Hindawi Publishing Corporation
Journal of Computational Engineering
Volume 2015, Article ID 812835, 14 pages
http://dx.doi.org/10.1155/2015/812835

Research Article

Holistic Ontology-Based Assistance System for Efficient Process Model Parameter Identification

Burkhard Hensel,[1] Thomas Wagner,[1] André Gellrich,[1] Klaus Kabitzsch,[1] and Bernd Kauschinger[2]

[1]*Chair for Technical Information Systems, Dresden University of Technology, 01062 Dresden, Germany*
[2]*Institute of Machine Tools and Control Engineering, Dresden University of Technology, 01062 Dresden, Germany*

Correspondence should be addressed to Burkhard Hensel; burkhard.hensel@tu-dresden.de

Received 30 July 2014; Accepted 20 January 2015

Academic Editor: Franca Giannini

Accurate models of technical systems are the basis for many tasks like system analysis, predictions, or controller design. Usually, the values of several important parameters cannot be determined by theoretical analysis only; instead, process identification is necessary. For several applications, the efficiency of the identification procedure is very important, for example, for the creation of thermal models of machine tools, because of the large time constants and the expensive machine time. The goal of the authors is the support of this task as far as possible by software. This paper contributes to that goal twofold: on the one hand, it provides a collection of influences which have to be considered for supporting the identification procedure. On the other hand, concepts for computer-based support are presented—ontologies and automatic design methods based on evolutionary algorithms.

1. Introduction

Process models of technical systems are the basis for many tasks of engineering disciplines. While the internals of such models can be very different, from an abstract point of view the "interface" of the models is often similar. Figure 1 shows a process model as a "black box" as it is assumed in this paper. The inputs of the model are either manipulated variables, which can be adjusted manually or by a controller, or (measurable or unknown) disturbances. Also parameters could be regarded as inputs, especially if they are time-variant. However, often parameters are separated from inputs logically. Parameters are these quantities which describe the process itself and are changing relatively slowly or seldom compared to input and output signals, for example, coefficients of a differential equation. In the context of this paper, the outputs of the process are physical quantities which can be measured by sensors.

The aim of this paper is software support for the identification of unknown parameters of such models. Such software support can be useful for many applications, but it becomes especially clear at the example of thermal models of machine tools. Therefore, this example is explained in the following paragraphs.

When a machine tool operates, heat is generated by motors and points of friction like gears and bearings. This heat spreads through the overall machine and leads—via thermal expansion—to deformation of the machine tool [1]. This deformation is a reason for inaccuracy of products. Particularly, the accuracy of the "tool centre point" is important.

As avoiding this deformation (by constructional compensation approaches) is often too expensive or impossible, the best solution is to be aware of the deformation and correct the production error by adjusting the tool or workpiece position. Unfortunately, the deformation cannot usually be measured directly during production (online). Hence, there is much research in how to model and compute the deformation and correct the tool centre point. In that sense, the model is used as a *virtual sensor* [2].

If it was possible to simulate a sufficiently accurate model in parallel to the process, it would be possible to *produce goods more exactly although the machine is deformed*. Unfortunately, usually some parameters of the process are unknown and cannot be calculated a priori, for example, if they depend on

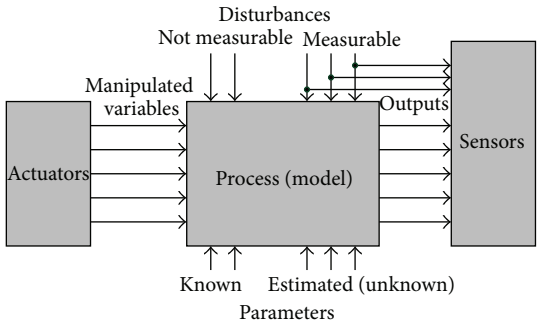

Figure 1: Process (model) with inputs, parameters, and outputs.

the installation (e.g., preload), if they are strongly scattered, or if they are time-variant. However, it is very important to know the values of all (significantly influencing) parameters, because tool centre point correction based on an inaccurate temperature model can even degrade production quality. The differences (residuals) between model outputs and real process outputs are used—often together with the input signals and measurable disturbances—to update the process model using standard nonlinear optimization methods as shown (simplified) in Figure 2 or—for linear models—standard process identification methods like least squares or maximum likelihood.

In practice, machine time is very expensive, but, for parameter identification, appropriate experiments must be carried out. The design of such experiments is difficult because of the large number of influences which must be considered. Currently, identification of parameters is very time consuming and expensive, because most steps are done manually, specific expert knowledge is necessary, there is no systematic adjustment methodology, only limited supporting tools are found, and there is no automation of the parameter adjustment. So it would be advantageous to use *tool support* (an *assistance system*) for tasks which can be formalized and automated. Tool support is desired for planning and setting up the identification process itself, for evaluating measured data, and for computing the parameters. This would have three possible advantages:

(i) time for experiments can be used more effectively;

(ii) time for experiments can be reduced;

(iii) experiment planning and data evaluation time can be reduced.

Many of the necessary tasks to solve the mentioned problem are standard for process identification. However, there are some requirements which are specific for the parameter identification of thermal models for machine tools, which make tool support especially valuable.

(1) Thermal processes are very slow. Time constants of minutes or hours are typical. Thus, parameter identification requires time-intensive experiments, especially for reaching the stationary state. This fact together with the problem of extremely expensive machine time (because in most cases nothing can be produced during the experiments) justifies outstanding efforts for reaching *effective experiments and tool support*. Additionally, this requires that *experiment preparation including installation of additional sensors is reduced to a minimum*. On the one hand, as much information as possible must be taken out of the available sensors; on the other hand, compatibility of necessary additional sensors must be checked before installation as a trial-and-error approach would be extremely expensive.

(2) Experiment and simulation results depend strongly on the *initial temperature distribution* at the beginning of the experiment. So, either the temperature distribution must be measured before each experiment or a time-consuming and complicated preparation of the machine (cooling down or heating up until the required temperature distribution is reached) is required.

(3) Due to the large number of model parameters in combination with the very limited experiment time (compared to the time constants) it is reasonable to invest exceptional efforts in the preceding *model analysis* to detect the critical parameters among the large parameter set and to prepare the experiments such that exactly these parameters can be identified optimally.

Because of these requirements, the parameter identification must be done up to now by skilled experts and is therefore even more expensive. To allow better product quality also for applications with lower cost limits, tool support could be a help as it assists in decision making.

The goal of the authors is to automate this parameter identification process as far as possible. This is content of work package B04 of the German Research Foundation Collaborative Research Centre Transregio 96 [3] as well as the Seventh Framework Programme project AMBI [4]. From a computer science point of view, two new steps are proposed in this paper:

(1) defining an extensible *ontology* which allows to *describe all necessary steps* for identification and for each of these steps the *possible solutions*;

(2) using an *evolutionary design algorithm* (automated design) to combine the appropriate steps for each specific machine to be modelled.

