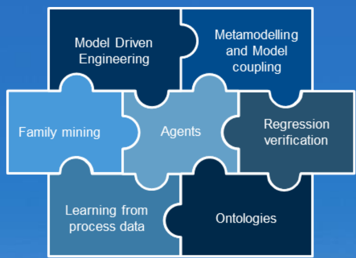
SUMMER SCHOOL
Enabling Technologies – Part II
(Agents, Modeling, Notations for Automation)



Univ.-Prof. Dr.-Ing. Birgit Vogel-Heuser

Ordinaria
Automation and Information Systems (AIS)
Mechanical engineering,
Technische Universität München
www.ais.mw.tum.de; vogel-heuser@tum.de

TUM ASIA SUMMER SCHOOL
24TH – 30TH August 2017

Automation
and Information Systems
Technical University of Munich

Agenda – TUM Asia Summer School 29th August

	29th August	30th August
9:00AM – 10:30AM	**Comparison of Industry 4.0, IoT, Smart Factory, Smart Data**	**Case Studies & Successful Demostrators: Applying Enabling Technologies**
10:30AM – 11:00AM	MORNING TEA BREAK	
11:00AM – 12:30PM	**PART I: Enabling Technologies (Agents, Modelling Notations for Automation)**	**Smart Data Enabled Learning During Operation**
12:30PM – 01:30PM	LUNCH BREAK	
01:30PM – 03:00PM	**PART II: Enabling Technologies (Agents, Modelling Notations for Automation)**	**Security and Human in the Loop**

Complete Agenda: https://tum-asia.edu.sg/i4ss/

Automation
and Information Systems
Technical University of Munich

Outline of part II

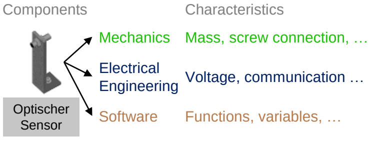
1. SysML4Mechatronics to model cross-disciplinary relations
2. Formal methods: OCL to model constraints
3. Formal methods: inconsistency management based on ontologies

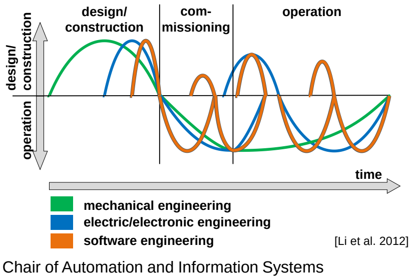
Automation and
Information Systems
Technical University of Munich

Complexity in the development of mechatronic systems

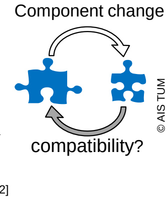
Challenges in the handling of mechatronic systems

- Close interaction of mechanical design, electrical engineering and software development in mechatronic systems → implicit dependencies of the components
- Development and modularization of technical systems are often aligned with mechanics → Lack of mechatronic observation and development of the overall system
- Frequent changes in the system during the utilization phase (disciplines-specific and interdisciplinary) → Lack of analysis of the change effects and lack of return of the changes in the models from the development
- Compatibility of the modified system elements with the existing system must be ensured → Insufficient compatibility analysis of interfaces



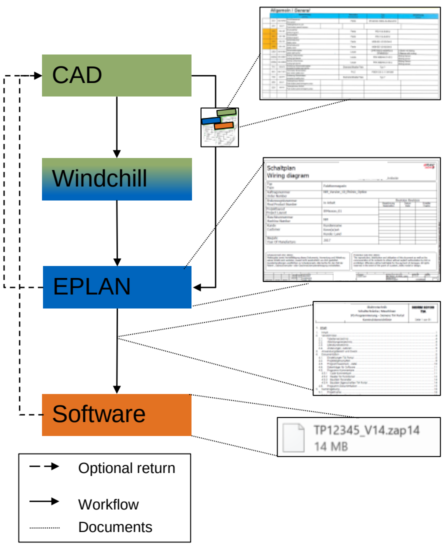


[Li et al. 2012]



Automation and
Information Systems
Technical University of Munich

Status quo: Workflow and Documents



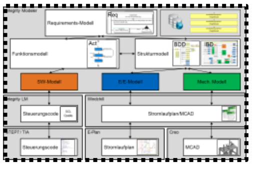
„limit-switch"-plan is the default document to create the EPLAN

Wiring diagram is the result of the electrical/electronics engineering

Coding-Guidelines for the TIA project

TIA project of the plant

- Workflow and document analysis via workshops
- Most significant engineering documents for the real industry plant



© AIS TUM

Automation and
Information Systems
Technical University of Munich

Solution approach SysML4Mechatronics

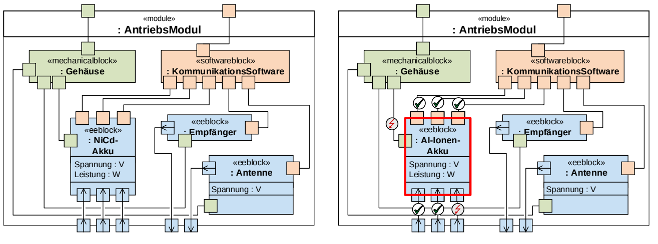
How can the SysML be extended to include the discipline-specific aspects of a mechatronic PSS?

Adaptation of the SysML meta model by a SysML profile for mechatronic systems→ SysML4Mechatronics

- Interaction of the components of the different disciplines and formation of interdisciplinary modules, definition of the element interfaces (ports)
- Modeling the component features (e.g., from product classification systems)
- Analysis of change effects by testing for structural compatibility

Example on PSSycle: Adaptation needs of the bicycle battery to achieve higher range → Some interfaces are compatible, some are not.

