On the Suitability of Modeling Approaches For Engineering Distributed Control Systems

A. Lüder[1], L. Hundt[1], S. Biffl[2]
[1] Otto-von-Guericke Universität, Faculty Mechanical Engineering, Center Distributed Systems,
Universitätsplatz 2, D-39106 Magdeburg, Germany
{arndt.lueder|lorenz.hundt}@ovgu.de
[2]Institute of Software Technology and Interactive Systems, Vienna University of Technology
Favoritenstr. 9/188, A-1040 Vienna, Austria
Stefan.Biffl@tuwien.ac.at

***Abstract*- Modern complex and flexible manufacturing systems require new types of control system architectures and design methodologies. An essential new applied architectural approach is the distributed control architecture and an essential new methodology within automation is model-based engineering. There is a wide range of modeling approaches available from traditional automation, domain-oriented systems engineering, and software engineering. Unfortunately, there are very few reports on the pros and cons of these modeling approaches for the practical design tasks along the engineering process for distributed control systems (DCSs). This paper gives a first evaluation for the suitability of models for engineering DCSs to provide the researcher and practitioner with combinations of models that are well suited for their engineering projects.**

I. INTRODUCTION

An essential trend within industrial control is Model-Based Engineering (MDE) and the implementation of increasingly distributed control systems (DCSs) [1,2]. MDE approaches range from Model-Driven Development (MDD) to Model-Based Verification (MBV). However, the modeling approaches used for different engineering tasks are often not easy to combine in an effective and efficient way, since some types of models are not compatible while others need significant extra effort for content translation between model notations.

To enable a consistent model-based approach, at least for DCSs, it is necessary to find a compatible and efficient combinable set of models that allows describing all relevant information for the desired model-based approaches. Since MDE and MBV are particularly relevant for research and practice, this paper focuses on these types of engineering processes.

The aim of this work is to provide an overview on relevant candidate model types and evaluate these model types for their suitability to support MDE and MDV processes for DCS. The remainder of the paper is structured as follows. Section 2 defines the modeling needs for DCS engineering. Within Section 3 follows a summary of the selected model types for this analysis. Section 4 motivates the evaluation criteria for model type suitability and explains the process for model type selection. Section 5 presents the detailed results of the analyses and gives a rating for the applicability of different types of models within the design process of DCSs this is followed by a conclusion in section 6.

II. MODELING NEEDS FOR DCS ENGINEERING

The key research question of this paper is how to select a set of models to support MDE and MDV in a project with the goal to design a DCS. To answer this question we derive 3 research issues: 1. *Engineering process steps*. Define the engineering process steps and information sets that the models are to support. 2. *Model type definition*. Motivate the set of model types that need to be considered to support the engineering process steps. 3. *Evaluation procedure and criteria*. Evaluate the pros and cons of candidate model approaches.

Engineering Process Steps. The scope of application scenarios of models in the design process and the data needed for MDE and MDV drive the selection of models and the model evaluation.

With the progress of the design process of a DCS models have to capture information on different levels of scope and detail. On the one hand the scope of information coded within one model decreases. On the other hand the level of detail of the involved systems that is considered increases [3, 4].

Figure 1 shows the usual structure of an engineering process of a manufacturing system. Initially the basic process structure and its inherent sequence of manufacturing steps is defined. Based on this process, the geometry and kinematics of the required manufacturing components are designed within the mechanical engineering including the cell and line layout. Afterwards, the electrical engineering, the PLC and robot, and the HMI programming are started exploiting the results of the prior steps. Finally, the complete cell including the geometry, kinematics, logic and I/O design can be passed into a virtual commissioning environment.