The goal is *not* to unify or restrict the simulation models or methods and steps of process identification. Instead, the goal is to *allow the integration of any models and methods by describing their interfaces and properties semantically and in a unified form*. As will be explained later, ontologies are predestined to describe such complex dependencies. Furthermore, ontologies are well suited to be used by a computer-based assistance system using formal, automated queries. This enables, for example, plausibility, compatibility, consistency, and completeness checks. In addition, by having these possibilities, an evolutionary optimization algorithm can be used to design the identification procedure. This

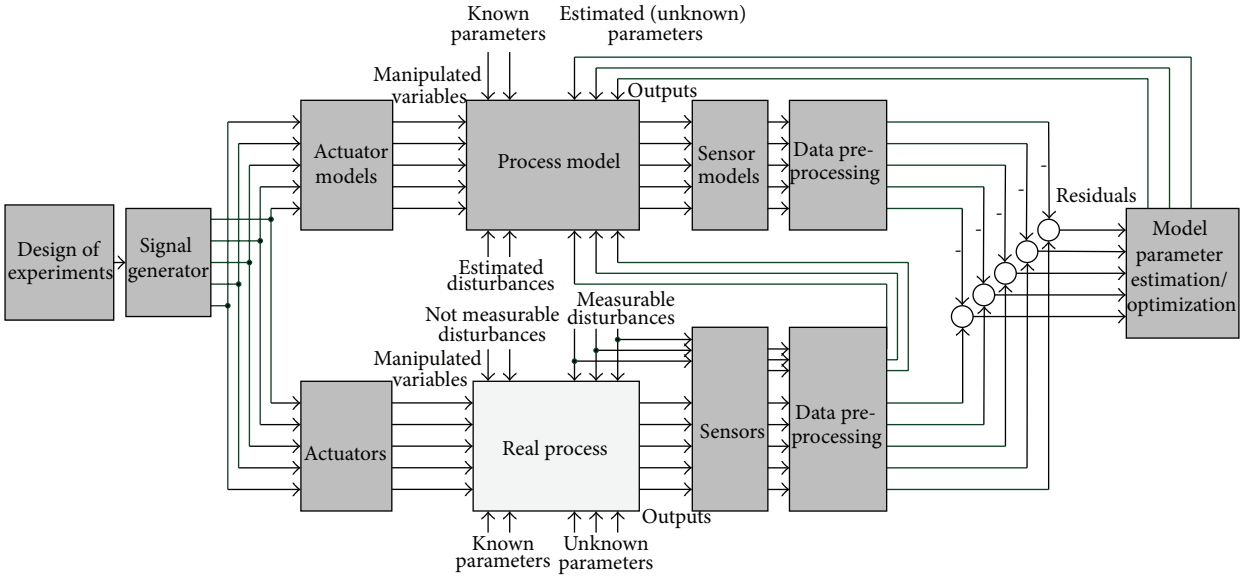

Figure 2: Process identification (simplified signal flows using only the residuals for model parameter update).

requires the development of a set of sufficiently generic, but semantically unambiguous interfaces for the models, methods, machines, and so forth.

It is clear that this goal can never be fulfilled completely as continuously new or updated methods are developed which require new definitions of properties and interfaces. Nevertheless, the authors expect that this approach leads to higher acceptance in practice than an intentionally "closed" approach limiting the allowed tools, methods, and models in the identification step chain.

Additionally, not all decisions can be automated and experience from building automation shows that making suggestions to the user is better accepted than deciding automatically.

To give some examples, subject of a study of the Dresden University of Technology was a machining centre for which a thermal model has been created [5]. Figure 3 shows a part of this model. The original model included about 540 parameters. However, it has been found that only about 15 of these parameters are significant (several power losses and heat transfer coefficients). Only these have been identified to improve the model in the most efficient way. In another example, the Dresden University of Technology analyzed a vacuum coating process [6, 7]. After designing experiments, the most influencing parameters have been identified using statistical methods, and after preprocessing measured data, dynamical models for different working points have been identified. The experiences of such (manual) process identification tasks delivered the practical knowledge about the necessary steps which shall in future be supported by a holistic software tool using the ontology described in this paper.

The whole solution must be developed and published stepwise in several articles. In this paper the focus lies on *a structured assembling of the steps* (the things one must think about) and the state of the art in the automatic design which is transferred to the process identification task in the mentioned projects.

This paper is structured as follows. First, the state of the art is analyzed in Section 2. Section 3 defines the steps of the identification procedure and discusses the questions to be answered at each of these steps. These things shall be described as an ontology. The rough structure of this ontology is presented in Section 4. Section 5 gives an overview on the automatic design which has been used in building automation successfully for years and will now be transferred to support the identification of machine tools. Finally, conclusions are drawn in Section 6.

2. Literature Review

2.1. Software Support for Process Identification. Although system identification (also known as process identification) is often needed, especially for controller design, to the authors' knowledge holistic tools for supporting that are rare.

Ljung gives a generic overview regarding how a tool for system identification should be composed [8, chapter 17.1], together with an overview on several example implementations.

MATLAB is a commercial interactive environment which is often used for process identification (as well as controller design), but without an additional toolbox the identification process is time consuming. The probably most well-known solution for this is the System Identification Toolbox [9], which automates the parameter estimation and provides a graphical user interface. However, it does not consider how the data is exchanged from the machine to MATLAB or how the experiments should be executed to be effective. Scilab is an open source alternative to MATLAB and toolboxes like CACSD can be used for process identification.

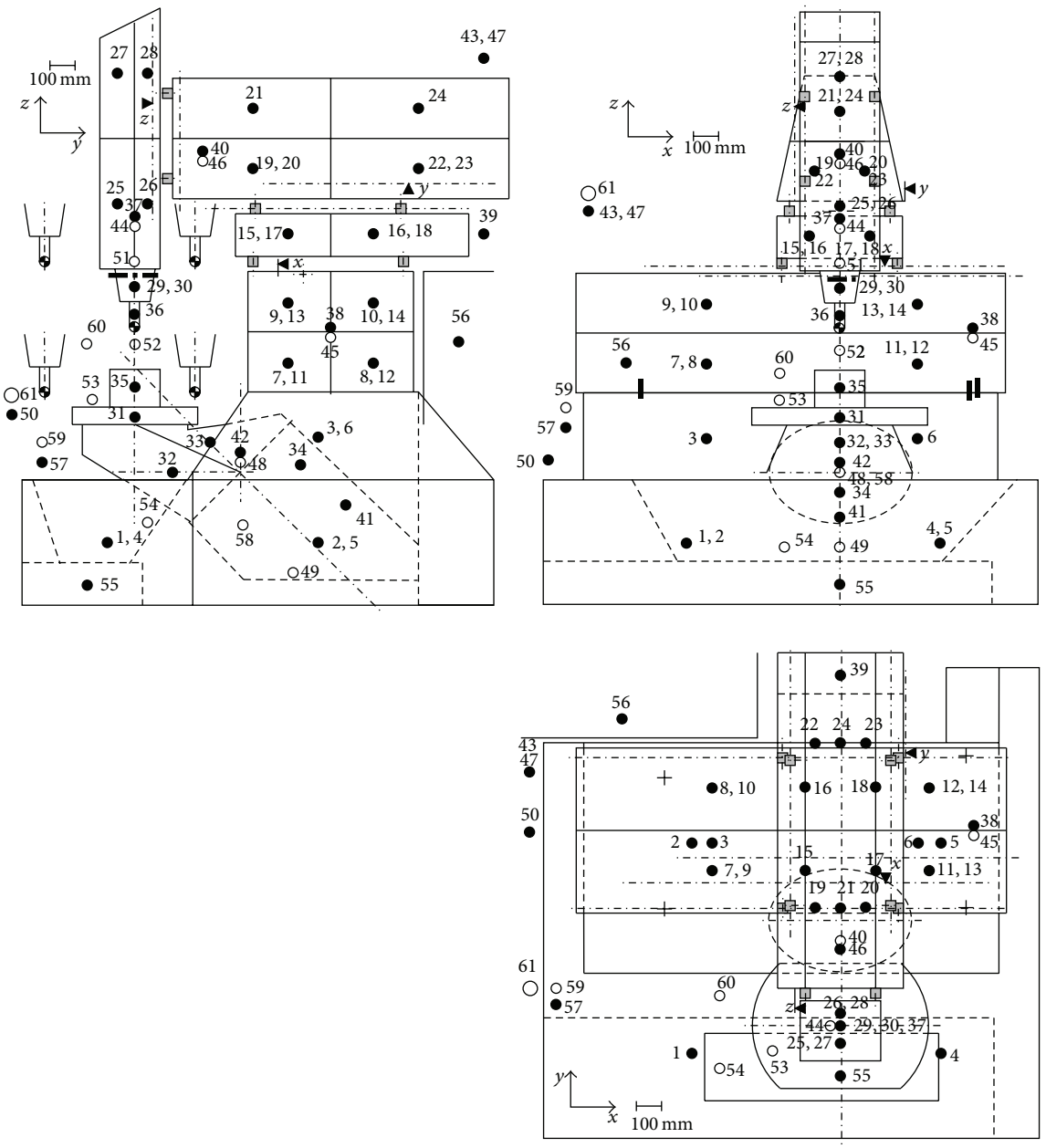

Figure 3: Nodes of the simulation model for a machining centre. See the source [5].