Current version (NiCd battery)



New version (Al-Ion battery)

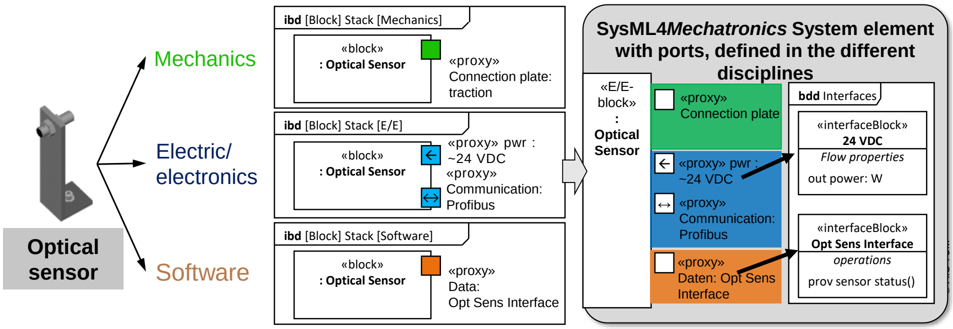
© AIS TUM

Port based modeling of the system structure for analyzing and illustrating the influences of changes

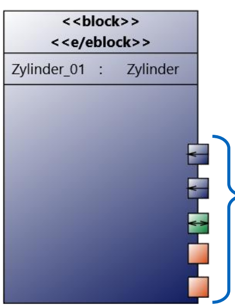
How can the SysML be extended to include the discipline-specific aspects of a mechatronic PSS?

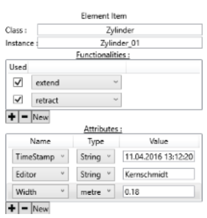
Adaptation of the SysML meta model by a SysML profile for mechatronic systems→ SysML4Mechatronics

- Interaction of the components of the different disciplines and formation of interdisciplinary modules, definition of the element interfaces (ports)
- Modeling the component features (e.g., from product classification systems)
- Analysis of change effects by testing for structural compatibility



Elements of SysML4Mechatronics (excerpt)





Element type:
- Mechanics (e.g., skeleton, mechanical construction)
- E / E: Electrics / electronics (e.g., actuators, sensors)
- Software
- Module: Functionally related elements

Ports: (Specified in the interface block)
- Mechanics (e.g., mechanical connections)
- E / E: Electricity / electronics (e.g., communication, power supply)
- Software (e.g., offered, required interfaces)

Class: Type of element (template)

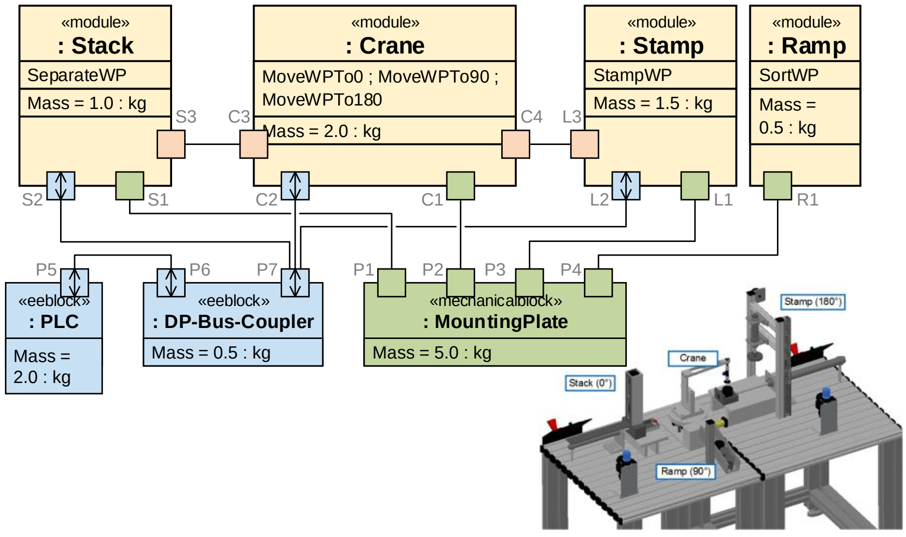
Instance: Specific, used component of the *class*

Functionalities: Functions that the component performs

Attributes: Features of the component (classified as a basis for compatibility)

