thermo GmbH developed a TISC-to-java interface which enables AnyLogic to communicate with TISC via defined Java classes. This support is gratefully acknowledged!

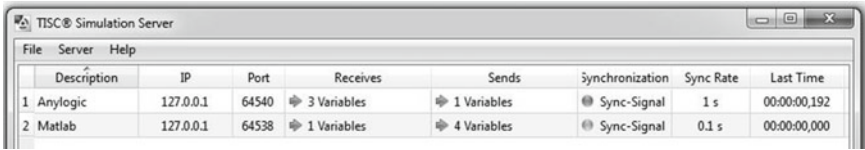


Fig. 5.12 Screenshot of the TISC server control window

5.5 Verification

The multiscale core model was verified during implementation. To demonstrate the used verification procedure, a generic job and model configurations were defined which support testing of all model functions and generating transparent results which are comparable with static calculations. The job defines the production of one batch of a product type. The product type requires three production steps, each of which is allocated to one machine and needs 60 min of processing time. The three machines are located in two different building zones both with an installed lighting power of 1500 W. Table 5.3 lists the characteristics of the machines.

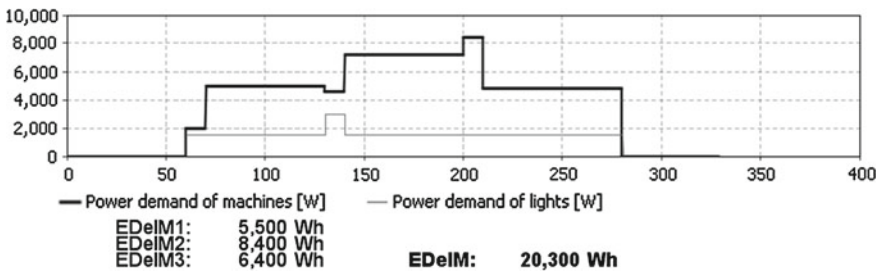
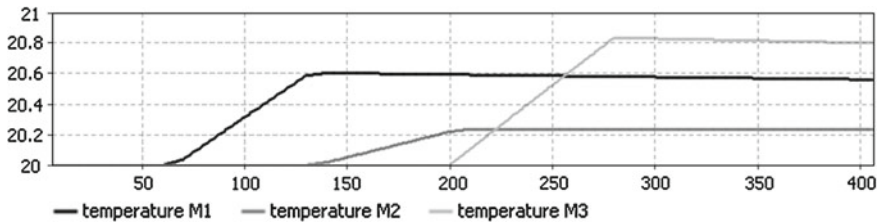
The energy demand per machine can be calculated based on the time shares for each operational state and the associated power demands. The results of static calculations state that the energy demands of the machines for the production of one batch are 5,500 kWh for machine 1, 8,400 Wh for machine 2, and 6,400 Wh for machine 3.¹⁷ Figure 5.13 presents a plot of the resulting power demand profile and the values of the resulting electrical energy demands. The figure also shows the power demand of lighting. During the operation of the first machine, the lighting in zone 1 is turned on. Afterwards, the lighting in zone 2 is turned on. Since machine 1 is in idle mode for 10 min after processing, there has to be a short period in which the lighting in both zones is turned on. This behavior can be seen in Fig. 5.13. Moreover, the power demand profile allows verifying the correct order of production steps. And since the lighting is turned on and off by workers, the power demand profile of lighting allows to conclude that workers enter and leave the zones correctly.

Similarly, in order to verify the determination of the machine temperature, the simulation results are compared to static calculations. The temperature of a machine can be calculated by using the equation $Q = m \cdot c \cdot \Delta T$. With the values for the mass and specific heat capacity, the power inputs, and an initial temperature of 20 °C for each machine, the resulting temperatures of the three machines can be calculated to be 20.61, 20.23, and 20.84 °C. During simulation, the model determines the increase of the temperatures up to the expected values, as shown in Fig. 5.14. After the power is switched off, the machines cool down due to the colder surrounding environment. The calculations of heat emissions of machines, workers, and lights as well as their allocation to building zones were also verified.