A framework based on MATLAB from Lund University [10] supports not only process identification, but also PI controller design (not needed here) and provides also a graphical user interface but supposes that measured data in form of step responses is already available in a format compatible with MATLAB. Also the work of Wills et al. [11] is based on MATLAB, provides a graphical user interface, and supports a variety of model types but does not help the user in getting the data from which the process shall be identified. Both solutions are not suited for FE models that are typical for thermal models of machine tools.

ControlAvH Tune [12] is software for system identification and simulation allowing different models and signal forms. It supports a variety of interfaces for communication with process equipment.

A variety of other process identification tools are provided by universities and enterprises. To the authors' knowledge, most of these tools are collections of process identification methods, but the way how to combine the methods is up to the user's skills. The new software to be developed in the described work shall not reimplement any process identification methods but shall describe and formalize existing methods and tools, allowing their simpler integration and reuse.

The originally academic, now commercially available software ADM supports the user in interpreting the quality of

identified models and decisions on which step should follow next [6]. Other than the tools mentioned before it supports the user in detecting interesting signals from a set of recorded signals using different correlation matrices. Further, it provides an adaptive database connection that simplifies the acquisition of data in a suitable format [13]. It also supports the design of experiments in a basic way.

A broader workflow is supported by [14]. The tool presented there gives "instructions for the identification operating procedures" and can produce input signals according to the user's requirements about frequency characteristics or power spectra. Also, several preprocessing steps are provided and the confidence interval of the identified parameters is displayed.

JMP is a tool for statistical data evaluation including design of experiments [15]. This is helpful for the tasks of this work but only a support for some of the necessary steps.

A generic workflow for process identification can be found in usual text books on that subject like [8]. The hints in that book for the workflow are (usually) limited to requirements from a mathematical point of view for good results of the estimation algorithms, like *persistence of excitation*.

2.2. Ontology Support for Process Identification. The inputs, attributes, parameters, outputs, and implementation details of the models and the workflow described in Section 3 have to be described semantically and formalized in a way suitable for later simulation and analysis. Special attention must be given to the ease of extensibility.

Traditionally, engineering knowledge is represented either in the heads of experts, in descriptive form like textbooks, or in purpose specific or domain specific formats such as SysML models. For sharing and (automatic) (re)use of models this imposes severe obstacles such as unspecified semantics and ambiguity of terms. For several years now, formal methods of knowledge capturing and processing, namely, ontologies, have gained importance. An ontology is a specification of a representational vocabulary for a shared domain of discourse including definitions of classes, relations, functions, and other objects [16].

In 2006, more than ten thousand ontologies existed in different domains [17]. The advantages of the usage of ontologies for knowledge representation unlike, for example, databases, are described, for example, in [18]. Ontologies are especially useful as a human readable form of knowledge representation supporting a common understanding of meta- and domain knowledge. For ontologies are extensible by simply importing other ontologies, it is possible to reuse and enrich upper level ontologies with knowledge about specific domains and even keep parts of the knowledge proprietary, for example, sensible information about production sites.

From a theoretical point of view, particularly the description logic (DL) is important for the content of this work as DL aims to "represent the knowledge of an application domain (the 'world') by first defining the relevant concepts of the domain (its terminology) and then using these concepts to specify properties of objects and individuals occurring in the domain (the world description)" [19]. DL as logical foundation was chosen because of the following reasons.

(i) Its core concepts like classes, instances, relations, and restrictions are intelligible for domain experts.

(ii) It offers automatic reasoning by providing logically entailed implicit conclusions.

(iii) The central goal—to represent domain knowledge and thus either formalizing it in the first place or transferring it from software program code to encapsulated knowledge bases—eases both the development of domain specific yet flexible software and the maintenance of the knowledge.

The OWL language family [20] has been designed for enabling knowledge sharing. Therefore, it supports collaboration between computer systems and human experts. In the domain of this paper, formal OWL ontologies will be used to explicitly define the meaning of terms and their relations and to formally capture and automatically reuse knowledge about systems, models, and other important concepts. OWL is endorsed by the W3C and was chosen as the framing technology for processing the knowledge base because of the following.

(i) It is sufficiently expressive. Several OWL dialects are available that offer a trade-off between expressiveness and reasoning tractability.

(ii) It is supported by the majority of ontology software tools. Furthermore, the so-called Manchester Syntax [21] is available to provide user-friendly syntax for the descriptions.

(iii) Knowledge can be organized in a modular fashion. This greatly helps concentrate on a specific domain because base knowledge can be simply imported. Large "upper world ontologies" freely available on the Internet provide basic knowledge from many fields such as physics, chemistry, and mathematics. Based on that modular approach, it is possible to reuse and enrich upper level ontologies with knowledge about specific domains and even keep parts of the knowledge proprietary.

In the current research, the use of ontologies is investigated for factory automation [22–24] as they have already been successfully applied to different domains, for example, medical engineering, biological science [25], or building automation [26, 27]. Recently, ontologies have been introduced in the field of modelling and simulation; in [28, 29] an overview is provided. From the most promising approaches—the PlaSMA ontology [30] and the DeMO ontology [31]—some concepts could be incorporated into the description language. Unfortunately, some promising ontological approaches as the SoPT ontology for simulation optimization [32], for example, effectively used in [33], are no longer available for integration. Therefore, the authors focused on well documented, readily available ontologies as a basis for the description language.

OntoCAPE [34, 35] is a semantically rich large-scale ontology in the domain of computer-aided process engineering. Although designed for that specific purpose, it is

composed of several subontologies ranging from the definition of a metamodel down to domain-specific description of knowledge in chemical engineering. Each of these modules can be reused easily. Modules of special interest for system identification are the *system* and *technical_system* modules that thoroughly define (technical) systems, subsystems, their relations, and different aspects as well as system properties and models. In addition, several supporting concepts, like units and physical dimensions, are reusable.

SWEET [36] is another large-scale ontology composed of several reusable modules covering areas like space, time, physical quantities, and science knowledge concepts like phenomena and events. Although it covers a large amount of concepts, it is semantically less rich than OntoCAPE because most of the concepts lack strict axiomatic definitions and informal descriptions.

3. The Identification Procedure

In this section, the steps of process identification are explained in detail. The goal of this is the analysis of each step on systematic, formalizable parts. The steps have been defined based on the state of the art (like [8]) and practical experiences of the authors.

Without loss of generality, the steps are explained using the example of thermal models of machine tools.

3.1. Creation of a Real-Time Simulation Model. The first requirement is having a simulation model which is able to simulate the process in sufficient detail and with sufficient performance for the given tasks. If the deformation of the machine tool is to be corrected continuously, the simulation must be so fast that it can be calculated in real time. Otherwise, the requirements are less strict.