Automation and
Information Systems
Technical University of Munich

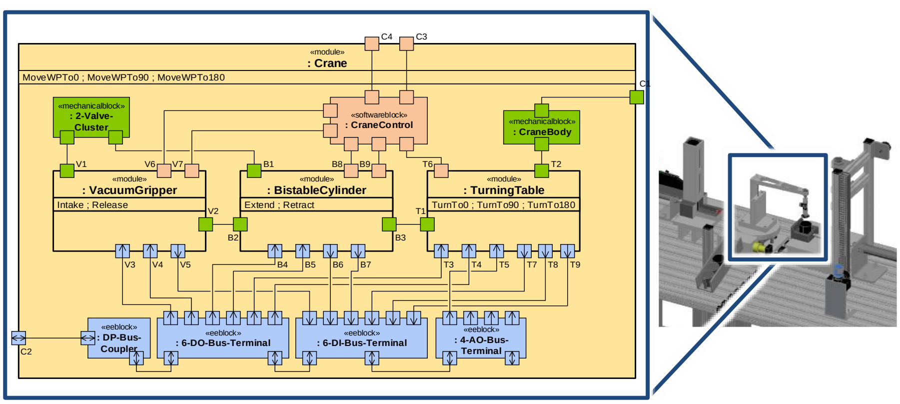
Example SysML4*Mechatronics* model: PPU



© AIS TUM

Automation and
Information Systems
Technical University of Munich

Modeling in SysML4Mechatronics? Example: crane of the PPU

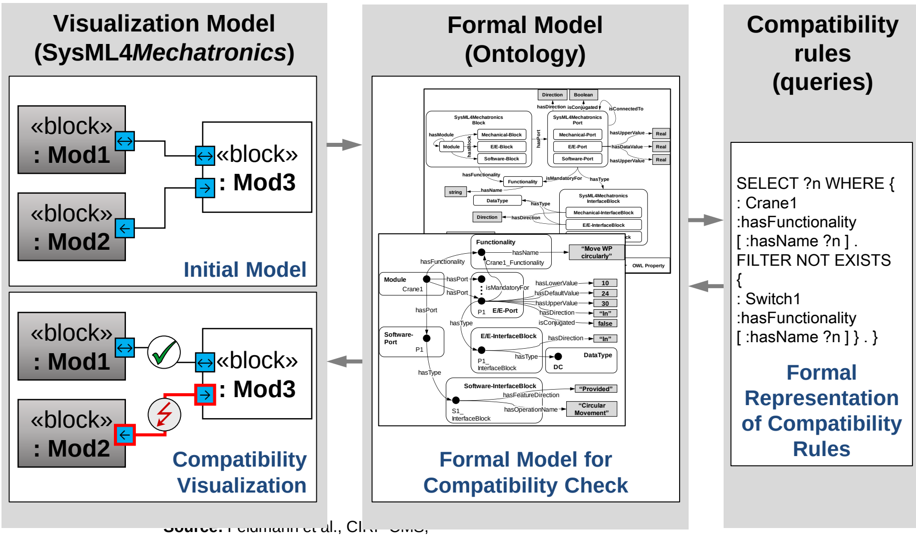


- Display of discipline-specific components (mechanical, electrical / electronics, software), interdisciplinary modules and visualization of interfaces and interdependencies

© AIS TUM

Automation and
Information Systems
Technical University of Munich

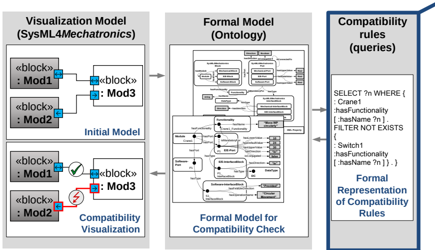
Compatibility check through transformation into formal model



Source: Feldmann et al., CIRP CMS, 2014

Automation and
Information Systems
Technical University of Munich

Compatibility check through transformation into formal model



- Compatibility rules enable modelling of compatibility criteria based on the component / module-properties
- **Inherent compatibility rules** need to be fulfilled by each model
- **Application-specific rules** extend the framework by further e.g. plant-specific compatibility criteria

Compatibility Rules

Inherent

- **Data type compatibility:** Same data types
- **Direction compatibility:** In ↔ Out, InOut ↔ InOut, In ↔ InOut, Out ↔ InOut
- **Range compatibility:** Range(In) ≥ Range(Out)
- **Operation:** Same required/ provided operation
- **Port fulfilment:** Mandatory ports must be connected

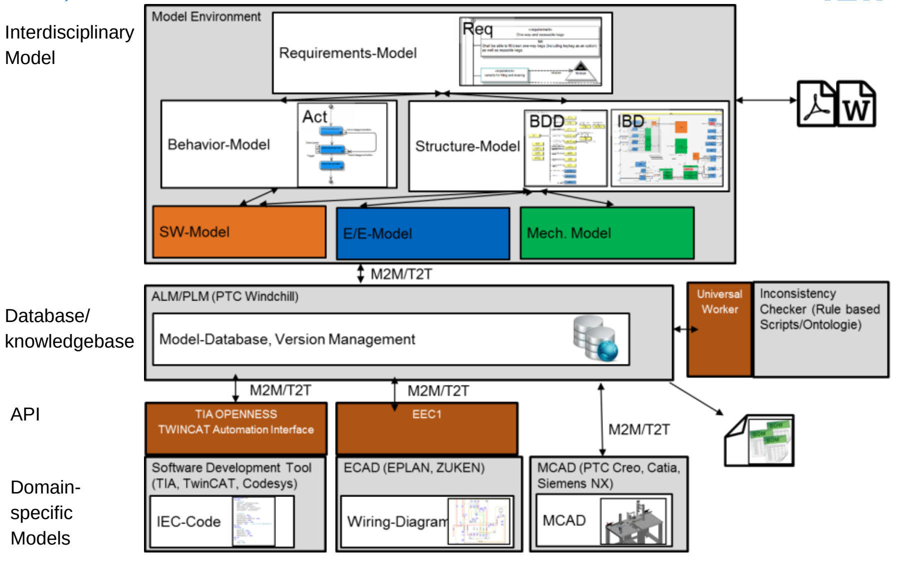
Application-specific

- Maximum mass
- Maximum energy consumption
- …

Source: Feldmann et al., CIRP CMS, 2014

Automation and
Information Systems
Technical University of Munich

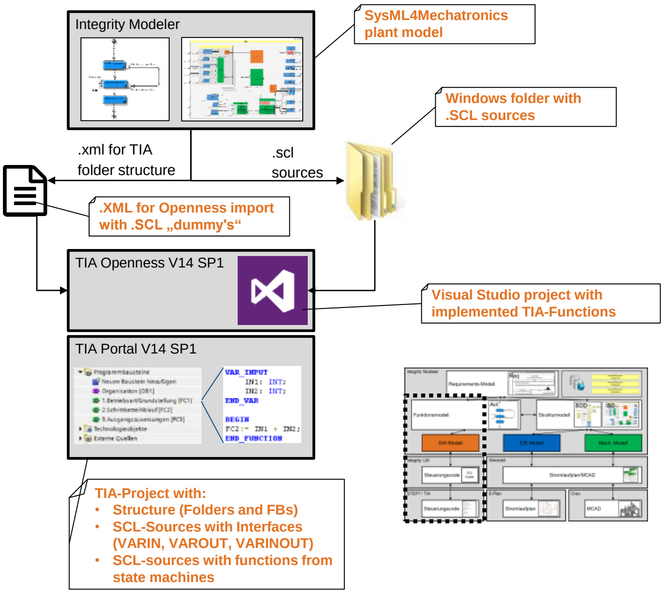
Structure T3



Automation and
Information Systems
Technical University of Munich

Use-Case: Code generation with the PTC-Toolchain

- SysML4Mechatronics model including behavior and structure
- Export of .XML-File for the Visual Studio Openness
- Generation of folder structure with empty .SCL dummy's
- Generation of .SCL sources with defined interfaces (VAR IN/OUT/INOUT) and functions
- Overwriting the dummy .SCL-Sources