In early design phases mainly the addressed manufacturing process gets mapped to control activities on a high level of requirements. Based on given *Bills of Operation* and *Bills of Material* of the products to be produced and the physical layout of the manufacturing system required control processes, sequences of control activities, and sequences of input/output combinations are developed for the complete system on a global level. The sequences of control activities are used to express the control system behavior in relation to the system environment including the user and controlled system which

978-1-4244-3760-3/09/$25.00 © 2009 IEEE

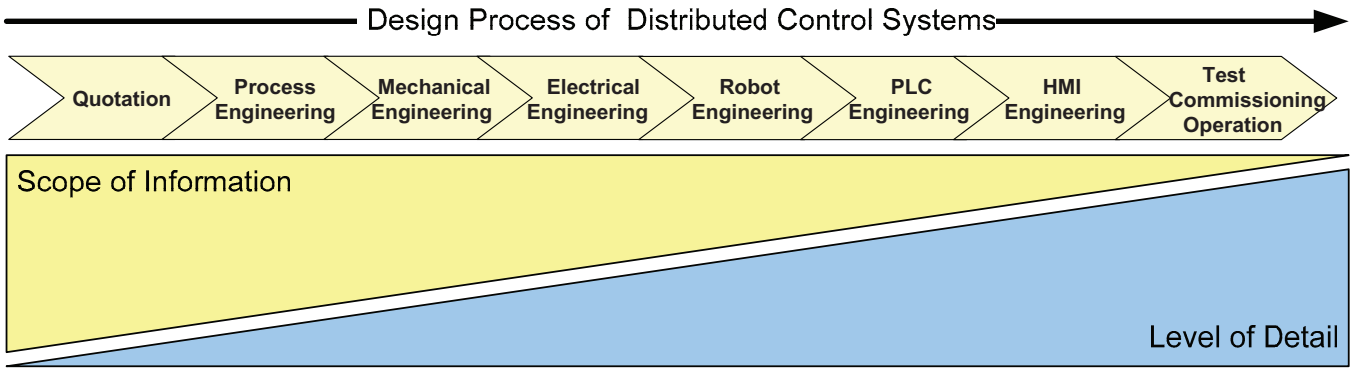

Fig 1. Information content of models used along the DCS Design Process

are necessary for contract preparations including the functional specifications and contract specification.

In later design phases the requirement specifications gets detailed with respect to four main fields of interest:

- *control system components and devices* like PLCs, sensors, actors, communication systems;
- *sequences of control* activities covering the necessary data process and control decisions to ensure the relevant manufacturing process;
- *control activities for system safety and security* preventing dangerous system behavior; and
- *distribution* of control activities, control decisions, and communication activities within the system of applied control components.

Within the model-based engineering processes (i.e., MDE and MBV) behavior models of the system describe the control-system behavior, the controlled-system behavior, the uncontrolled-system behavior, and the closed-loop behavior [16].

MDE is a special way of exploiting sets of guidelines for structuring specifications expressed as models within system engineering and application. Thereby, MDE separates design from implementation. Decoupling design from implementation allows system developers to choose from the best and most fitting approaches in both domains. MDE exploits (implementation) platform-independent models (PIM) and platform-specific (dependent) models (PSM), where the PSM usually implements the PIM [5].

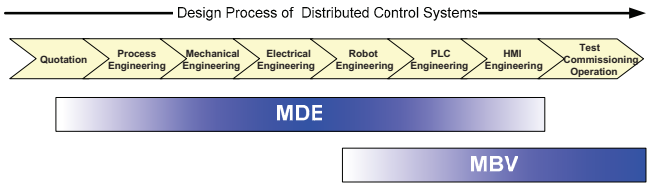

Fig. 2. Scope of MDE and MBV along the DCS Design Process.

Implementation technologies for control applications, particularly, distributed control applications are:

- *programming languages* IEC 61131-3, C, C++, Java;
- *for devices* PLCs, SoftPLCs, CNCs, RNCs, etc.;
- *for the communication system* field busses like Profibus, Interbus, CAN; and Industrial Ethernet protocols like PROFINET, EtherNet/IP, EtherCAT.

MBV exploits formal behavior models of the closed-loop behavior and models of the intended system behavior for formal comparison. Usually the intended behavior is coded by specifications of intended and avoided system states and behavior sequences while the closed-loop behavior is generated from the formal combination of models of the uncontrolled plant behavior and the control system behavior [17].