If a model is available but its simulation is too slow (typical for finite element models), there are several possibilities to create a simplified model. Examples of models with reduced complexity are Wiener and Hammerstein models [9], consisting of a static nonlinear part and a linear dynamic part.

If even the model structure is not clear yet, there are several possibilities for creating such a structure; consider the following.

(1) Using general *design patterns* (templates) for building a model out of a set of simple component models the machine is composed of. This can be supported by an automatic design approach similar to that described in Section 5.

(2) *Theoretical modelling* from analyzing physics, CAD models, or complex simulation models. Sometimes it is necessary to reduce the model order afterwards.

(3) Basic process identification methods using *experiments* (e.g., principal component analysis (PCA), correlation matrix) are another possibility.

(4) Basic process identification methods using *simulations of complex models* (e.g., PCA, correlation matrix) to get simpler models are also another possibility.

For machine tools there exists the problem that the model *structure* can change over time, as this influences the heat flows. Possible reasons are the time-variable relative position and orientation of components and points of contact. Hence, used simulation environments must support time-varying model structures for such cases.

3.2. Semantic Description of the Model. The model must be semantically described in a machine readable form. This is necessary for automating or automatically supporting the following steps. The semantic description includes the following:

(i) model *inputs* (manipulated variables, disturbances) with respect to physical quantities, units, and semantic meaning;

(ii) model *parameters* (physical quantities, units, and semantic meaning);

(iii) model *outputs* (physical quantities, units, and semantic meaning);

(iv) *implementation details* like the realization method (DLL, MATLAB script, Ansys, etc.) and interfaces to control the simulation.

It has to be noted that each model describes a selection of aspects of a system to enable the understanding or command over that system [37]. Therefore, to enable use and reuse of a (simulation) model, it is necessary to describe the system or class of systems it is modelling. While the mere description of these structures can be reached with a variety of methodologies (e.g., SysML), rigorous semantics and reasoning can only be achieved with an approach based on formal logic, for example, on OWL ontologies.

3.3. Analysis of Which Parameters Must Be Identified. Here it is assumed that the simulation model is available and semantically described using the ontology as stated above.

The next important step is the analysis of which model/process parameters have to be identified. This can again be divided into several substeps as follows.

(1) *Which parameters are important/significant for the real production?* Examples could be heat capacities, conductance values, temperatures, viscosity, or prestressing force. A good result would be a list of parameters in decreasing order of relevance for the later setting of weights when load regimes are created. Parameters with large uncertainty but small effect on the temperature behaviour have only small relevance.

(2) *Which of the important parameters are not known a priori?* Only some of the parameters are significantly uncertain because of effects occurring during the construction, the running-in phase of the machine, or deterioration effects. It must be analyzed which parameters are available at which accuracy a priori. Support for this qualitative model analysis (relations, variance, and uncertainty) is needed. A closely related question is the following: "Which parameters can be

held constant during an experiment and how can this be achieved?"

(3) *Are the remaining (unknown) parameters orthogonal?* Redundant parameters can lead to infinite solutions. There should again be support for quantitative model analysis (sensitivity, orthogonality). Also it has to be decided how to deal with nonorthogonal parameters.

Further, a support tool for specification and configuration of the model structure and parameters (including visualization) has to be developed.

3.4. Selection of Needed Inputs and Outputs. The next step consists of answering the question regarding which inputs and outputs of the process are needed (must be excited) to identify the unknown model parameters.

Many simulation models are created to allow the simulation of a variety of machines or components and have inputs and outputs which are not needed to analyze their temperature distribution and deformation. It is necessary to identify such inputs and outputs. This simplifies the experiment and its duration, and it can be even necessary to avoid ambiguous results (and thus an underdetermined set of equations).

This step can be divided into several substeps, too, as follows.

(1) *Which model inputs and outputs are needed to find the missing parameters?* Which influences are the missing parameters dependent on (e.g., time, load, state, or history)? How are the missing parameters identifiable? Which inputs are needed to identify all necessary parameters (by variation of model parameters and computation of parameter effects)?

(2) Which ranges of each input have to be reached to identify the parameters in all relevant conditions and operating points? This is especially difficult for nonlinear relations between several parameters.

(3) Based on the needed inputs and outputs and the original model, it may be necessary that the model must be extended to simulate also the actuators and sensors. If this is the case, some of the steps, starting with the step mentioned in Section 3.2, will have to be repeated.

(4) Finally, there must be a visualization of the model (e.g., using 3D or graph based visualization methods). Support tools for model visualization are needed.

The result of this step is a *reduced version (or subset) of the simulation model, containing only the needed inputs and outputs*. This may be independent of the complexity of the simulated model.

To support the steps mentioned in Sections 3.3 and 3.4, existing strategies defined in state-of-the-art ontologies could be used. Parameters that reflect some property of the modelled system should be explicitly linked to these properties including their units of measurement and to phenomena known to exist in this system to support decisions about parameter relevance and model structure. OntoCAPE already provides concepts to achieve that.

3.5. Mapping of the Model Inputs and Outputs to Real Automation Devices. To automate the experiment itself, it is necessary to have knowledge about the actuators and sensors installed at the real machine. This is not necessarily the same as in the simulation model. Thus, semantics of and access methods to the sensors and actuators must be described in a machine-readable form.

After that, it has to be tested which of the necessary inputs and outputs for finding the unknown model parameters are already available at the machine. This can be done in a similar way like the "detailed design" described in Section 5.

If there are some needed model inputs or outputs which do not fit to any of the available sensors and actuators, it should be checked whether this can be solved by software. Preprocessing steps (like coordinate transformation) could make signals of model and real machine fit together. The automatic design method described in Section 5 is suited for being adapted to assist in such cases as its purpose is to make suggestions for semantically proper and formally compatible structures. In several cases, for not measurable, but important outputs virtual sensors can be set up.

If there are still some missing model inputs or outputs, it is necessary to install additional sensors or actuators at the machine. The questions are the following: Which? Where? How? This can especially be supported by the "detailed design" (Section 5), using the product database. It may happen that parameters of measurement setup and actuator drives have to be estimated, too, leading back to the step "semantic description of the model", described in Section 3.2. It is also important to analyze whether new disturbances are caused by the additional sensors.

Finally, if for some model inputs or outputs no fitting sensors or actuators could be found, then it is necessary to treat them as unknown disturbances.

3.6. Design of Experiments. Design of experiments is a technique used for efficiently determining statistical models (see [38] for a modern overview about the principles). Based on the results of the previous steps, especially Section 3.4., a particular plan for driving the actuators must be created to reach all needed operating points.

Suitable excitation signals (e.g., sequences of signal states or load regimes) have to be created to reliably identify the values of the significant parameters within the measured data.

A central requirement for the generated experiments from the view point of cost-effective parameter identification is a short measurement duration, if necessary adjustable for conditions (number and accuracy of identified parameters versus available time) as well as avoidance of the repetition of long-time experiments and cooling processes. Furthermore, plausibility tests (e.g., for consistency of requirements) and the automated reduction of design space (variety) by decision support are necessary. Finally, attention has to be paid for choosing suitable initial parameters (e.g., for the temperature distribution), thus inherently supporting the automatic generation of the best possible starting conditions.

The trajectories must be planned such that both the process inputs and the effect of the parameters on the measured signals are orthogonal.

Several other constraints have to be paid attention for, mainly due to technical reasons, not to the process identification. These include the following:

(i) the conditions of the machine and the measurement equipment (e.g., TCP must always be on the surface of a sphere around a given point for a constant measurement radius, minimum and maximum velocity, freedom (range) of movement, deterioration, safety, interactions of chemical reactions, or constant velocity for conversion from time to place);

(ii) the exactness of measurements and drives, depending on the regions of the trajectory;

(iii) technically realizable start and end position;

(iv) limits of the interface to the machine (e.g., only stepwise changes, quantization, sampling rate, or internal memory for transferred trajectory);

(v) different accuracy of the actuators and sensors (try to use especially exact devices), making—if necessary and time allows it—additional measurements for calibration of sensors and actuators;

(vi) that the identification should be done online while the machine continues producing or at least during short breaks while keeping the machine in working order.