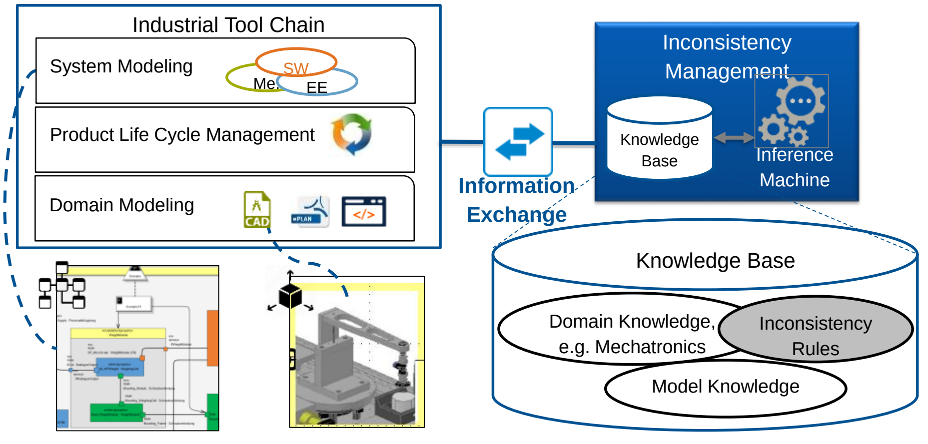


Automation and
Information Systems
Technical University of Munich

Integrated Inconsistency Management

Example: Coupling MCAD and SysML4Mechatronics in the tool chain
- Components changed in MCAD

→ Question: Configurations in SysML4Mechatronics still consistent?



Automation
and Information Systems
Technical University of Munich

Outline of part II

1. SysML4Mechatronics to model cross-disciplinary relations
2. Formal methods: OCL to model constraints
3. Formal methods: inconsistency management based on ontologies

Automation
and Information Systems
Technical University of Munich

Introduction and Motivation
Why should we deal with inconsistencies?

Examples for inconsistencies and their consequences

Mars Climate Orbiter
Costs: approx.
$ 125 Mio.

Source: http://en.wikipedia.org/wiki/Mars_Climate_Orbiter

Ariane 5 Flight 501
Costs: approx.
$ 500 Mio.

Source: http://www.lrr.in.tum.de/~walterm/ariane5.jpg

Denver Int'l Airport
Costs: approx.
$ 560 Mio.

Source: http://calleam.com/WTPF/wp-content/uploads/2008/12/denverbag5.jpg

→ **Assumption #1**: Handling inconsistencies early saves money

Alternatives to deal with inconsistencies

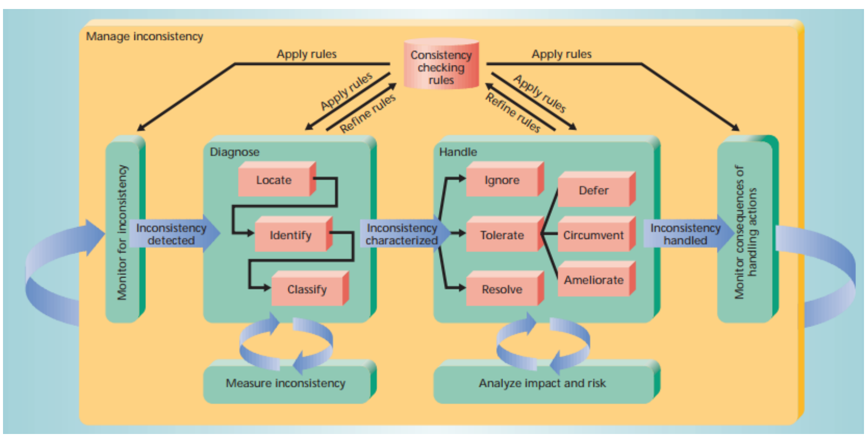
Handle inconsistencies
- Ignore inconsistencies — costly
- Deal with inconsistencies
  - Avoid inconsistencies — time-consuming
  - Identify and resolve inconsistencies ✓

→ **Assumption #2**: Continuously identifying and resolving inconsistencies ensures quality and saves time and money