In the case of DCSs the models of the uncontrolled plant behavior and the control system behavior have to reflect the distribution of control decisions by combining behavior model components and interaction structures in a proper way resulting in modular and cooperative system models.

Modeling needs: information sets. In summary the information models to be used within the engineering of DCSs have to be able to cover the following sets of information:

1. *Architecture*: distribution of control decisions on system components;
2. *Processes of control decisions* consisting of process states and process state transitions;
3. *Signals and states*: logical combinations of input signals, output signals, and internal states;
4. *Communication:* distribution of control-relevant information;
5. *Control-relevant data structures*; and
6. *Interaction and communication processes* between control decision taking components and processes.

Section 3 provides an overview on types of models that have at least some capabilities to model these 6 sets of information.

III. Model Types for DCS Engineering

Various types of models are needed to cover the information needs along the DCS engineering process. Following [6, 7] here a set of model classes with some of its model types will be considered. The applicable models can be classified as follows.

A. Cognitive models and natural means for description

This class of models summarizes mental and linguistic models, natural and material languages, and taxonomies.

Mental and linguistic models are based on the application of the activity of the human brains and the human languages [13]. They exploit a layered structure starting with basic sig-

TABLE I
DETAILED EVALUATION RESULTS

Model type	Statesystems						Modularity							Data					Control flow						
	Global states	Global state transitions	Local states	Local state transitions	Timing	Processes/activities within states	Modules	Hierarchies	Interfaces	Event signals	Data signals	Connection between events and data	Composability	Local data	Global data	Distinction of data types	Connection between data and local states	Connection between data and global states	Sequence of control decisions	Data flow among control decision taking processes/entities	Involved decision taking entities	Degree of formality	Availability of analysis techniques	Availability of analysis tools	Expressability of requirements	Availability of exchange format
Mental and linguistic models	--	--	--	--	--	--	+	+	--	o	--	--	--	--	--	--	--	--	--	--	+	--	--	--	+	--
Natural and material languages	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	--	--	--	+	o
Taxonomical systems	--	--	--	--	--	--	+	++	o	--	--	--	++	--	--	--	--	--	--	--	--	o	--	--	--	o
Boolean Algebra	++	o	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	+	--	o	++	++	++	o	o
Boolean Differential Calculus	++	++	--	--	o	o	--	--	--	--	--	--	--	--	o	--	--	o	++	o	o	++	++	++	o	o
Max-Plus-Algebra	++	++	--	--	o	o	--	--	--	--	--	--	--	--	o	--	--	o	++	o	o	++	++	o	o	--
First order logic	o	o	o	o	o	o	o	o	o	o	o	o	o	o	o	--	o	o	o	o	o	+	+	o	+	o
Ladder diagram	--	--	+	+	+	+	--	--	+	o	+	--	--	++	+	+	+	--	+	o	o	--	o	o	o	++
Logic plan	--	--	+	+	+	--	--	--	+	o	+	--	+	++	+	+	+	--	o	o	o	--	o	o	o	++
Entity Relationship diagrams	--	--	--	--	--	--	++	++	o	--	--	--	++	--	--	--	--	--	--	--	o	+	o	o	--	--
Fault Trees	--	--	+	+	--	--	--	--	--	--	--	--	+	--	--	--	--	--	o	--	--	o	+	o	o	--
Class diagrams	--	--	+	+	o	+	++	++	++	+	+	+	++	++	o	++	++	--	--	+	++	--	--	--	+	++
Package Diagrams	--	--	--	--	--	--	++	++	++	--	--	--	+	--	--	--	--	--	--	--	o	--	--	--	--	++
Gantt charts	+	+	+	+	++	--	--	--	--	o	--	--	--	--	--	--	--	--	+	--	o	o	o	o	--	--
Sequential Function Charts	+	+	++	++	++	++	o	+	+	o	+	o	o	++	++	++	++	+	++	o	o	o	o	+	--	++
IEC 61499 function block systems	--	--	++	++	++	++	++	++	++	++	++	++	++	++	--	++	++	--	+	++	++	o	o	o	--	++
Sequence diagrams	--	--	--	--	--	--	+	--	++	+	+	+	--	o	--	--	--	--	+	++	+	o	o	o	--	++
Communication diagrams	--	--	--	--	--	--	+	--	++	+	+	+	--	o	--	--	--	--	o	++	++	o	o	o	--	++
Program flow charts	+	+	--	--	--	+	--	--	+	+	+	+	--	o	+	o	o	+	++	o	o	o	o	o	--	--
Data flow diagrams	+	+	--	--	--	+	o	--	+	+	+	+	--	+	+	+	--	+	++	+	+	o	o	o	--	--
Collaboration diagrams	o	o	o	o	o	o	+	+	+	o	o	o	+	o	--	--	--	--	--	+	++	o	o	o	+	++
Programming languages	++	+	++	+	+	+	+	+	+	++	++	+	+	+	+	+	--	--	--	o	o	--	--	--	--	+
State Charts (Harel automatons)	+	+	o	o	+	+	+	+	o	+	+	o	o	o	+	o	o	+	++	++	o	++	++	++	o	o
Activity diagrams	+	+	o	o	+	++	+	+	+	+	+	+	o	+	+	+	o	+	++	++	o	+	+	+	o	++
State diagrams	+	+	o	o	--	+	o	+	o	+	+	o	o	+	+	+	o	+	++	++	o	+	+	+	o	++
Petri Nets	--	--	++	++	+	--	--	--	--	--	--	--	--	o	--	--	++	--	+	+	o	++	++	++	+	+
Net-Condition/Event-Systems	--	--	++	++	+	--	++	++	++	++	++	--	++	+	--	--	++	--	+	++	+	++	++	+	+	o
Impulse diagrams / Timing diagrams	--	--	++	++	++	+	+	--	+	++	o	o	o	+	--	--	+	--	+	++	+	+	+	o	o	++
Use Case diagrams	--	--	--	--	--	--	+	--	+	--	--	--	o	--	--	--	--	--	--	+	++	--	--	--	++	++