An important aspect of this step is the following question: What happens in the case of faults or manual abort? The machine must be brought into a safe state, the experiment must perhaps be started again, and an analysis should be carried out on what possibly can be salvaged from the partial measurement (Only several parameters? Only for several working points?).

The result of the step—the generated experiment procedure—must be written down in a formal language which should be both machine and human readable. A new domain-specific language may be necessary with carefully defined syntax and semantics. This content must be translated or transformed afterwards for the concrete control and drive technology. Also, a visualization is needed for this step which ensures the comprehensibility of the generated trajectory (why has the trajectory been selected and how?).

3.7. Setting Up the Monitoring. When the experiment has been designed theoretically, the experiment must be prepared and executed.

This requires a lot of tasks regarding interfaces of the machine, such as describing interfaces to bus protocols and databases, as well as automatically configuring access to measurement data on the one hand (typically via field buses through, e.g., connect, configure, start, store, stop, and disconnect methods) and to controls or drives on the other hand, which is necessary for active experimentation. In addition, it is necessary to prepare the data storage in a semantically understandable form, perhaps by creating a new database (table) or file or perhaps by preparing data communication over a network, which implies a proper exception handling if the network fails. Furthermore, high-precision synchronization of movement (control) and measurement as well as of clocks and sampling times of controllers, actuators, and sensors is necessary. Lastly, a careful selection of a suitable sampling method (periodic, send-on-delta, or other event-based) and sampling rate with attention to jitter is needed.

If some of these tasks fail because of limited machine interfaces, it should be considered to go back to one of the former steps and try an experiment using other sensors or actuators. If this is not possible, the shortcomings must be carefully recorded and considered later at the evaluation of the suboptimal data.

Before the experiment begins, the measurement has to be set up and configured, the initial conditions (the initial state) have to be measured or—better, but more time-consuming—created automatically, and it has to be decided how accurately this has to be done as this has influence on the accuracy of the estimated parameters.

3.8. Data Preprocessing. Data preprocessing is more than the obligatory outlier elimination. Examples of data preprocessing are the following:

(i) activities for finding the important parts of longer data (data reduction): for example, sampling rate reduction by averaging several consecutive sampling times; excerpt of measurement from longer data; and elimination of erroneous data (e.g., short sticking of device or machine downtime/idle periods);

(ii) classic correction methods: for example, outlier correction; smoothing; estimation of missing data (sporadic dropouts), for example, by interpolation; averaging of several cycles; and plausibility tests of data;

(iii) conversion activities because of inevitable device limitations: for example, conversion of sampling times if simultaneous sampling was not possible;

(iv) semantic data conversions: for example, normalization; conversion of time to place; conversion of coordinate systems (one dimensional data with the help of placement knowledge to 3D data, or spherical coordinates into Cartesian coordinates); and conversion to values which can directly be compared to simulation or vice versa;

(v) building auxiliary models: for example, conversion of value sequence into static regression model; Fourier transformation; and finding relevant frequencies in a spectrum.

The goal is an automatic—or at least automatically supported—design of data preprocessing for each channel (signal, inputs, and outputs) based on requirements of the signal attributes. Data preprocessing may be necessary not only for the machine but also for the model, if the model outputs do not fit to the optimization procedure.

Understanding and reasonable preprocessing of monitoring data are greatly enhanced if the context that produced that data is made explicit. That includes linking that data to

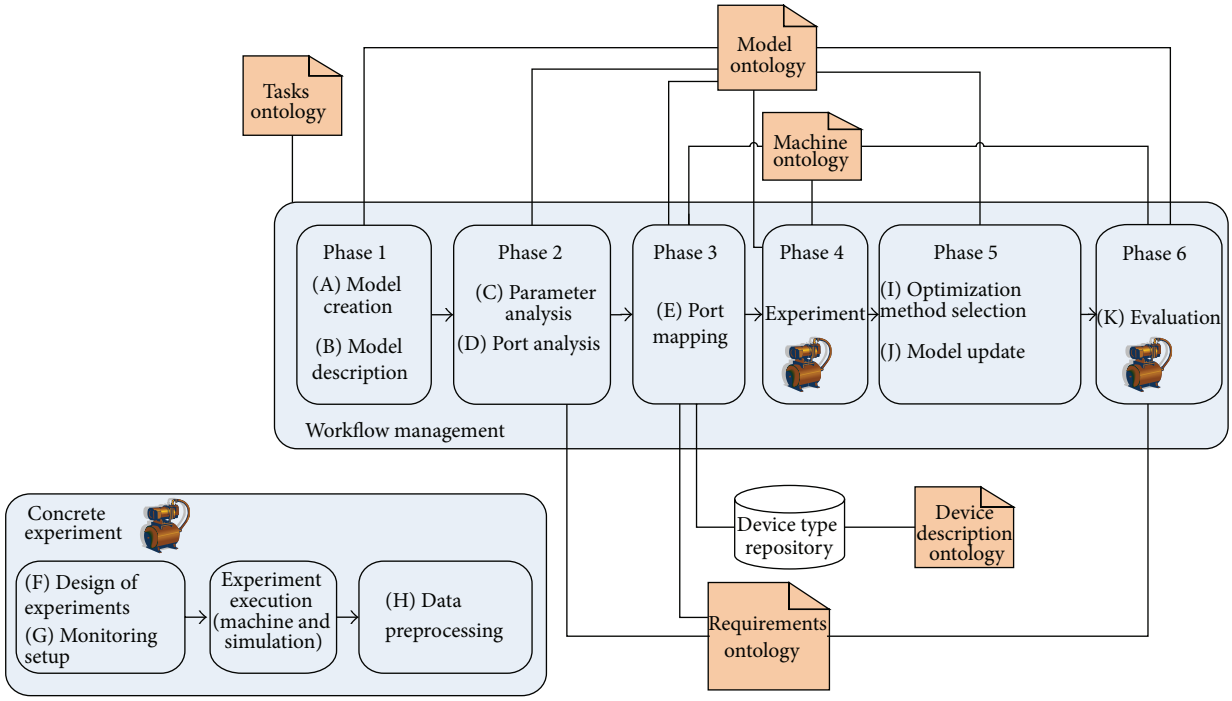

Figure 4: Overview on the use of ontologies for the automated process identification.

the respective (sub)system properties. Therefore, the monitoring set up and data preprocessing mentioned in Sections 3.7 and 3.8 should be supported by formalizing knowledge about the involved systems and contexts.

Finally, a suitable visualization method has to be developed, too.

3.9. Selection of a Suitable Optimization Method. After carrying out the experiment, the monitored data can be used to improve the parameters of the simulation model. This includes an analysis of which identification and optimization methods are available and suited for the specific application, what usually depends on the model type, for example, nonlinear FE model or thermal equivalent circuit diagram. It must be decided whether nonlinear optimization is necessary or linear optimization methods (e.g., recursive least squares or maximum likelihood) are sufficient. The optimization problem and tool must be adapted automatically to the specific task (model structure and parameters). A suitable implementation (like Lapack, MATLAB, Maple, and MathCAD) must be chosen, depending on costs, availability, features, visualization, and compatibility. In many cases it may be necessary to convert data for the used tool because of implementation specific interfaces. A support tool for model and result visualization is needed again.