¹⁷For example, the calculation for machine 1 is $ED_{MACH_1} = 10/60 \text{ h} \cdot 2,000 \text{ W} + 1 \text{ h} \cdot 5,000 \text{ W} + 10/60 \text{ h} \cdot 1000 \text{ Wh} = 5,500 \text{ Wh}$.

Table 5.3 Machine configuration for model verification

	Machine 1	Machine 2	Machine 3
Zone	1	2	2
$t_{ramp-up}$ (min)	10	10	10
$t_{shutdown}$ (min)	10	10	10
PD_{rampup} (W)	2,000	3,600	4,800
PD_{idle} (W)	1,000	3,600	4,800
$PD_{processing}$ (W)	5,000	7,200	4,800
CAD_{idle} (m^3/h)	0	0	0
$CAD_{processing}$ (m^3/h)	122	10	10
Mass (kg)	5,000	11,000	3,000
Spec. heat capacity ($kJ/(kg \cdot K)$)	0.6	0.9	0.7

**Fig. 5.13** Simulated power demand of the three machines and lighting**Fig. 5.14** Temperatures of the three machines

The compressed air system model was verified with an approach similar to Thiede (2012), who used a calculation example shown by Bierbaum and Hütter (2004) to verify his compressed air module. They assumed a constant compressed air demand of $122 \text{ m}^3/h$, a system volume of 3 m^3 , as well as a compressor with a maximal volume flow of $201 \text{ m}^3/h$, 30 kW power demand during operation and 10 kW during idle. Furthermore, Thiede assumed an initial pressure of 8 bars and that the compressor stays in idle mode for 60 s before being switched off. He simulated a period of one hour and the results showed eight compressor cycles and a power demand of 19.4 kWh . This scenario was also used for the verification of the developed Simulink

Fig. 5.15 Plots of compressed air demand (top), system pressure (middle), and compressor power demand (bottom)

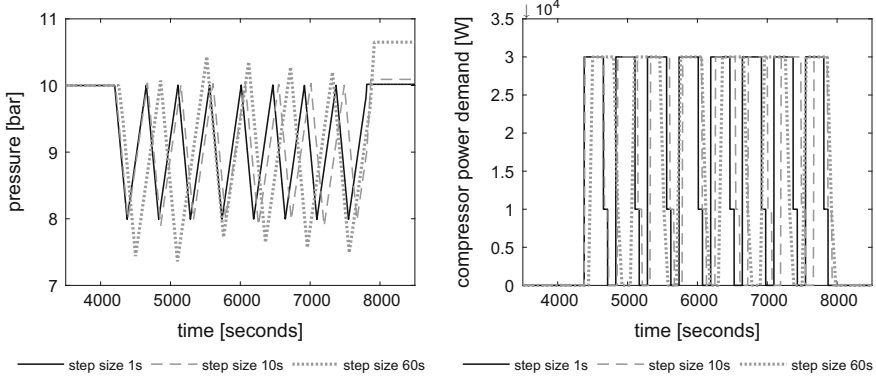
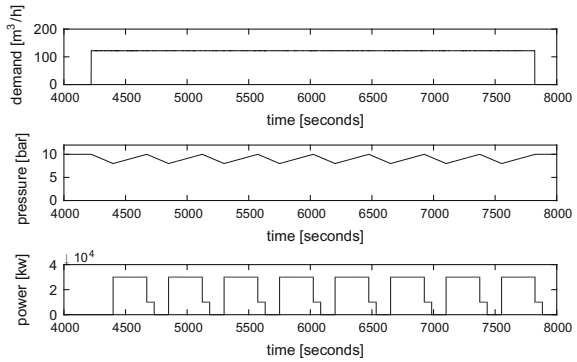


Fig. 5.16 Plots of pressure (left) and compressor power demand (right) for different simulation time step sizes (1, 10, 60s)

model. One of the machines was specified with a compressed air demand of 122 m³/h. The model generated the same results with eight cycles and 19.55 kWh.¹⁸ Figure 5.15 shows plots of the compressed air demand of the machine, the system pressure, and the resulting compressor power demand.

