On the Architecture and Models for Intelligent Manufacturing Control

Christian Brecher, Kamil Fayzullin, Frank Possel-Dölken, Tilman Buchner

Abstract — **The increasing automation is an evident trend in manufacturing industry. The individualization of the products requires an increasing flexibility of automated manufacturing systems. Thus, concepts like Flexible Manufacturing Systems (FMS) gain importance in different manufacturing sectors. More complex usage scenarios and increasing complexity of the control tasks require higher intelligence of the manufacturing control solutions. Such intelligent manufacturing control must be able not only to execute a predefined control logic, but also be able to predict the development of the situation, schedule, plan and optimize the processes in FMS and give hints and recommendations to the operator concerning the optimization of usage scenarios. The development of such systems is impossible without a comprehensive modeling layer describing various system architectures and manufacturing processes. The following paper analyses the requirements on the modeling of intelligent manufacturing control systems and proposes an architecture for future control solutions.**

Index Terms — **Flexible Manufacturing Systems (FMS), manufacturing control, modeling, order processing, optimization, scheduling**

I. Manufacturing Control and Order Processing in Flexible Manufacturing Systems

The cost pressure in the manufacturing industry requires an increasing automation of manufacturing processes particularly in the area of small and medium-sized batch production. Relatively compact automated Flexible Manufacturing Systems (FMS) by now are technically mature enough for the industrial application and have proven their competitiveness in different manufacturing sectors [1]. The use of FMS is especially attractive with respect to small to medium lot size production of relatively large and frequently changing part spectrums. Technically, FMS consist of multiple manufacturing centers, linked by an automated, undirected material transport and loading / unloading system. The typical application domains of FMS are metal cutting, such as milling, turning and sawing, as well as sheet metal processing [2].

The manufacturing technology is one of the major factors influencing the layout of FMS and their internal logistics. Differences concern not only the arrangement of the manufacturing stations, but also the methods and means of material handling and transport. For instance, parts must be clamped with fixtures mounted on manufacturing pallets for the milling process. This determines the classical layout of a FMS for milling technology, which comprises an automated pallet storage system, equipped with a guided transport device or a stacker crane. Pallet loading stations as well as NC-controlled machining centers are arranged along the pallet storage system and can be loaded and unloaded automatically. Pallet and fixture configurations in FMS for milling technology are static, i.e. pallets and fixtures possess defined places.

Technology requirements of sheet metal processing differ considerably, thus the design of the corresponding FMS differs as well. For instance, no fixtures are needed for the sheet metal processing (e.g. for laser beam cutting) and since the part geometry varies considerably, storage places on pallets are assigned dynamically to single parts.

In order to develop unified approaches for FMS control a comprehensive model description must be found which will be able to reflect such differing FMS use cases concerning the structure of the manufacturing system, relations between components and varying manufacturing processes and technologies.

The main task of any manufacturing system is the execution of a defined workflow, which describes the sequence of operations to be performed in order to transform the raw material into finished parts. The specific feature of FMS is that such a workflow is not necessarily determined completely by the physical layout of the manufacturing system (as is the case for dedicated transfer lines for instance), but is only partially constrained by its structure. Workflows in FMS may be flexible as well and include decision alternatives, so that custom decisions, e.g. selection of alternative resources, routes, etc., can be made on the fly during workflow execution. Another characteristic property of the FMS is their "multitasking ability", i.e. the possibility of execution of multiple workflows

Manuscript received July 22, 2007. The work has been supported by the German Research Foundation (DFG).

Prof. Dr.-Ing. Christian Brecher is head of the Chair of Machine Tools at the Laboratory for Machine Tools and Production Engineering (WZL), RWTH Aachen University, 52056 Aachen, Germany, as well as head of the Department for Production Machines at the Fraunhofer Institute for Production Technology IPT, Aachen, Germany (e-mail: c.brecher@wzl.rwth-aachen.de).

Dipl. Wirt.-Ing. (RUS) Kamil Fayzullin, M.O.R. is with the Laboratory for Machine Tools and Production Engineering (WZL), RWTH Aachen University, 52056 Aachen, Germany (phone: +49 241 80 28230, fax: +49 241 80 6 28230, e-mail: k.fayzullin@wzl.rwth-aachen.de).

Dr.-Ing. Frank Possel-Dölken is with the Laboratory for Machine Tools and Production Engineering (WZL), RWTH Aachen University, 52056 Aachen, Germany (e-mail: f.possel-doelken@wzl.rwth-aachen.de).

Dipl.-Ing. Tilman Buchner is with the Laboratory for Machine Tools and Production Engineering (WZL), RWTH Aachen University, 52056 Aachen, Germany (e-mail: t.buchner@wzl.rwth-aachen.de).

(or copies of one workflow) in parallel. Both the diversity of technical cases and operation flexibility explain the particularly demanding requirements on manufacturing control systems for FMS. In order to achieve an optimized utilization of the FMS, advanced control system architectures are required, which will be able not only to trigger the execution of control commands, but also to make intelligent control decisions by means of capacity simulation and prediction of the situation development in FMS.