3.10. Update of the Process Model. The next step is the system identification itself in the mathematical meaning, followed by the parameter update of the model.

Updating the controller-integrated behavioural models requires a description of and access to the model interface. In addition, the initial values, the transient behaviour, and the temperature distribution at the beginning of the measurement have to be accounted for. A visualization of the parameter changes is important for these plausibility checks which cannot be automated.

3.11. Evaluation/Verification. Finally, the model with the new parameters must be evaluated and verified. For this task, concepts for model quality monitoring have to be adapted. A critical task of this step is the generation of *another* trajectory than that which has been generated for the original experiment (section 3.6). While the trajectory must be different from the first one in order to find possible weaknesses of the identification results, it must fulfil the same requirements.

4. Ontology Overview

This section gives a very rough overview about the ontology set which is in development, also shown in Figure 4 together with steps (A) to (K) (which fit to Sections 3.1. to 3.11.).

The ontology can be divided into the parts about *models*, *machines*, *devices*, *requirements*, *tasks*, and *visualization*. While it is clear that models and machines have to be described by an ontology for the given tasks, this may not be clear for requirements, tasks, and visualization. The reason for using an ontology for these subjects lies within the diverse workflow of process identification, which is specific for each machine. If all steps of the procedure are described by an ontology, it is possible to support the user in deciding which steps are suitable for the given application and the current state of knowledge.

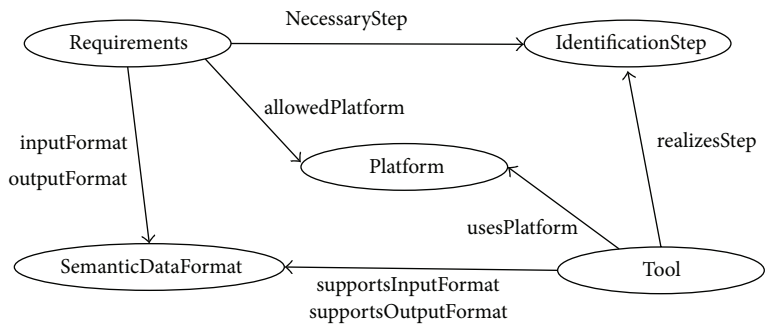

Figure 5: Sample ontology (terminological knowledge).

In addition to this structure, also another view on the ontology system is possible:

(i) ontologies for the description of the *overall workflow* (as a basis for an assistance environment);

(ii) ontologies for the description of the *methods* (model analysis, measuring, identification, validation, etc.) for describing tasks or software components semantically;

(iii) ontologies for the description of *components* (models, load templates, data transfer configurations, tool realizations, etc.) that are used by the methods or that are concrete realizations of the methods.

The concept of using an ontology for supporting the selection of appropriate tools in parameter identification is explained in the following using a simplified example.

An ontology can be divided into *terminological* knowledge (the specification of *classes* and their *properties*) and *assertional* knowledge (concrete *instances* of the classes, their relations, and axioms). Figure 5 shows the terminological knowledge of the sample ontology. The class *Requirements* collects all user-defined requirements for the concrete application, for example, some necessary identification steps (*necessaryStep*), available platforms (*allowedPlatform*, e.g., MATLAB), the data format of available a priori data that can be used (*inputFormat*, e.g., older measurements and model information), and the data formats that are desired as output (*outputFormat*). For each *Tool* it is stored which steps it can realize (*realizesStep*), which platform it bases on (*usesPlatform*), and which input and output formats it supports (*supportsInputFormat* and *supportsOutputFormat*).

It is advisable to use available ontologies as the basis for designing new ontologies. For example, the simple ontology shown in Figure 5 can be based on OntoCAPE (see Section 2.2). One advantage of integrating OntoCAPE is the semantic enrichment of the meaning of the classes and properties. For example, *usesPlatform* is a subproperty of the OntoCAPE property *hasRealizationAspect*, *realizesStep* is a subproperty of the OntoCAPE property *hasFunctionalAspect*, and the class *Tool* is a subclass of the OntoCAPE class *SoftwareSystem*. For the sake of readability, the OntoCAPE superclasses and superproperties are not shown in Figure 5. Anyhow, the true value of OntoCAPE cannot be expressed in this simple example, because in OntoCAPE a lot of knowledge that is needed by the assistance system is formalized in modules, including model description, physical quantities, SI units, and metaconcepts like n-ary relations and data structures.

OWL represents both kinds of knowledge (terminological and assertional knowledge) in a set of RDF [40] triples (subject, predicate, and object). Regarding assertional knowledge (instances), triples like (<ToolX> <realizesStep> <DesignOfExperiments>) and (<ToolY> <usesPlatform> <Matlab>) are stored in the ontology.

We assume in our example that the user already defined the desired steps to be executed, probably with the help of the assistance system. Let these steps be called <step1>, <step2>, and <step3>. Further we assume that the user stored the supported platforms and the format of available data in the assertional knowledge (with a user-friendly graphical user interface). The following task of the assistance system is to find a set of tools which realize the desired steps on one of the supported platforms, fitting to the specified input data format. That problem can be solved using only one SPARQL [41] query (simplified notation is shown in Algorithm 1).

The SPARQL query engine resolves this query by pattern matching, delivering all compatible tool chains for any platform which allow the complete tool chain. Each triple in the query is matched to one triple in the ontology by replacing the variables (the identifiers beginning with "?") by values found in the ontology. Due to the "OPTIONAL" and "ORDER BY" clauses the result set delivers first all solutions with an output format fitting to the user's requirement specification (if there are some) and after that alternative solutions which also support all steps but deliver another output format (as a fall-back solution). Each result is a tuple with the five replacements of the variables ?tool1, ?tool2, ?tool3, ?platform, and ?format4.

Although it is of course possible to solve this problem also without an ontology, other implementations will probably be more complex and less flexible. Additionally, with ontologies it is possible that parts of the ontology are developed by separate teams, with separate nomenclatures, as the "equivalent classes" feature of OWL [20] allows detecting solutions with changing neither the ontologies nor the SPARQL queries. Further, since ontologies originate from the semantic web,

```
SELECT ?tool1 ?tool2 ?tool3 ?platform ?format4
WHERE {
        ?tool1 <realizesStep><step1>.
        ?tool2 <realizesStep><step2>.
        ?tool3 <realizesStep><step3>.
        ?tool1 <usesPlatform> ?platform.
        ?tool2 <usesPlatform> ?platform.
        ?tool3 <usesPlatform> ?platform.
        ?requirements <allowPlatform> ?platform.
        ?requirements <inputFormat> ?format1.
        ?tool1 <supportsInputFormat> ?format1.
        ?tool1 <supportsOutputFormat> ?format2.
        ?tool2 <supportsInputFormat> ?format2.
        ?tool2 <supportsOutputFormat> ?format3.
        ?tool3 <supportsInputFormat> ?format3.
        ?tool3 <supportsOutputFormat> ?format4
        OPTIONAL {
                ?requirements <outputFormat> ?formatR.
                FILTER sameTerm(?format4, ?formatR)
        }
}
ORDER BY (!BOUND(?formatR))
```

Algorithm 1

the ontologies can be located on different computers and their integration is simple.

As ontologies can store information which does not fit to a predefined set of classes and properties, it is even possible to generate parts of the assistance software including its user interface dynamically from that knowledge base [42].

5. Automatic Design Method

The steps described in Section 3 can be supported by an automatic design method which has been used successfully in room automation for years [27, 39, 43, 44]. Therefore, this concept is explained in this section.

When designing a building automation system (sensors, actuators, controllers, control panels, and their connections), the first step is the formalization of requirements. For that, a requirements tool has been developed which automatically detects the required BAS components based on the functional needs of the building owner. Translated to the contents of this paper, the requirements can be transformed to a collection of possible process identification steps by formalizing the concepts, methods, and workflow items stated in Section 3.