nals which are combined to words and sentences while these grammatical or syntactical objects will get a meaning on a semantic layer.

Natural and material languages are based on the a categorical classification of objects, a collection of objects and its meaning in dictionaries, and the analysis of information regarding grammar, words with meaning, and the structure of sentences with respect to truth values [14].

Taxonomical systems are based on descriptive terms and its classification structure [15]. They provide a hierarchy of descriptions with semantics and relations among them.

B. Algebraic-logical models

Algebraic-logical models subsume propositional logic and Boolean algebra, Boolean Differential Calculus, Max-Plus-Algebra, and further logics like first order logics.

Boolean algebra is based on the connection of logical values by logical operators to new logical terms. The Boolean Differential Calculus extends this logic by means of differential equations over Boolean expressions.

Max-Plus-Algebra is used to express processes. Therefore it combines states with an existence interval with two types of operators expressing process synchronization and process time progression.

First-order logic expresses the relations among logic expressions. It enables the description of dependencies and independencies of logical facts as well as the validity of expressions for sets of objects/expressions.

C. Implementation-oriented models

The next class of models contains implementation-oriented models. Within this class there is a subclass of s*tructure-oriented models*. This subclass contains Ladder diagram, Logic plan, Entity Relationship diagrams, Fault Trees, Class diagrams, and Package Diagrams. The second subclass is related to process-oriented descriptions and contains Gantt charts, Sequential Function Charts, IEC 61499 function block systems, Sequence diagrams, and Communication diagrams.

Ladder diagram is part of the IEC 61131-3 programming languages. The Ladder diagram is based on electrical flow sheets and enables the graphical representation of connections

between logical values of inputs, outputs and internal variables of a PLC with means of circuit diagrams.