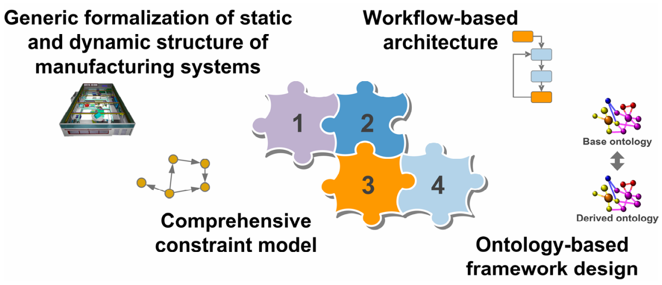
A wide variety of custom usage scenarios for FMS require comprehensive configurability of the control and scheduling software. Lots of custom rules and constraints may exist concerning the clamping operations, the material transport and the storage (ways, delays etc.), the definition of work plans, the structure of job orders (quantity, mixed part orders etc.), the lot-sizing, the shift calendars or the scrap parts processing. Additional restrictions may be stipulated by the shop floor organization outside the FMS. It must be possible to flexibly configure such user-specific constraints in the scheduling, simulation and control process, as well as to express custom optimization objectives.

Since the whole diversity of practical use cases cannot be addressed during design-time of the FMS control, configurable architectures for such systems must be developed which will allow a flexible adaptation of the manufacturing control system to custom constraints. Such customization of the manufacturing control system must be easy and effortless to accomplish, thus enabling fast changes and rapid ramp-up of the production after some modifications have been made. The systematic approach necessitates the generalization of use-cases, which makes it possible to separate the case-specific and generic (case-independent) functions. This assures that common features will be implemented only once and can be just "referenced" when realizing special cases. The requirements on the modeling of manufacturing systems were not considered sufficiently systematically in the past. Therefore, most PC-based solutions for manufacturing control systems possess case-specific design aimed to support some certain manufacturing system configuration. As the simplest approach certain rules and algorithms are hard-coded in such architectures, with the drawback that it becomes impossible to alter any data structures or control rules later without major changes in the control system's program structure.

II. Solution Approach for New Generation Manufacturing Control Systems

As a summary of the features described in the previous section a list of key requirements on manufacturing control architectures was derived. These key requirements are as follows:

- workflow-orientation, since the primary task of a control system is to execute defined workflows. A flexible and powerful mechanism must exist in order to enable the user to describe, manage and execute custom workflows.
- clear separation between use-case-specific and use-case-independent functionality
- adaptability to different manufacturing system structures and different manufacturing processes
- rapid and uniform interfacing with the manufacturing hardware controls (NC, MC, PLC or RC)
- scalability of the manufacturing system (applicability to manufacturing systems of any size)
- scheduling of operations and predictive optimization of control decisions based on capacity simulation
- flexibility while defining user-specific control, scheduling and optimization constraints, rules, algorithms and objectives.


Generic formalization of static and dynamic structure of manufacturing systems
Workflow-based architecture
1
2
3
4
Base ontology
Derived ontology
Comprehensive constraint model
Ontology-based framework design
Figure 1: Conceptual components of the of the generic next generation manufacturing control platform *cosmos 4*

The results from the analysis of these key requirements led to the development of a new architecture concept for manufacturing control systems. This architecture, developed at RWTH Aachen Universty's Laboratory for Machine Tools and Production Engineering WZL, is based on the following conceptual components (Fig. 1):

- workflow engine based on a generic FMS reference model
- highly-expressive object-oriented constraint model
- integrated ontology-oriented meta-modeling approach.

These concepts will be discussed in detail in the next sections.

A. *Workflow-based architecture for manufacturing control systems*

As mentioned before, the main purpose of any manufacturing system is the execution of a workflow, which consist of a sequence of operations (possibly with some alternatives available), transforming some physical object(s), e.g. raw part(s), from their initial state into a desired final state, e.g. finished part(s). The workflow idea is widely used to describe time-variant processes in different areas: the most common example is the modeling of business processes. Discrete manufacturing processes in automated manufacturing show similarities with business processes. As in the case of business processes any manufacturing workflow possesses clear start and end points and defines a sequence of actions for the execution.

Although the general idea is quite similar for both, the business process modeling and the modeling of manufacturing processes in automated manufacturing systems, there still exist significant differences concerning the structure of the corresponding workflows. Thus, reference models aimed to describe business processes (like BPEL [3], OMG Workflow Facility Specification [4], WFMC [5], etc.) and existing Workflow Management Systems (WfMS) are not directly applicable for the description and execution of manufacturing processes. The formalization of workflows in automated manufacturing systems requires the consideration of more details. In particular the following features are specific to manufacturing workflows: (i) manufacturing processes result in the modification of the internal state of any entity within the manufacturing system; these transformations must be described by the workflow, (ii) the execution of actions as elements of a manufacturing workflow demands the interaction with the controller and the actuator/sensor level and the invocation of device-specific interfaces, (iii) certain similar or completely identical series of actions (defined by the capabilities of manufacturing systems) replicate as elements of more complex workflows, (iv) the possibility of a parallel or sequence-dependent execution of multiple workflows increases the requirements on the synchronization of single operations as well as planning and optimization of the workflow execution. A generic formalization of structures of and processes in manufacturing systems is thus one of the most important basics for the workflow model for automated manufacturing systems.