In the next step, the selected BAS components from the requirements are brought together to build one common *abstract design*. This is realized by combining *templates* of partial BAS systems. Abstract designs show only semantic device *types*, not specific realizations (no specific vendor, no specific network technology, etc.). This method can be used in the process identification domain in various ways:

(1) to combine needs of several necessary identification steps in the optimal order;

(2) to extend an existing system (e.g., machine or building) by software preprocessing steps or even additional equipment for fitting to the model.

The last step is the generation of a *detailed design*. A detailed design is a realization of an abstract design with concrete devices from real vendors. Usually, there are many detailed designs which solve the problem according to the abstract design. Thus, the algorithm produces several proposals and the end user can select one of them manually. The selection is supported by showing several performance indices of each proposal, for example, price, network load, vendor count, used network types, and so forth. Because of the huge amount of BAS devices existing at the market, it is not recommendable to check each combination of devices for being a suitable solution. Instead, an *evolutionary optimization algorithm* is used.

The workflow is shown in Figure 6. Also in the BAS domain these algorithms are supported by a large ontology [27].

The detailed design algorithm can be transferred to the process identification problem in different ways as follows.

(1) Several implementations (tools) of the necessary steps can be checked for compatibility (Figure 6, right lower corner).

(2) Suitable sensors and actuators can be selected from a database of available devices if it is necessary to add new sensors and actuators to the machine according to Section 3.5.

(3) The mapping between the model inputs and outputs and the machine inputs and outputs can be supported by this method.

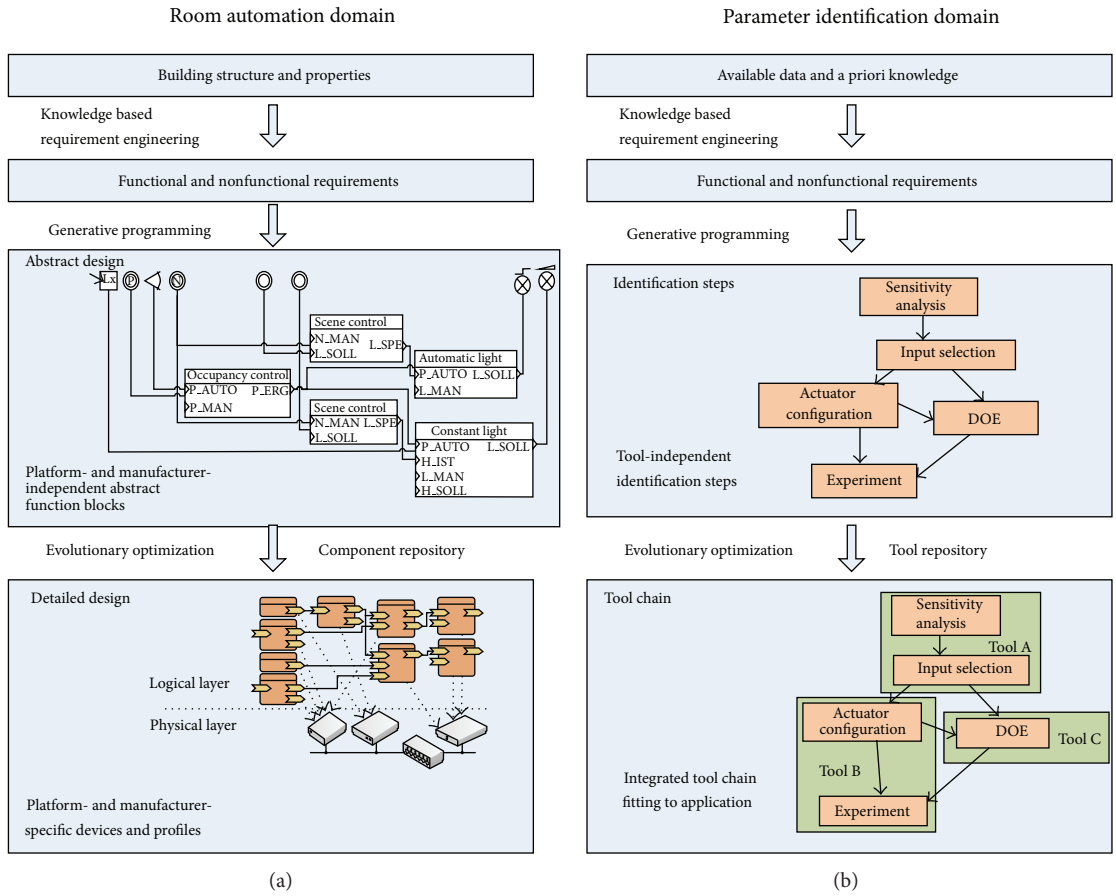

FIGURE 6: Workflow of the automatic design for designing building automation systems (a) [39] and its adaption to the assistance in parameter identification (b).

6. Conclusions

Parameter identification is a complicated task for which a lot of influences have to be cared about. Ontologies are a suitable concept for describing such complex dependencies. In addition, ontologies can be used as a backbone of an assistance system which makes the complexity more structured for the engineer. One such possible assistance is the automatic, evolutionary optimization of the identification procedure with queries to the ontologies. To the authors' knowledge, no such solution exists yet, although comparable algorithms have successfully been used for the design of building automation systems in the last years.

In this paper, the authors identified 11 steps and corresponding methods which are necessary for process identification and described an ontology based approach to support these steps by software. Such knowledge management is helpful not only for tool support but also for avoiding misunderstandings in communication between different teams or for exchanging data between them.

Further research will focus on developing the necessary ontologies in detail and use them to formalize the required knowledge. The approach will then be tested on selected use cases involving suitable machines provided by the project partners of the projects mentioned in Section 1. The focus will lie on the steps which can be supported by the described automatic design methods, because they have already been proven to be beneficial in the building automation domain.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work has been funded by the German Research Foundation in the Collaborative Research Centre Transregio 96 and by the European Commission in the Seventh Framework Programme project AMBI (Grant Agreement no. 324432).

References

[1] J. Mayr, J. Jedrzejewski, E. Uhlmann et al., "Thermal issues in machine tools," *CIRP Annals—Manufacturing Technology*, vol. 61, no. 2, pp. 771–791, 2012.

[2] A. Dementjev and H.-D. Ribbecke, "Improvement of prediction performance in virtual metrology," in *Proceedings of the 11th European AEC/APC Conference*, pp. 71–76, Dresden, Germany, April 2011.

[3] Sonderforschungsbereich Transregio 96, *Thermo-Energetic Design of Machine Tools*, 2014, http://141.30.37.177/SFBweb/?page_id=453.

[4] AMBI—advanced methods for building diagnosics and maintenance, 2014, http://www.ambi-project.eu/.

[5] K. Großmann and G. Jungnickel, "Prozessgerechte Bewertung des thermischen Verhaltens von Werkzeugmaschinen," Tech. Rep., Technische Universität Dresden, 2006.

[6] H. Kubin, M. Benesch, A. Dementjev et al., "ADM: a process identification tool for experts and technologists," in *Proceedings of the 35th Annual Conference of IEEE Industrial Electronics (IECON '09)*, pp. 1444–1449, Porto, Portugal, November 2009.

[7] H. Kubin, M. Benesch, A. Dementjev et al., "Identification of process models and controller design for vacuum coating processes with a long dead time using an identification tool with advisory support," in *Proceedings of the 14th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '09)*, Mallorca, Spain, September 2009.

[8] L. Ljung, *System Identification—Theory for the User*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1999.