Another important aspect is the simulation step size of the compressed air system model as well as the step size for synchronization with the core model. In general, shorter time steps result in more accurate results but correspond to a longer execution time. The definition of the step size may have significant impact on the results. As an example for verification, three different step sizes of the compressed air system model have been tested. The results – shown in Fig. 5.16 – reveal that the step size effects the simulated system pressure and compressor power demand. If the simulation step size is larger, the lower and upper pressure limit is detected too late. This caused the compressor to start and stop the air supply too late which also effects

¹⁸The slight deviation is caused by the initial pressure of 10 bars and by the fact that the last compressor cycle was completely simulated and not stopped after exactly 3600s.

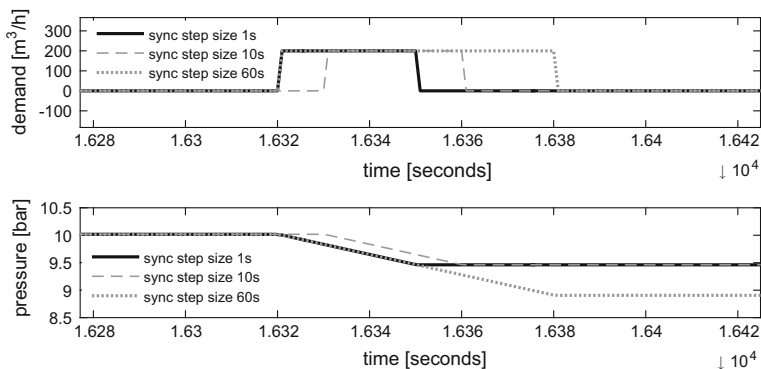


Fig. 5.17 Plots of demand (*top*) and pressure (*bottom*) for different synchronization time steps (1, 10, 60 s)

the resulting power demand of the compressor. The values for the power demand are 19.55, 19.583, and 19.50 kW for a model step size 1, 10, and 60 s respectively. Although these variations are rather low, they could add up to a larger error over the simulation period.¹⁹

In addition to the simulation step size, the synchronization step size is another relevant factor. This step size indicates how often variables are synchronized between the two models. At every synchronization step, the core model provides the current compressed air demand to the compressed air system model. That means that peaks in the compressed air demand which occur between two synchronization steps will not be communicated. To demonstrate this effect, a compressed air demand peak of 200 m³/h was generated for 30 s. Three simulation runs were conducted with synchronization step size of 1, 10, and 60 s. Figure 5.17 shows the results. It can be seen that the time of occurrence and the duration of the peak are detected wrongly for synchronization step sizes of 10 and 60 s.²⁰ This leads to the incorrect simulation of the system pressure. Since short synchronization time steps may result in longer simulation execution times, this experiment has shown that the step size must be defined according to the desired accuracy.

In summary, all relevant model functions were verified and found to provide accurate results. This includes also the correct creation of value stream maps for processing units and the functionality to use process models for the modification of product characteristics within processing units. Consequently, the model can be used as a core model for multiscale simulations. However, the exemplary results of the experiments with different model and synchronization step sizes show that it is of

¹⁹The simulation runs have been repeated for the production of 50 instead of one product unit. Shift schedules have been ignored. With a model time step of 1 s, the energy demand was 974.417 kWh. For 10 s it was 976.78 kWh and for 60 s it was 962.67 kWh.

²⁰In this example, the demand started exactly simultaneously to the synchronization step of 60 s. If it would have started just a bit later, it would have been ignored completely.

importance to set time steps which are suitable for the specific machine behavior, desired result accuracy, and acceptable simulation execution time. Otherwise, even a verified model may deliver inaccurate results.

References

- U. Bierbaum and J. Hütter. *Druckluft kompendium*. Hoppenstedt Bonnier Zeitschriften GmbH, Darmstadt, 2004. ISBN 3935772114.
- M. Schönemann, C. Schmidt, C. Herrmann, and S. Thiede. Multi-level Modeling and Simulation of Manufacturing Systems for Lightweight Automotive Components. *Procedia CIRP*, 41: 1049–1054, 2016. doi: [10.1016/j.procir.2015.12.063](https://doi.org/10.1016/j.procir.2015.12.063).
- S. Thiede. *Energy Efficiency in Manufacturing Systems*. Sustainable Production, Life Cycle Engineering and Management. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-25913-5. doi: [10.1007/978-3-642-25914-2](https://doi.org/10.1007/978-3-642-25914-2).