1) Reference model to describe manufacturing systems and processes

A model is a formal representation of the elements and relations within some domain. Any formal model is built up by means of certain constructs and rules. The model for the language defining these constructs and rules is called the meta-model with respect to the corresponding model. A meta-model restricts and constitutes the application domain to be described and thus may possess a different grade of universality (or inversely: specialization for some certain domain). An example of a generic or widely applicable meta-model is the UML. A kind of tradeoff exists between the universality of the model and its level of detail, the common dilemma between scale and scope.

The reference model for automated manufacturing systems must primarily address two major domains: the structure of the manufacturing system (static) and the general process description (dynamic). In order to enable its universal usage, such a meta-model must provide means to describe common or fundamental system features and an extension mechanism to describe special cases. A generic formalization for structures of and processes in manufacturing systems was developed at the WZL based on their comprehensive analysis and classification. Processes within flexible manufacturing systems (e.g. manufacturing, assembly, handling, measuring as well as material movement and storage) all result in some kind of transformation of the properties of certain components of a manufacturing system. Such transformation may basically refer to the modification of the part geometry or its properties

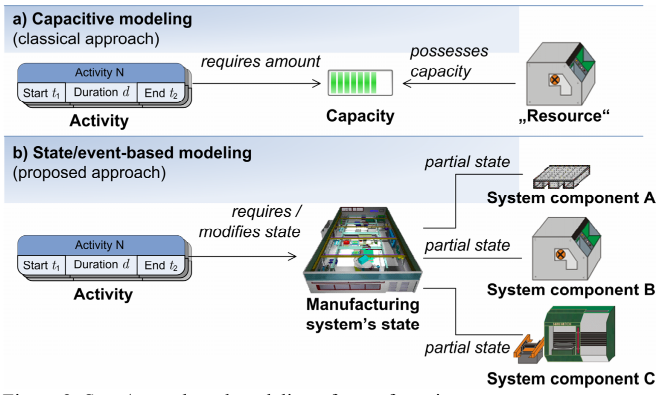


Figure 2: State/event-based modeling of manufacturing systems

(e.g. processes like forming, separating, assembly, coating or hardening) or some modification of the mutual arrangement or location of components of a manufacturing system (e.g. processes like transportation, placing/removing parts to/from stock, changing of component layout or orientation). A state of the manufacturing system results from the superposition of the states of its single components (see Fig. 2). The classical approach considering only the capacity of manufacturing "resources" is not sufficient.

The structural model for automated manufacturing systems must provide means for the description of its various components. Each custom manufacturing system possesses custom components differing by their kind, number, functions and properties. Such components can be machine tools, work places, machine tool elements (e.g. spindles), pallets, workpieces, tools, fixtures, transport devices etc. The meta-model must provide means for the description of properties, roles, layout and mutual arrangement of components within a manufacturing system. The meta-model developed at WZL solves this problem by introducing an object-oriented view to the architecture of a manufacturing system. Therefore it becomes possible to apply well-known fundamental concepts from the area of software systems engineering to the description of the physical structure of manufacturing systems. These are: object-orientation, classification of entities, generalization of classes as well as typed class and instance associations. The model of the manufacturing system thus can be defined by means of two layers (type and instance layers) as shown in Fig. 3. Physical dependencies between components (on the instance layer) are restricted by means of typed relations (relations defined on the type layer). The type layer thus serves as a template in order to define possible component types, their relevant states and possible relations. The instance layer describes actual individual components of a manufacturing system as well as their actual relations and states at certain moments of time.

Production processes are defined as a transformation of relations and properties. Some processes (e.g. assembly or separation of components) may require a "removal" or "creation" of new instances of system components.

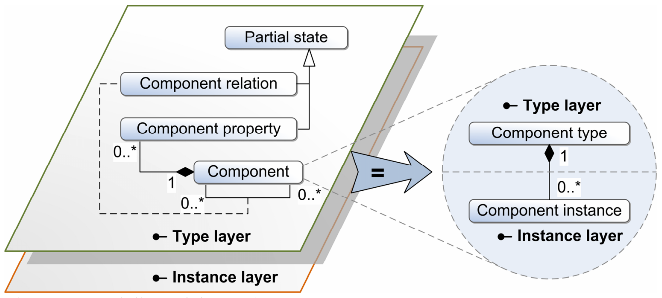


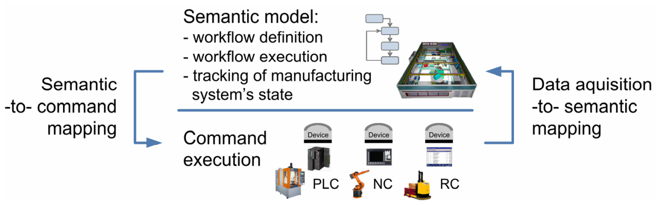
Figure 3: Modeling of the static structure of manufacturing systems

The classification of manufacturing processes (by means of a class layer) makes it possible to distinguish consistently between such fundamental concepts as process definition and execution.