Logic plan is also part of the IEC 61131-3 programming languages. It is a graphical expression of logical relationships between inputs, outputs and internal variables within PLC programs. They are used to encode logical dependencies among sets of Boolean values.

Entity Relationship diagrams provide a graphical notation for relationships among objects covering especially containment relations and Is-Type-of Relations. They can express hierarchical structuring of systems.

Fault Trees express a logical combination of events and states to enable special types of system behavior.

Class diagrams are one element of the UML/ SysML set of models [12; 18]. They are used to model relationships among objects as well as their internal behavior, the data, and the interfaces. Class diagrams enable the expression of object dependencies, object containment hierarchies, object specialization, etc. Thereby, class diagrams enable the description of the complete modular structure of larger systems.

Package Diagrams are part of the UML/ SysML [12; 18]. They describe the hierarchical structure of systems (especially the containment relations of software components).

Gantt charts describe the timing of processes and process steps. They depict the execution order of processes over a timeline and can be used to represent logical relations among process steps.

Sequential Function Charts (SFC) is a programming language of IEC 61131-3. It provides a process-oriented description of systems consisting of half-ordered sequences of steps and transitions and activities attached to steps. Within the activities variables can be manipulated, which are used within the transitions as firing conditions. Initially, SFCs were intended to provide a means for step-chain programming. But the temporal conditions attachable to an activity enable its execution scheduling widely independent from steps. Thereby, SFCs can be seen as an extended implementation of automaton execution systems.

IEC 61499 function block systems (FBSs) have been defined in IEC 61499. FBSs provide a modeling means for communication automatons based on data and event flows. Initially, FBSs were intended to describe DCSs. With the development of IEC 61499 Function Block Network execution environments FBSs start to become a programming language.

Sequence diagrams are a model type of UML/ SysML [12; 18]. They express the sequential communication structure between entities, especially covering the exchange of information and the call of object functionalities.

Communication diagrams are also a model type of UML [12]. Similar to sequence diagrams they express the data and function call exchange between objects. In contrast to sequence diagrams they concentrate on the interaction structure instead of the sequence of interactions.

D. Programming-oriented models

The fourth class of models is programming-oriented models. This class consists of program flowcharts, data flow diagrams, collaboration diagrams, and programming languages.

Program flowcharts are used to model the partially half ordered sequence of procedures and decisions taken within a program within information sciences. They can be exploited to describe each type of complex process consisting of different process steps and decisions.

Data flow diagrams (DFDs) enable the description of functional dependencies among different processes/ functions by the exchanged data. DFDs provide means for the modeling of information flows and information storing among and within information processing functions of a system.

Collaboration diagrams are a diagram type of UML [12]. They are used to depict role-based structural dependencies among system components. Especially, they are applied for the description of design patterns.

Strictly speaking, textual programming languages are no real modeling means. But several programming languages (especially textual PLC programming languages of IEC 61131-3 like Instruction List or Structured Text) can be formally analyzed based on their internal grammar. Hence, they can be seen also as models.

E. State-and-behavior-oriented models

The last class is related to state- and behavior-oriented models. It covers the models coded in Automatons, Activity diagrams, State diagrams, Petri Nets, Net-Condition/Event-Systems, Impulse/Timing diagrams, and Use Case diagrams.

Automatons are formal models. Most familiar automatons are the Moore automaton, Mealy automaton [8], and the State charts of Harel [9]. They are all used to express global system states and their state change dependencies including incoming events triggering state changes and outgoing events signalizing the reached states. In addition State charts provide means for modeling of hierarchical states and parallel states to enable modularity, concurrency, and composability.

Activity diagrams are a diagram type of UML/SysML [12; 18]. They depict the network of basic activities a system unit is able to execute. Thereby, they can represent the involved data and control flows as well as state-dependent decisions.

In contrast to activity diagrams, state diagrams, also a model of UML/SysML, depict dependencies of system component states [12; 18]. They are based on State charts.