[9] *System IdentificationToolbox—User's Guide*, The Mathworks, Natick, Mass, USA, 2014, http://www.mathworks.com/help/pdf_doc/ident/ident.pdf.

[10] M. Andersson, *A Matlab tool for rapid process identification and PID design [M.S. thesis]*, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, 2000.

[11] A. G. Wills, A. Mills, and B. Ninness, "A MATLAB software environment for system identification," in *Proceedings of the 15th IFAC Symposium on System Identification (SYSID '09)*, vol. 15, pp. 741–746, Saint-Malo, France, July 2009.

[12] L. García, M. J. López, and J. Lorenzo, "Hardware/software environment for process identification, robust controller design and hard real time implementation," in *Proceedings of the 5th WSEAS International Conference on Telecommunications and Informatics*, pp. 503–508, Istanbul, Turkey, May 2006.

[13] M. Benesch, H. Kubin, and K. Kabitzsch, "Adaptive database for universal data source access in industrial environments for ADM," in *Proceedings of the 12th Mechatronics Forum Biennial International Conference, Proceedings Book 2/2*, pp. 210–215, Zurich, Switzerland, June 2010.

[14] S. Tani, S. Takahashi, and T. Sekozawa, "Identification tool for chemical processes," in *Proceedings of the IEEE International Conference on Control Applications*, vol. 2, pp. 1561–1566, IEEE, Kohala Coast, Hawaii, USA, August 1999.

[15] SAS Institute, *JMP 9 Design of Experiments Guide*, SAS Institute, Cary, NC, USA, 2010.

[16] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, no. 2, pp. 199–220, 1993.

[17] D. Wang, B. Parsia, and J. Hendler, "A survey of the web ontology landscape," in *The Semantic Web—ISWC 2006: Proceedings of the 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5–9, 2006*, vol. 4273 of *Lecture Notes in Computer Science*, pp. 682–694, Springer, Berlin, Germany, 2006.

[18] I. Horrocks, "Ontologies and databases," in *Proceedings of the Semantic Days Conference*, Stavanger, Norway, April 2008.

[19] F. Baader, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The Description Logic Handbook: Theory, Implementation, and Applications*, Cambridge University Press, 2003.

[20] W3C OWL Working Group, *OWL 2 Web Ontology Language: Document Overview*, 2nd edition, 2012, W3C recommendation, 2012, http://www.w3.org/TR/2012/REC-owl2-overview-20121211/.

[21] M. Horridge, N. Drummond, J. Goodwin, A. Rector, R. Stevens, and H. H. Wang, "The manchester OWL syntax," in *Proceedings of the Workshop on OWL: Experiences and Directions (OWLED '06)*, vol. 216, November 2006.

[22] C. Popescu and J. L. Martínez Lastra, "On ontology mapping in factory automation domain," in *Proceedings of the 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '07)*, pp. 288–292, Patras, Greece, September 2007.

[23] J. L. M. Lastra and M. Delamer, "Semantic web services in factory automation: fundamental insights and research roadmap," *IEEE Transactions on Industrial Informatics*, vol. 2, no. 1, pp. 1–11, 2006.

[24] A. Gellrich, D. Lunkwitz, A. Dennert, and K. Kabitzsch, "Rule-driven manufacturing control based on ontologies," in *Proceedings of the 17th International Conference on Emerging Technologies and Factory Autmation (ETFA '12)*, Kraków, Poland, September 2012.

[25] M. Ashburner, C. A. Ball, J. A. Blake et al., "Gene ontology: tool for the unification of biology," *Nature Genetics*, vol. 25, no. 1, pp. 25–29, 2000.

[26] J. Ploennigs, B. Hensel, H. Dibowski, and K. Kabitzsch, "BASont—a modular, adaptive building automation system ontology," in *Proceedings of the 38th Annual Conference on IEEE Industrial Electronics Society (IECON '12)*, pp. 4827–4833, Montréal, Canada, October 2012.

[27] H. Dibowski and K. Kabitzsch, "Ontology-based device descriptions and device repository for building automation devices," *EURASIP Journal on Embedded Systems*, vol. 2011, Article ID 623461, 2011.

[28] P. A. Fishwick and J. A. Miller, "Ontologies for modeling and simulation: issues and approaches," in *Proceedings of the Winter Simulation Conference*, vol. 1, pp. 251–256, IEEE, Savannah, Ga, USA, December 2004.

[29] J. A. Miller, G. T. Baramidze, A. P. Sheth, and P. A. Fishwick, "Investigating ontologies for simulation modeling," in *Proceedings of the 37th Annual Simulation Symposium*, pp. 55–63, Arlington, Va, USA, April 2004.

[30] T. Warden, R. Porzel, J. D. Gehrke, O. Herzog, H. Langer, and R. Malaka, "Towards ontology-based multiagent simulations: the plasma approach," in *Proceedings of the 24th European Conference on Modelling and Simulation (ECMS '10)*, pp. 50–56, Kuala Lumpur, Malaysia, June 2010.

[31] A. Miller, G. T. Baramidze, A. P. Sheth, G. A. Silver, and P. A. Silver, "Ontologies for modeling and simulation: an initial framework," 2004, http://www.cs.uga.edu/~jam/jsim/DeMO/paper/journal/final4/demojournal_B_1_7.pdf.

[32] J. Han, J. A. Miller, and G. A. Silver, "SoPT: ontology for simulation optimization for scientific experiments," in *Proceedings of the Winter Simulation Conference (WSC '11)*, pp. 2909–2920, Phoenix, Ariz, USA, December 2011.

[33] J. A. Miller, M. E. Cotterell, and S. J. Buckley, "Supporting a modeling continuum in scalation: from predictive analytics to simulation modeling," in *Proceedings of the Winter Simulation Conference*, pp. 1191–1202, Washington, DC, USA, December 2013.

[34] J. Morbach, A. Yang, and W. Marquardt, "OntoCAPE—a large-scale ontology for chemical process engineering," *Engineering*

Applications of Artificial Intelligence, vol. 20, no. 2, pp. 147–161, 2007.

[35] J. Morbach, A. Wiesner, and W. Marquardt, "OntoCAPE—a (re)usable ontology for computer-aided process engineering," *Computers & Chemical Engineering*, vol. 33, no. 10, pp. 1546–1556, 2009.

[36] R. G. Raskin and M. J. Pan, "Knowledge representation in the semantic web for Earth and environmental terminology (SWEET)," *Computers & Geosciences*, vol. 31, no. 9, pp. 1119–1125, 2005.

[37] K. Wüsteneck, "Zur philosophischen Verallgemeinerung und Bestimmung des Modellbegriffs," *Deutsche Zeitschrift für Philosophie*, vol. 11, no. 12, pp. 1504–1523, 1963.

[38] G. E. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters*, Wiley, New York, NY, USA, 2005.

[39] H. Dibowski, J. Ploennigs, and K. Kabitzsch, "Automated design of building automation systems," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3606–3613, 2010.

[40] D. Brickley and R. V. Guha, "RDF Schema 1.1," W3C Recommendation, 2014, http://www.w3.org/TR/2014/REC-rdfschema-20140225/.

[41] The W3C SPARQL Working Group, "SPARQL 1.1 Overview," W3C Recommendation, March 2013, http://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

[42] H. Dibowski and K. Kabitzsch, "Generic specification toolchain for ontology-based device descriptions," in *Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '10)*, pp. 1–4, Bilbao, Spain, September 2010.

[43] H. Dibowski, *Semantischer gerätebeschreibung ansatz für einen automatisierten entwurf von raumautomation systemen [Ph.D. dissertation]*, Dresden University of Technology, 2013.

[44] A. C. Özlük, *Design space exploration for building automation systems [Ph.D. dissertation]*, Dresden University of Technology, 2013.