2) Workflow reference model for automated manufacturing systems

The event-driven decision mechanism is the state-of-the-art concerning the architecture of manufacturing control solutions. Decisions are mostly made on demand right before their execution. Often only the actual system state (at the time of decision making) is taken into account. An execution of a manufacturing operation is triggered as soon as the actual system allows, e.g. after a confirmation for a finished previous operation has been received. Such event-driven processing is common practice to reduce the complexity of the control problem, since it helps to eliminate all unknown variables, e.g. the duration of a certain operation for instance, and makes it possible to use a rather simple rule-based control logic. The drawback of event-driven solutions is the lack of foresight, e.g. a limited planning horizon (normally limited to the next operation to execute). In contrast to an event-driven architecture, a predictive shop floor control requires extended information about the influence of the command execution on the modification of the manufacturing system's state. The planning instance needs information about the rules which describe if certain decisions and events may or may not take place. The corresponding temporal information concerning the single events is required as well (start/end times of certain actions, their durations etc.). Furthermore, the control logic of the manufacturing control software must be transparent to the planning instance, otherwise the "execution as planned" is not guaranteed. These fundamental observations apply without regard to the control architecture (central or distributed).

The reference model introduced in the previous section enables a semantic description of the manufacturing system structure and processes on the shop floor level. For the purpose of actual command execution the information must be forwarded from the shop floor level via cell level to the controller level. Modern device controls (NC, MC, PLC or RC) are themselves relatively complex systems and in most cases enable a comprehensive parameterization of the command execution, mostly in form of custom control programs (e.g. G-code or PLC-code).


Semantic model:
- workflow definition
- workflow execution
- tracking of manufacturing system's state
Semantic -to- command mapping
Data aquisition -to- semantic mapping
Command execution
Device
Device
Device
PLC
NC
RC
Figure 4: Data interfacing gap between the semantic and execution levels

Unfortunately, due to the decisive differences in the corresponding data models there is no way to exchange the semantic information available at the shop-floor level with the controller level. A model being able to describe both the shop floor and the controller level in an uniform manner does not exist as state-of-the-art. Therefore, the information about the transformation of the manufacturing system's state must be somehow mapped to the control programs to be executed (see Fig. 4). This can be done by means of the semantic interface model developed at WZL, thus providing the semantic link between the parameterization of the functions to be executed on the controller level and their impact on the manufacturing system state which is relevant for the shop floor level. Analogous to the classes in object-oriented concepts, integrating data and functions, the command execution functionality is encapsulated by components of a manufacturing system. The component model provides a possibility to define capability functions, taking other relevant components as parameters and mapping the transformation of their properties to calls on a standardized equipment interfaces (by means of special equipment drivers).

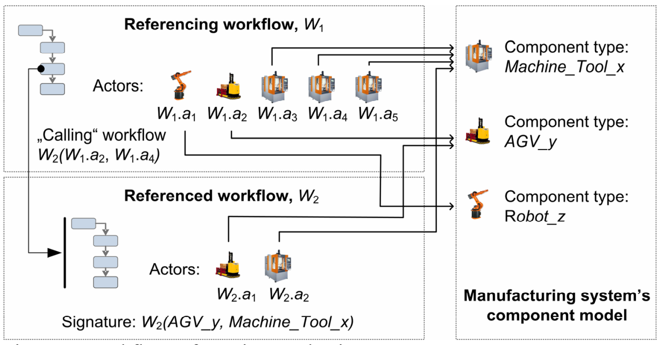
In automated manufacturing systems series of actions are often replicated as elements of more complex workflows. Thus, different definitions of workflow boundaries become possible. Any elementary sequence of operations, e.g. a transport job executed by a stacker crane or an automated guided vehicle (AGV), can also be considered as a workflow. In the latter case it would consist of the following operations: move to source cell, load pallet, move to destination cell, unload pallet. This workflow applies to pallets of any type a stacker crane is able to transport. Such transport jobs may be executed independently in the system (although they rarely make sense just as standalone actions) or as a part of a more complex manufacturing chain. There are many examples for such repeatable mini-workflows in manufacturing systems (aggregate operations consisting of multiple simple steps and as a rule requiring an interaction of multiple devices). Typically this refers to all logistical operations like material transport, handling and storage. These will later be called operation workflows.

Another kind of workflow is the product workflow. These workflows have the distinct aim of producing a certain amount of a certain product. Given a custom manufacturing system and a product to be manufactured on that system, it would be preferable from the user's point of view to be able to define a product workflow on the level of aggregated operations (e.g. workpiece setup – milling in first setting – transport –

workpiece setup change – transport – milling in second setting – transport to storage) without caring too much about the execution details of single manufacturing steps. The reason is obvious: this makes the workflow definitions more flexible avoiding a redundant definition of detailed sub-workflows. The idea is analogous to the code reuse as one of the key concepts in the area of software engineering. The reuse helps to save time and increase the reliability, since reusable components must be tested only once, and simplify the reengineering considerably in the case of changes. The solution with respect to the manufacturing workplans is to enable a workplan sub-referencing. In this case operation workflows can be used as integral parts of product workflows which will avoid the redundant definition of similar sub-processes. This subdivision into product and operation workflows is only conceptual, since it reflects the logical purpose of such workflows. In fact, depending on the individual manufacturing system, an arbitrary depth of the workflow referencing hierarchy may be possible. The atomic building blocks of any workflow (leaf nodes of the workflow reference graph) are the capability functions of the manufacturing equipment.