While automatons describe global system states, Petri Nets are designed to represent local states and state transitions and its mutual dependencies. They are used to express processes based on causal dependencies and concurrencies [10].

Net-Condition/Event-Systems have been derived from Petri Nets and Condition/Event-systems [11]. They extend Petri Nets by explicit signals enabling/disabling as well as enforcing state transitions and, thereby, providing means for the modularization of Petri nets based models.

Impulse diagrams and timing diagrams of UML are a modeling means to express the temporal behavior of a system

component states in combination with its dependencies and exchanged signals [12]. They can be used to describe the signal and event between synchronized system components or processes.

Use Case diagrams are part of UML/SysML developed to express global dependencies of system functionalities [12; 18]. They are used to depict the functional dependencies of global system behavior from involved system components and system environment.

IV. Evaluation Criteria and Procedure

Based on the criteria informally given defined in section 2 the 29 types of models will be evaluated with respect to the following information and modeling capabilities. The evaluated describable 6 information sets (see section 2) will cover the expressive-ness of state systems with global states, global state transitions, local states, local state transitions, timing, and processes and activities within states respectively, modularity including modules, hierarchies, interfaces, event signals, data signals, connection between events and data, and composability, expressible data containing, local data, global data, distinction of data types, connection between data and local states, and connection between data and global states, as well as ability to represent control flows including sequence of control decisions, data flow among control decision taking processes and entities respectively, and involved decision taking entities. The evaluated model-based capabilities will contain: the degree of formality, availability of analysis techniques, availability of analysis tools, expressiveness of requirements, and availability of exchange formats.

For each of this information the expressiveness has been rated in a more high-level way as Table II represents. This is due to the comparability problems of strongly different model types and the problem to provide a fair rating with more levels for differentiation. The basement of this rating was an extensive literature survey and a Delphi research.

TABLE II Detailed Evaluation Results

++	for very well expressible,
+	for well expressible,
O	for fair expressible, and
--	for not expressible.

The Table I shows the results of the evaluation.

V. Evaluation Results and Discussion

The evaluation results enable the conclusion of suggestions for the application of model types within the two main phases of engineering of DCSs.

1. In early design phases best fitting model types for modeling of manufacturing process and its mapping to control activities on a high level of abstraction are Use cases, Class diagrams, and IEC 61499 function block diagrams for structuring and Gantt charts and IEC 61499 function block diagrams for process description. Also well applicable are Package diagrams, Collaboration diagrams, and Net-Condition/Event Systems for system structuring and Sequential Function charts, Sequence diagrams, and Communication diagrams for process description.

2. In later design phases the following types of models can be applied to describe the following information. For modeling of applied control system components Class diagrams, IEC 61499 function block systems, and Net-Condition/Event systems can be exploited best while Communication diagrams can be used in a good way. For modeling sequences of control activities covering the necessary data process and control decision and reflecting system safety and security Sequential Function Charts, IEC 61499 function block systems, State Charts, Activity diagrams, State diagrams, Petri Nets, Net-Condition/Event-Systems, and Impulse diagrams / Timing diagrams can be used very well and Boolean Algebra, Boolean Differential Calculus, First order logic, and Sequence diagrams can be used well. For modeling the distribution of control activities, control decisions, and communication activities within the system of applied control components Class diagrams, IEC 61499 function block systems, and Net-Condition/Event-Systems can be used best and Communication diagrams ad Sequence diagrams can be used good.

Based on this rating we can conclude the correlation of the degree of formality, the applicability within the different phases of the design process, and the commonality of practical usage within the manufacturing control engineering processes by the different classes of model types. This is depicted in the figure 3.