Automation
and Information Systems
Technical University of Munich

Introduction Inconsistency Management

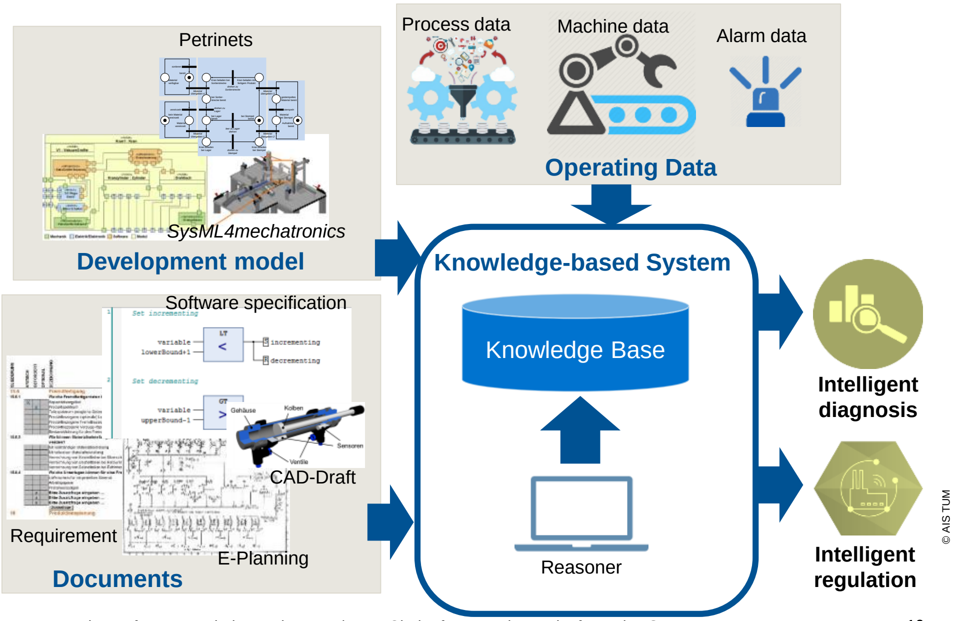
Inconsistency: any situation in which a set of descriptions does not obey some relationship that should hold between them



Source: Nuseibeh, Bashar, Steve Easterbrook, and Alessandra Russo. "Leveraging inconsistency in software development." Computer 33.4 (2000): 24-29.

Automation
and Information Systems
Technical University of Munich

Motivation: Knowledge-based systems in the context of intelligence in automation engineering



Univ.-Prof. Dr.-Ing. Birgit Vogel-Heuser| TUM Chair of Automation and Information Systems 19

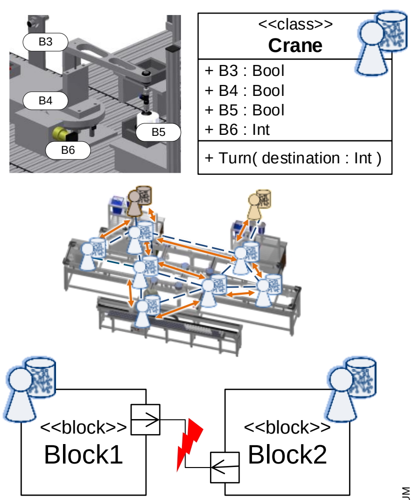
Automation
and Information Systems
Technical University of Munich

Motivation: Applications of Conclusion in Automation Engineering

Diagnosis agent:
- Detection of incorrect states and incorrect system configurations
- E.g. „If B3 and B4 get *true*, there is an incorrect state."

Intelligent control agents
- Identification of executable services by management agents
- E.g. „ If a workpiece is on the convoyer, it can be transported."

Agent-based consistency check
- Recognition of inconsistent model constructs
- e.g. „ Each port of a SysML block must be connected consistently."



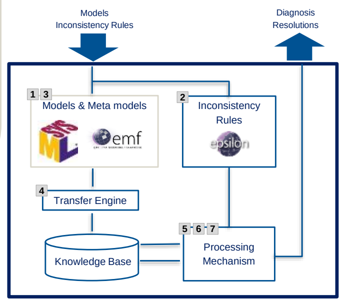
→ Suitable knowledge base of the agents necessary to realize the scope of application

Univ.-Prof. Dr.-Ing. Birgit Vogel-Heuser| TUM Chair of Automation and Information Systems 20

Automation
and Information Systems
Technical University of Munich

Example 1: Inconsistency Management in the PSSycle AG

In the course of the planning of the new product line of PSSycle AG different models were developed in the different planning phases. These relate, for example, to the product requirements, the product functions or the specific product components. Shortly before the start of the production process, PSSycle AG noticed that inconsistencies between the models are likely to be found due to incorrect data transmission. For example, the requirements for the battery range in the models are assigned to different values. Variable naming also makes cross-model communication difficult.



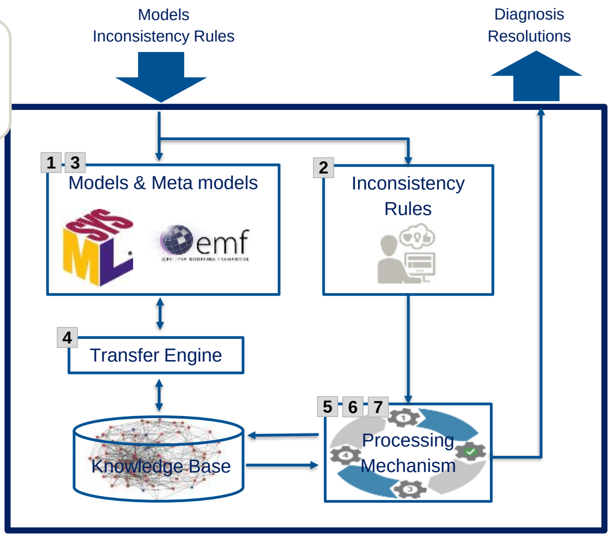
© AIS TUM

Automation
and Information Systems
Technical University of Munich

Example 1: Inconsistency Management in the PSSycle AG

The aim of PSSycle AG is now to identify and resolve these inconsistencies. For this, proceed according to the procedure of [1].