The idea of the developed interfacing mechanism is similar to the realization of function calls in function-oriented programming languages. A function provides a signature or an interface, defining its formal parameters. The call on a function specifies the association of the actual parameters, if any, with formal parameters of the function. Thus, applied to workflows, the concept is as follows: each workflow is seen as a black box defining operations on distinct abstract instances of typed system components (actors). In order to describe optional or alternative actions, a workflow contains a description of its decision logic and may define certain decision (state) variables to be referenced in its internal logical expressions. Furthermore, each workflow possesses an interface which exposes its signature to the "outer world". Such a signature includes formal parameters for actors and initial variables. "Calls" on such an interface, which is a referenced workflow, can be performed from any other (referencing) workflow. In this case a referencing workflow must provide actual parameter values for the referenced workflow, i.e. component instances for the actor assignment and initial decision variable values (see Fig. 5). Calls on referenced workflows may also be done conditionally depending on the decision variables of the referencing workflow. The developed interfacing mechanism makes it possible to integrate workflows of arbitrary complexity.

Bringing together the concepts described, the overview of the workflow-based architecture for manufacturing control systems is shown in Fig. 6. As described, the workflow sub-referencing mechanism allows a flexible and reliable definition of workflows. Workflows reference the definitions of system component types which define properties, relations, and capability functions (capability modules) of single components. Capability functions allow the encapsulation of


Referencing workflow, W_1
Actors:
$W_1.a_1$ $W_1.a_2$ $W_1.a_3$ $W_1.a_4$ $W_1.a_5$
„Calling" workflow $W_2(W_1.a_2, W_1.a_4)$
Referenced workflow, W_2
Actors:
$W_2.a_1$ $W_2.a_2$
Signature: $W_2(AGV\_y, Machine\_Tool\_x)$
Component type: Machine_Tool_x
Component type: AGV_y
Component type: Robot_z
Manufacturing system's component model
Figure 5: Workflow referencing mechanism

"physical" abilities of the manufacturing equipment and link the semantic description of the processes as seen on the shop floor level with the command execution (controller) level.

So far the discussion was only about the definition of the workflows. Another issue is the execution of the workflows, which requires certain intelligence concerning the decisions available with respect to the variable assignment and the selection of time moments for certain actions defined in workplans. In the case of multiple alternatives the assignment of real system component instances to actors is not unique. In order to find the best scenario the formulation and solving of an optimization problem becomes necessary. The benefit of the described architecture (in contrast to other approaches, where the complete information is not available, e.g. if the control logic is separated from the process model) is the logically complete description of transformations taking place as a result of the workflow execution. A complete model description is thus an inevitable condition for planning and predicting events in automated manufacturing systems and can be used while defining and solving the optimization problems, dealing with the workflow execution. The approach used to formulate and solve such optimization problems will be discussed in the next section.

Concerning the realization of the workflow-based architecture, nowadays certain tools and technologies are available, e.g. frameworks like Microsoft .NET 3.0 Workflow Foundation or Java Workflow Tooling (JWT). Such frameworks offer predefined design patterns, e.g. concepts like activities, workflows, means for their parallel execution and synchronization etc., thus, simplifying the development of workflow-driven applications.

B. Comprehensive object-oriented constraint model

Two aspects must be considered in order to realize the predictive control and optimization. Firstly, a consistent description of the workflow structure is required, and secondly, the execution of such workflows requires the formulation and solving of arising optimization problems. Although the general concept concerning the workflow description was introduced in the previous sections, it still was not shown how to describe all the rules and restrictions concerning the workflow structure.

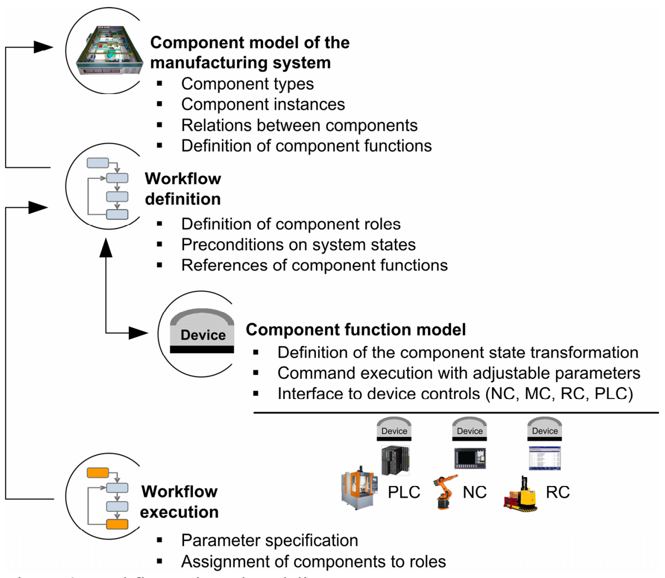

Component model of the manufacturing system
Component types
Component instances
Relations between components
Definition of component functions
Workflow definition
Definition of component roles
Preconditions on system states
References of component functions
Device
Component function model
Definition of the component state transformation
Command execution with adjustable parameters
Interface to device controls (NC, MC, RC, PLC)
PLC
NC
RC
Workflow execution
Parameter specification
Assignment of components to roles

Figure 6: Workflow-oriented modeling of the manufacturing control systems

Such rules for conditional transformations of the manufacturing system's state may be arbitrarily complex and concern the modifications of the states and relations of single system components as well as the precedence of operations. Some constraints might be global others might refer only to certain workflows. In order to describe these dependencies, a generic description language for constraints and rules is required.