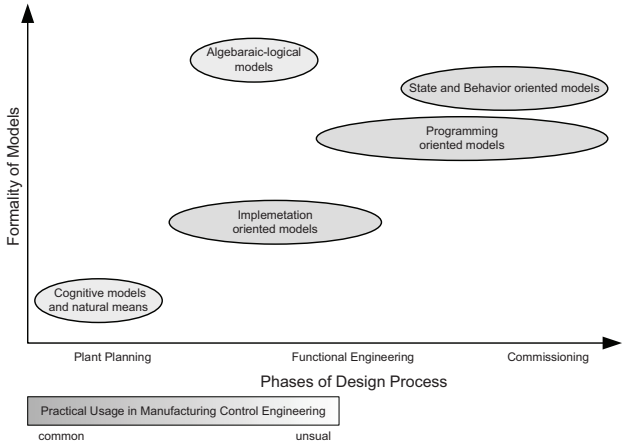

Fig. 3. Formality of Model Classes.

Within MDE the best and good fitting model types are Class diagrams, Package Diagrams, Gantt charts, Sequential Function Charts, IEC 61499 function block systems, Sequence diagrams, Communication diagrams, Collaboration diagrams, State Charts, Activity diagrams, State diagrams, Petri Nets, Net-Condition/Event-Systems, Impulse diagrams / Timing diagrams, and Use Case diagrams.

For MBV Boolean Algebra, Boolean Differential Calculus, First order logic, IEC 61499 function block systems, State Charts, Petri Nets, and Net-Condition/Event-Systems can be exploited best or good. This is given in Table III.

TABLE III
EVALUATION RESULTS FOR DCS ENGINEERING

Modell class	Model type	High level	Low level	MDE	MBV
Cognitive models and natural means for description	Mental and linguistic models	--	--	--	--
	Natural and material languages	o	--	o	--
	Taxonomical systems	--	--	--	--
Algebaraic-logical models	Boolean Algebra	--	+	--	+
	Boolean Differential Calculus	--	+	--	+
	Max-Plus-Algebra	--	--	--	--
	First order logic	--	+	--	+
Implementation oriented models	Ladder diagram	--	--	--	--
	Logic plan	--	--	--	--
	Entity Relationship diagrams	--	o	o	--
	Fault Trees	--	o	o	--
	Class diagrams	++	++	++	--
	Package Diagrams	+	--	+	--
	Gantt charts	++	--	++	--
	Sequential Function Charts	+	++	++	--
	IEC 61499 function block systems	++	++	++	+
	Sequence diagrams	+	+	+	--
	Communication diagrams	+	+	+	--
Programming oriented models	Program flow charts	o	o	o	--
	Data flow diagrams	o	o	o	--
	Collaboration diagrams	+	--	+	--
	Programming languages	--	--	--	--
State and behavior oriented models	State Charts (Harel automatons)	o	++	+	+ +
	Activity diagrams	o	++	++	--
	State diagrams	o	++	++	--
	Petri Nets	o	++	+	++
	Net-Condition/Event-Systems	+	++	+	++
	Impulse diagrams / Timing diagrams	--	++	++	--
	Use Case diagrams	++	--	++	--

According to Table III, the suitability of the single model type classes for the use within MDE and MBV respectively can be deduced. This is depicted in figure 4.

VI. CONCLUSION AND FURTHER WORK

The evaluation results indicate that several sets of model combinations are suitable for MDE and MBV within the engineering of DCSs.

One often executed process within MDE is the automatic generation of control code based on uncontrolled plant models and specifications of the expected behavior. Following the evaluation of section 5 one appropriate model set for this case is Use Cases and Gantt charts for the high level specification of the expected system behavior, Impulse diagrams and activity diagrams for the low level specification of the expected system behavior, State diagrams and State charts for the modeling of uncontrolled plants, and finally IEC 61499 function block systems for modeling the control applications.

A common use case for MBV is the automatic validation of control code based on models of the controlled plant and behavior specifications. Here the most appropriate models seem to be State charts and Net-Condition/Event-Systems for modeling of the controlled plant and first order logics for the modeling of behavior specifications. Nevertheless, the specification models can be generated for example out of Gantt charts and Impulse diagrams.