1	Build Metamodels
2	Define inconsistency rules in Metamodels
3	Define links between models
4	Transfer models into general representation
5	Inconsistency Checking
6	Inconsistency Diagnosis
7	Suggest resolution

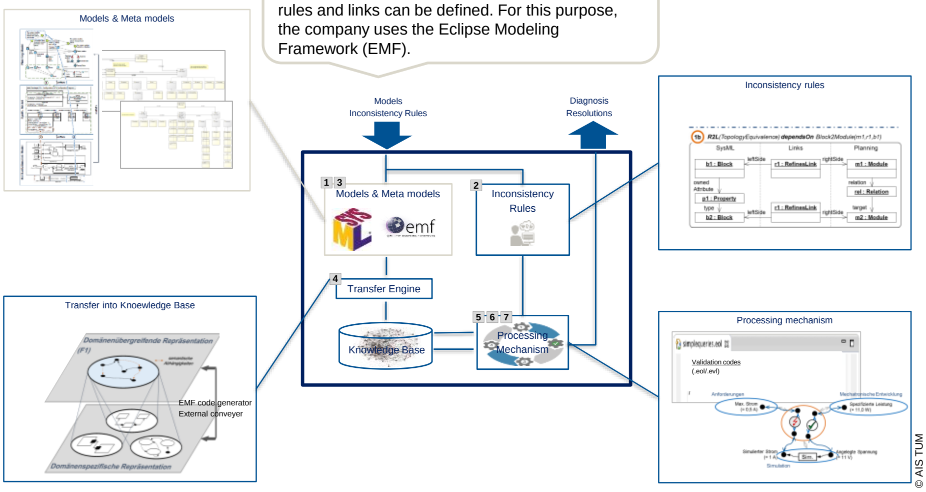


© AIS TUM

Automation
and Information Systems
Technical University of Munich

Example 1: Inconsistency Management in the PSSycle AG

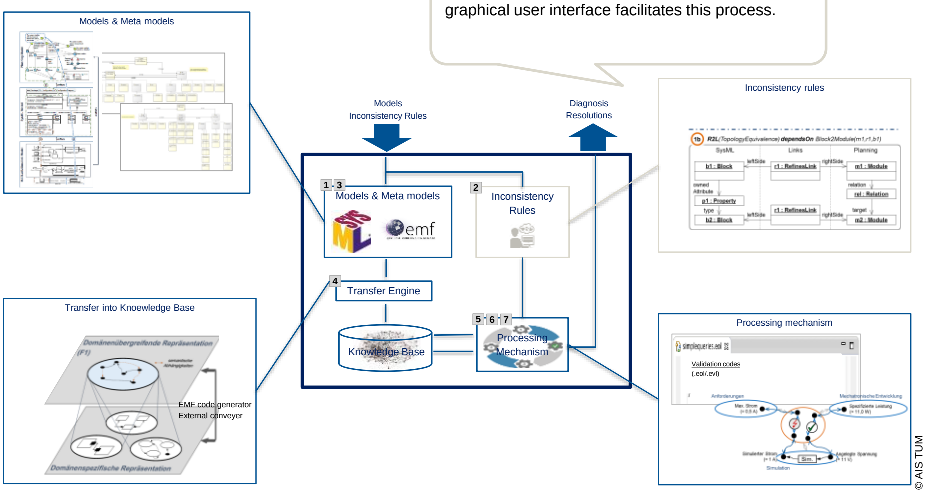
First of all, consistent metamodels are generated for the models. In these metamodels, inconsistency rules and links can be defined. For this purpose, the company uses the Eclipse Modeling Framework (EMF).



Automation
and Information Systems
Technical University of Munich

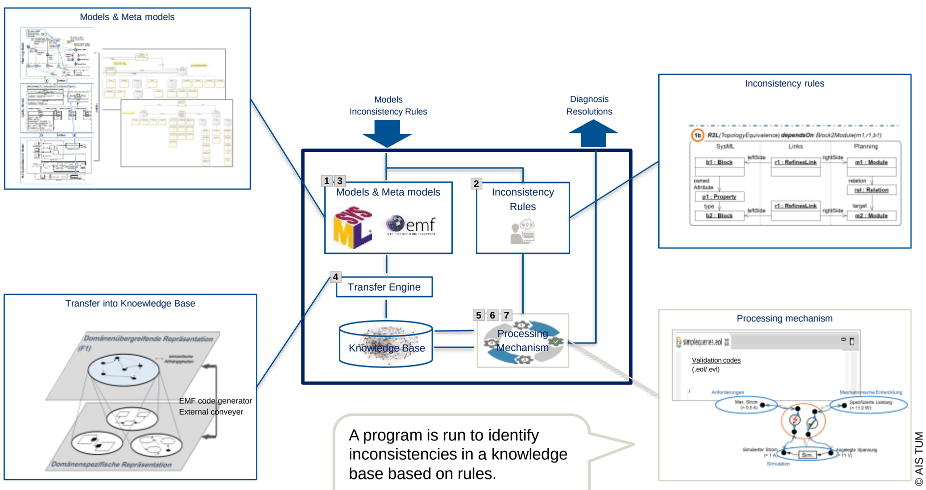
Example 1: Inconsistency Management in the PSSycle AG

Rules for consistencies are entered manually. A graphical user interface facilitates this process.



Automation
and Information Systems
Technical University of Munich

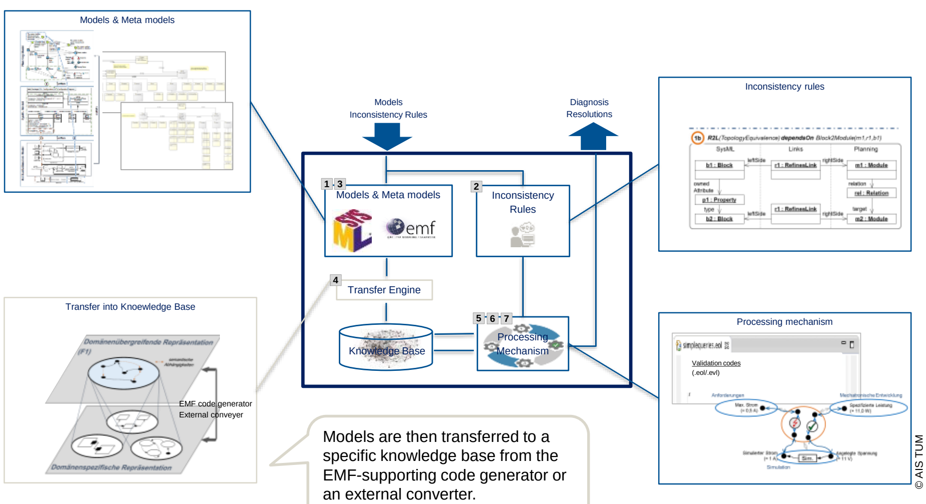
Example 1: Inconsistency Management in the PSSycle AG



A program is run to identify inconsistencies in a knowledge base based on rules.