One of the technologies being applicable for describing and solving complex optimization problems is the Constraint (Logic) Programming (CP). Its main idea is to describe an optimization problem by explicitly describing its solution space through constraints. An optimization problem is described in this case by means of variables each possessing a non-empty set of possible values (domain) and functional relations between these variables, called constraints, and stating which combinations of values are allowed. One of the key ideas of CP is that constraints can be used "actively" to reduce the computational effort needed to solve combinatorial problems [6]. By means of deductive "constraint propagation" it becomes possible to eliminate variable values, violating some constraints, and thus reduce the solution space of the optimization problem. Additional algorithms, decoupled from the constraint-based problem formulation can be used in order to guide the search for optimal solutions in the solution space. It makes it possible to separate the logical representation of the problem from the control over the problem solution.

In spite of the described benefits of the constraint programming, it also has some drawbacks considering its classical form. The formalization of an optimization problem requires an identification and definition of decision variables and linking them by means of functional or logical relations. This implies a low level of abstraction, since the direct link to the entities (classes and objects) from the business domain is getting lost. Object-Oriented Constraint Programming (OOCP) offers an opportunity to overcome these problems. The idea of OOCP is to state constraints in terms of classes, objects and their relations [7]. The idea is very similar to the Object Constraint Language (OCL) in UML where constraints can be stated on class properties and relations. A promising approach is the synthesis of both the OCL and the constraint propagation [8] in order to enable the application of propagation techniques to the problems described in terms of object-oriented constraints. Unfortunately to the best of our knowledge neither commercial nor freely available software implementations of an OOCP framework are available at the moment which could be used both, for generic modeling and the solution of optimization problems.

Constraint programming allows for a high flexibility while designing custom search algorithms for optimization problems. The range of possible actions refers to the order in which variables are assigned values as well as the order in which domains of those variables are inspected. While dealing with classified decision variables (which is the case for OOCP) it becomes possible to distinguish variables based on their type or associated properties. The search algorithm in this case of an OOCP may depend on the class model of the problem. The drawback of existing constraint systems is the isolated view on the search algorithms. Existing Constraint Solvers (e.g. ILOG Solver 6.2) enable only hard-coded definitions of search heuristics. The favored approach would be to integrate the search algorithm definition into the constraint meta-model. This is currently being investigated in our research work.

C. *Ontology-based integrated modeling*

In the previous sections concepts related to the semantic structure of meta-models for manufacturing control systems were examined. As shown, the meta-model as well as models describing custom cases (custom manufacturing systems and processes) become very sophisticated. The present section discusses the organization of the meta-modeling process and proposes an approach suited for development of configurable applications.

Usually three main components of any software system can be distinguished: (i) data tier, (ii) logic tier and (iii) presentation tier [9]. This applies to any data processing application, since in any case it is necessary first of all to define the data meta-model for the application and provide means for storing and assessing the data (data tier). The business logic (logic tier) which

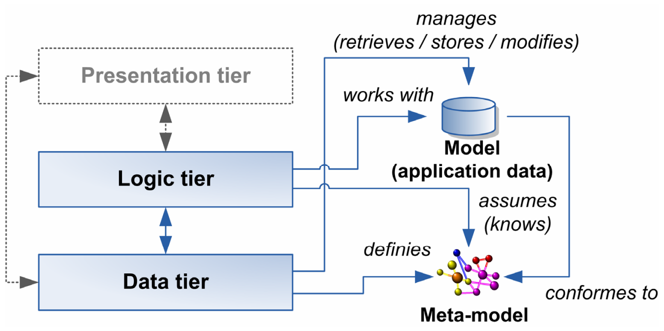


Figure 7: Classical three-tier architecture

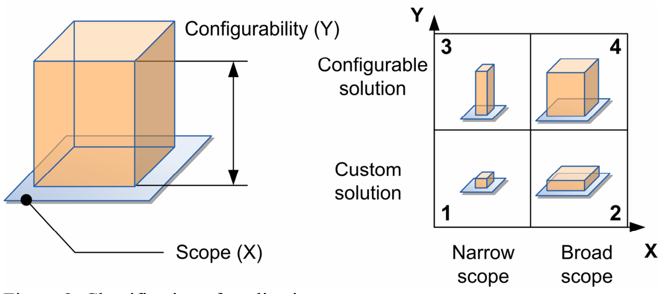


Figure 8: Classification of applications

encompasses algorithms to process the data is designed based on an assumption of a certain data structure (see Fig. 7). Thus, there exists a direct dependency between the data and logic tiers. All changes in the data meta-model affect the business logic and require its partial or complete reengineering.