We see the contribution of this paper as initial step to provide an overview on modeling types and reasoning on their suitability for DSC engineering to support researchers and practitioners who need to select a working set of models for their engineering projects. Further work will be to investigate

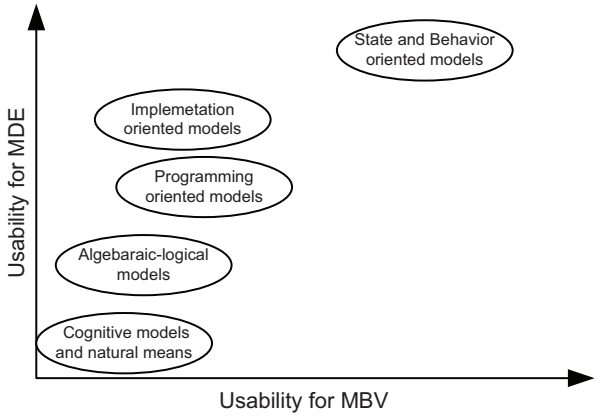

Fig. 4. Suitability of Model Classes for MDA and MBV.

combinations of models for MDE and MBV for their usability and usefulness in practical engineering environments especially with the background of different engineering processes.

REFERENCES

[1] J. Rode, D. Wünsch: A Research Agenda for Adaptive Manufacturing, 12th IEEE Int. Conf. on Emerging Technologies and Factory Automation, (ETFA'07) Sept 2007, Patras, Greece, Proceedings.

[2] H. Kühnle,: Post mass production paradigm (PMPP) trajectories, Journal of manufact: technol: managem:, JMTM.18,8, pp. 1022–1037, 2007

[3] R. Drath; A. Lüder;J. Peschke;L. Hundt (2008): AutomationML. The glue for seamless automation engineering;. 13th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2008, pp. 616–623, September 2008

[4] PABADIS'PROMISE Consortium; White Paper - Structure and Behaviour of a PABADIS PROMISE System, www.pabadis-promise.org, download area.

[5] T. Stahl, M. Völter, S. Efftinge: Modellgetriebene Softwareentwicklung. Techniken, Engineering, Management. d.punkt Verlag, 2007

[6] E. Schnieder: Methoden der Automatisierung, Vieweg verlag, 1999.

[7] VDI/VDE 3681: Classification and evaluation of description methods in automation and control technology, Verein Deutscher Ingenieure, Verband der Elektrotechnik Elektronik Informationstechnik, , Beuth Verlag, Berlin. Oktober 2005

[8] John E. Hopcroft, Jeffrey D. Ullman: Introduction to Automata Theory, Languages and Computation, Pearson; 2006

. [9] David Harel: Statecharts: A Visual Formalism for Complex Systems. In: Sciene of Computer Programming. 8, 1984 (87), S. 231–274.

[10] C. Girault , R. Valk: Petri Nets for System Engineering, Springer Publisher, 2003.

[11] H.-M. Hanisch, A. Lüder: A Signal Extension for Petri Nets and its Use in Controller Design, Fundamenta Informaticae 41, IOS Press, 2000, pp. 415-431

[12] B. Oestereich: Developing Software with UML, Addison-Wesley, 2002

[13] A. Blank: Einführung in die lexikalische Semantik. Tübingen, 2001

[14] Daniel Jurafsky, James H. Martin: Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition. Upper Saddle River, New Jersey: Prentice Hall, 2000

[15] Paul Michel: Ordnungen des Wissens. In: Ingrid Tomkowiak (Hrsg.): Populäre Enzyklopädien. Von der Auswahl, Ordnung und Vermittlung des Wissens. Chronos, Zürich 2002, S. 35–85

[16] R. Boel, G. Stremersch (Editors): Discrete Event Systems – Analysis and Control, Kluever Academic Publisher, 2000.

[17] H.-M. Hanisch, A. Lüder: A Signal Extension for Petri Nets and its Use in Controller Design, Fundamenta Informaticae 41, IOS Press, 2000, pp. 415-431.

[18] Tim Weilkiens: Systems Engineering with SysML/UML, Morgan Kaufmann 2008