Automation
and Information Systems
Technical University of Munich

Example 1: Inconsistency Management in the PSSycle AG



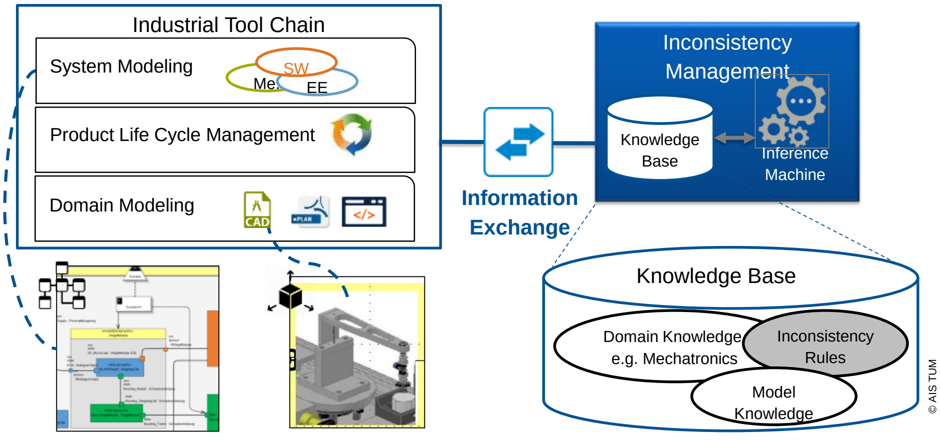
Models are then transferred to a specific knowledge base from the EMF-supporting code generator or an external converter.

Automation
and Information Systems
Technical University of Munich

Integrated Inconsistency Management

Example: Coupling MCAD and SysML4Mechatronics in the tool chain
- Components changed in MCAD

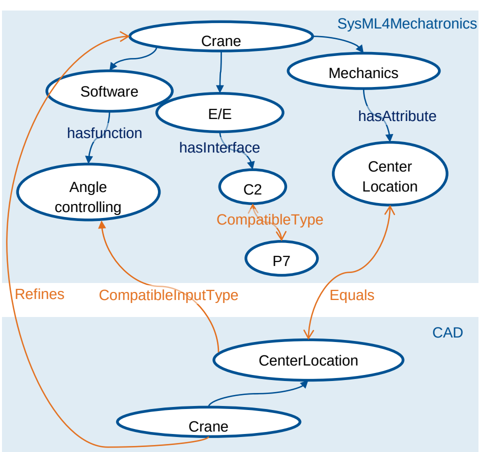
→ Question: Configurations in SysML4Mechatronics still consistent?

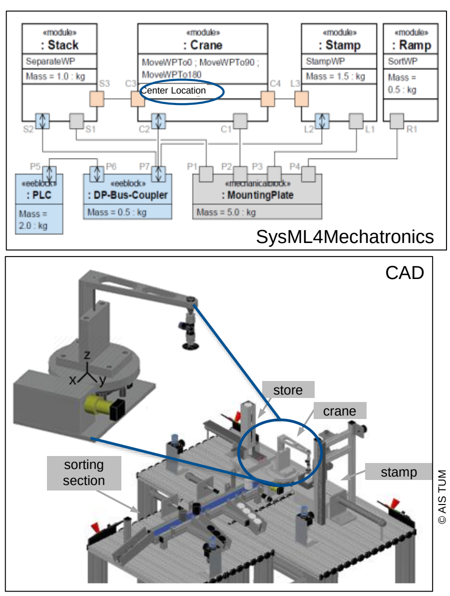


Automation
and Information Systems
Technical University of Munich

Approach: Ontology-Based Inconsistecy Checking

- Allows intra- and inter-model checking
- Allows intra- and interdisciplinary checking
- Checks different types of inconsistencies:
  - Sizes, function parameters, interfaces, names, requirements…





Automation
and Information Systems
Technical University of Munich

Outline of part II

1. SysML4Mechatronics to model cross-disciplinary relations
2. Formal methods: OCL to model constraints
3. Formal methods: inconsistency management based on ontologies

Automation
and Information Systems
Technical University of Munich

Introduction of Object Constraint Language (OCL) to model knowledge of sequences allowed

OCL as a formal language

- Software language to specify conditions for UML
- Easy to read
- Pure expression language, no change in the original model
- No programming language, i.e. aspecially
  - No formulation og program logic or control flow
- Typed language
  - Each expression in OCL has a particular type
  - Each OCL expression must use the correct type (e.g. no comparisson of strings and integers)
  - Status of the object is not changed during the validation

Application of OCL: Specification of

- Invariants in class diagrams
- Pre- and postcondotion for
- Conditions in sequence and state diagrams
- Condition of the UML metamodel

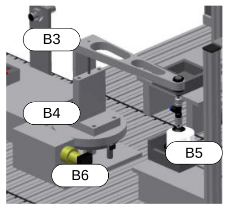
Automation
and Information Systems
Technical University of Munich

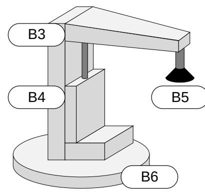
Interactive exercise – material collection by the crane