The lifecycle of most software applications is characterized by two phases: application design phase and operation phase. As a rule the requirements of the business domain are analyzed and the architecture and functions of an application are fixed during the design phase. This refers as well to the data meta-model of the future application. This classical scheme is not suitable anymore, if an advanced configurability becomes one of the major requirements for the future application. The problem becomes particularly challenging if the scope of the application (as determined by the addressed business domain) is rather broad, but still a high configurability during the operation phase is required (case 4, Fig. 8). In this case an application transforms to a framework enabling the customization of end-applications.

The aim of a framework is to describe basic phenomena from the relevant business domain and to provide the basic common functions in order to simplify the development of custom, case-specific applications as much as possible. In contrast to the classical application the lifecycle of a framework consists of three phases: (i) framework design, (ii) framework customization and configuration for some custom application scenario and (iii) the operation of the customized application. In relation with the framework idea the development process of the final application is thus spread over the phases (i) and (ii). In most cases it requires at least two meta-models to be developed and managed, one for the framework itself (meta-model I) and the second one within the framework for the final application (meta-model II), as shown in Fig. 9, a. Consequently, a set of rules and concepts describing the data a final application operates with is defined during the framework design phase and another one during the framework customization phase. Due to the separate meta-models it becomes impossible to examine the formal definition of the rules and concepts, defined in the meta-model I (reflection), and to extend them in some way (derivation) from within the meta-model II.

Related to the manufacturing control software the meta-model I is given by the formalization presented in the previous sections. It defines such concepts as a workflow, workflow referencing, manufacturing component type, manufacturing component instance, constraint types etc. The development of a specialized application supposed to control a certain manufacturing system requires a definition of custom classes of manufacturing components and custom instances as well as custom workflows, which are described by custom constraints (meta-model II). So in this case it would be advantageous to define some basic constraints which relate to all entities of some kind (e.g. to introduce a concept of a system component and admit e.g. an “assignment” relation on components) within the meta-model I. A meta-model II might extend this definition restricting the types and the cardinalities of the system components which can be “assigned” to each other, defining additional constraints if required.

The proposed solution approach thus strives for the integration of the meta-model space which will allow a transparent examining, referencing and extension of concepts defined on the framework-level during the design of customized applications. The proposed solution approach is inspired by the ontology models allowing a hierarchical referencing of dependent ontologies. The realization provides a mechanism allowing a definition of the meta-model I and its subsequent transformation to a program code supporting a reflective processing of the own meta-model and its extensions. The abstractions defined in the meta-model thus can be fully used and extended in derived meta-models, as illustrated in Fig. 9, b. At the moment, to the best of our knowledge, no other solutions are available which would support such an integrated framework development process.

III. Realization

The concepts presented in the previous sections are brought together in an integrated modeling infrastructure, whose realization is based on Microsoft DSL Tools [10]. The very basic meta-model is defined by means of DSL Tools, introducing such concepts as classes, instances, relations between classes and instances, expressions, paths and constraints on class members and relations. This meta-model mimics the concepts available in UML and OCL with the difference that it incorporates both the class and the instance

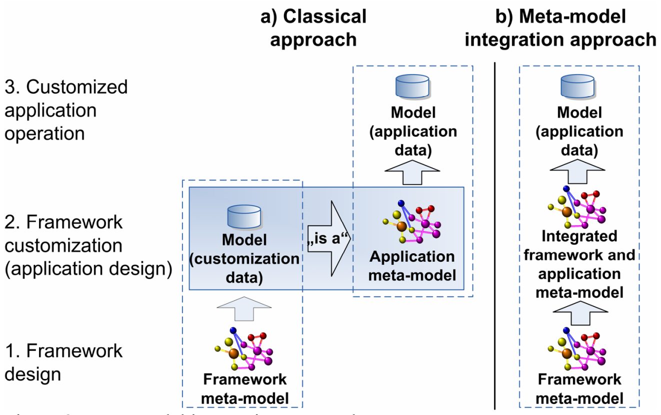
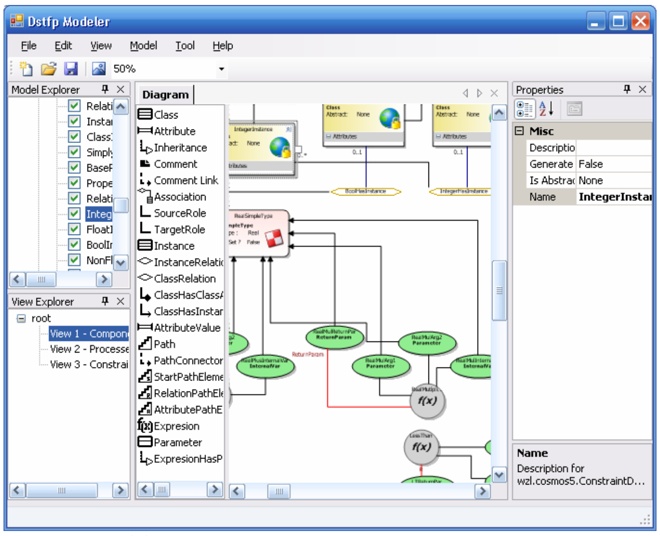

a) Classical approach
b) Meta-model integration approach
3. Customized application operation
2. Framework customization (application design)
1. Framework design
Model (application data)
Model (customization data)
„is a“
Application meta-model
Framework meta-model
Model (application data)
Integrated framework and application meta-model
Framework meta-model

Figure 9: Meta-model integration approach



Descriptio	
Generate	False
Is Abstrac	None
Name	IntegerInstar

Name
Description for wzl.cosmos5.ConstraintD...

Figure 10: Modeler GUI

levels within the same model space. Reflection is thus essentially possible in all models. In contrast to the UML a derivation of relations is added to the basic meta-model, which increases the descriptiveness of the models considerably. The concepts specific to the business domains addressed, i.e. manufacturing systems and processes, such as workflows, manufacturing system components and relevant constraints are all modeled by means of this basic meta-model.

The developed code generation toolkit allows the generation of .Net classes (as well as their instantiation) directly from models. Methods enabling the examining of the meta-model space (reflection) are added automatically to all generated classes. The code generation may be done partially for some model parts. Thus, code can be generated only for those model elements which belong (from the user's point of view) to the meta-model I. A meta-model II is accessible in the same modeling space but exists only in memory and can be persisted if necessary.

A GUI providing a visual design of models is possible (using the designer functionality of MS DSL Tools), see Fig. 10. It simplifies the editing and understanding of models by the user, making it possible to visualize the relevant dependencies. A multi-view interface extension was developed for the DLS tools, allowing the user to create and manage multiple views on the same model.

IV. Conclusion

The presented paper analyzed the requirements on architectures and data models of control systems for automated manufacturing systems. The diversity of practical cases requires such models to become widely configurable. In order to improve the effectiveness of use and maximize the utilization of the manufacturing system a higher intelligence of the control system is required, thus necessitating a predictive control and scheduling of operations. The amount and the organization of the information required for such intelligent predictive control was analyzed. As a result a comprehensive ontology-model being able to describe the processes and the architecture of different manufacturing systems was introduced, which consists of the integrated component, workflow and constraint sub-models. The workflow-based architecture enables a flexible definition of manufacturing processes and their self-referencing. The mechanism enabling the mapping of the semantic description of the transformations within the manufacturing system on the one side and the execution of commands on the other side was developed. Finally, a framework architecture for manufacturing control solutions and a solution for meta-model integration issues were introduced.

Although the problems discussed were related to Flexible Manufacturing Systems, some of the concepts described can be extended to other domains. The workflow orientation clearly can be applied to the modeling of other systems (business workflows etc.). Issues related to the framework architecture and model integration can be furthermore used while designing other software systems, which require high configurability. The particular complexity of the considered use-case (integrated manufacturing control and scheduling for flexible manufacturing systems) is determined by the combination of multiple factors: high required configurability of solutions, complexity and diversity of manufacturing system architectures, complex workflow structures effecting a transformation of system components' properties, high diversity of constraints and the need to interface the mechanisms for the command execution with the semantic description of the transformations.

References

[1] G. Hofmann, "Flexible Fertigungssysteme von heute", in *Werkstatt und Betrieb*, vol. 3 (133), 2000, pp. 46-49.

[2] M. Weck, *Werkzeugmaschinen. Band 1 – Maschinenarten und Anwendungsbereiche*, 5th ed. Berlin: Springer-Verlag, 1997.

[3] *Web Services Business Process Execution Language. Version 2.0. Committee Draft*. OASIS (2006, May 17). Available: http://www.oasis-open.org/committees/download.php/18714/wsbpel-specification-draft-May17.htm

[4] *Workflow Management Facility Specification*. *Version 1.2*. OMG (2000, April). Available: http://www.omg.org/docs/formal/00-05-02.pdf

[5] D. Hollingsworth, "The Workflow Reference Model 10 Years On", in *2006 Workflow Handbook*, L. Fischer, Ed. Lighthouse Point, FL: Future Strategies, 2006.

[6] P. Baptiste, C. Le Pape, W. Nuijten, *Constraint-Based Scheduling. Applying Constraint Programming to Scheduling Problems*. Norwell, MA: Kluwer, 2001.

[7] M. Paltrinieri, Some remarks on the design of constraint satisfaction problems. In A. Borning, Ed., in *Second International Workshop on Principles and Practice of Constraint Programming (PPCP-94)*, vol. 874 of *Lecture Notes in Computer Science*. Berlin: Springer, 1994, pp. 299-311.

[8] G. Renker, A Modeling Framework for Constraints, in *Proceedings of CP-02*, vol. 2470. Berlin: Springer, 2002, pp. 773-774.

[9] R.S. Pressman, *Software engineering: a practitioner's approach*, 5th ed. New York, NY: McGraw Hill, 2001.

[10] *Microsoft Domain-Specific Language (DSL) Tools. Visual Studio 2005 SDK Version 4.0*, February 28, 2007. Available: http://msdn2.microsoft.com/en-us/library/bb126235(vs.80).aspx