Formulate the following conditions for the crane:
Because of construction conditions the crane is only allowed to move in a rotation angle of $0° < \varphi \leq 360°$.
The pre- and postconditions at the bearing for the collection of material by the crane (Crane::GrabMaterial()) are:

- **Precondition**: Crane unloaded, lowered, at horizontal bearing position ($\varphi = 90°$)
- **Postcondition**: Crane unloaded, lowered, at horizontal bearing position ($\varphi = 90°$)
- The pre- and postconditions for the rotation of an angle X (Crane::Turn(X)) of the crane are:
- **Precondition:** Crane is not allowed to leave the angle range
- **Postcondition:** Crane is at a new position

Sensor	description	Data type
B3	Lift cylinder at top position	Boolean
B4	Lift cylinder at bottom position	Boolean
B5	Vacuum gripper loaded	Boolean
B6	Rotation angle of the rotary base	Integer





Automation
and Information Systems
Technical University of Munich

Interactive exercise – material collection by the crane

Formulate the following conditions for the crane:

- Because of construction conditions the crane is only allowed to move in a rotation angle of $0° < \varphi \leq 360°$.

- The pre- and postconditions at the bearing for the collection of material by the crane (Kran:: Grab Material()) are:
  - **Precondition**: Crane unloaded, lowered, at horizontal bearing position ($\varphi = 90°$)
  - **Postcondition**: Crane loaded, lowered, at horizontal bearing position ($\varphi = 90°$)

- The pre- and postconditions for the rotation of an angle X (`Crane::Turn(X))` of the crane are:
  - **Precondition:** Crane isn‘t allowed to leave the angle range
  - **Postcondition:** Crane is at a new position

Automation
and Information Systems
Technical University of Munich

Interactive exercise – material collection by the crane

Formulate the following conditions for the crane:

- Because of construction conditions the crane is only allowed to move in a rotation angle of $0° < \varphi \leq 360°$.

```
context Crane
inv: B6 > 0 and B6 <= 360
```

- The pre- and postconditions at the bearing for the collection of material by the crane (Kran::Grab Material()) are:
  - **Precondition**: Crane unloaded, lowered, at horizontal bearing position ($\varphi = 90°$)
  - **Postcondition**: Crane loaded, lowered, at horizontal bearing position ($\varphi = 90°$)

```
context Crane::GrabMaterial()
pre:    not B5 and not B3 and B4 and B6 = 90
post:   B5 and not B3 and B4 and B6 = 90
```

- The pre- and postconditions for the rotation of an angle X (Kran::Turn(x)) of the crane are:
  - **Precondition:** Crane isn't allowed to leave the angle range
  - **Postcondition:** Crane is at a new position

```
context Crane::Turn(X)
pre:    B6 + x > 0 and B6 + x <= 360
post:   B6 = B6@pre + x
```

Sensor	description	Data type
B3	Lift cylinder at top position	Boolean
B4	Lift cylinder at bottom position	Boolean
B5	Vacuum gripper loaded	Boolean
B6	Rotation angle of the rotary base	Integer

Automation
and Information Systems
Technical University of Munich

What for is the OCL or the temporal logic needed?

How can assumptions in a system model be considered related to its implementation?

In this context, what is the OCL?

What information does a model inspector need to test the requirements on an automated model?

In this context, what is the temporal logic for?

Automation
and Information Systems
Technical University of Munich

Outline of part II

1. SysML4Mechatronics to model cross-disciplinary relations
2. Formal methods: OCL to model constraints
3. Formal methods: inconsistency management based on ontologies

Automation
and Information Systems
Technical University of Munich

Application example: Intelligent Control Agents

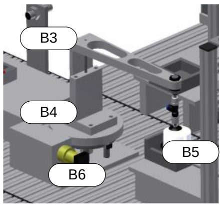
Question

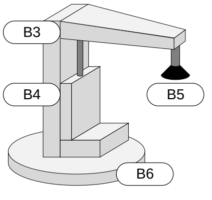
How can a control agent identify which operations are executable at the moment?

Application of ontologies

- An operation is executable if and only if its preconditions are fulfilled (cf. Petrinet, OCL)
- The precondition represents a quantity of states.

Example stamping module of the Pick & Place Unit (simplified)





Sensor	Description	Data type
B3	stroke cylinder at upper position	Boolean
B4	stroke cylinder at lower position	Boolean
B5	vacuum picker loaded	Boolean
B6	rotation angle of the basis	Integer

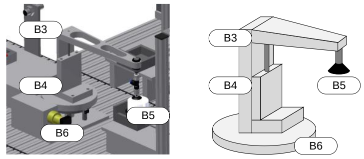
Precondition for the execution of the material grabbing at the bearing position

- Crane unloaded (**NOT B5**)
- Crane lowered (**NOT B3 AND B4**)
- Crane at horizontal bearing position (**B6 = 90**)

Application example: Intelligent Control Agents

Precondition for the execution of the material grabbing at the bearing position

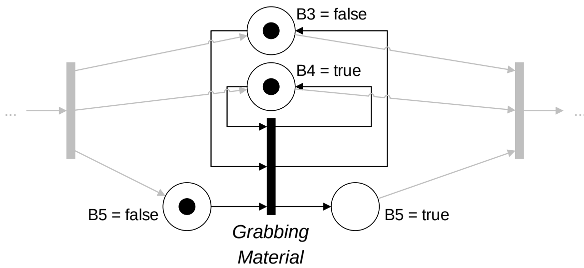
- Crane unload (**NOT B5**)
- Crane lowered (**NOT B3 AND B4**)
- Crane at horizontal bearing position (**B6 = 90**)



Analogy: Object Constraint Language (OCL)

```
context Crane::GrabMaterial()
pre:  (not B5) and (not B3 and B4) and (B6 = 90)
post: B5 and (not B3 and B4) and (B6 = 90)
```

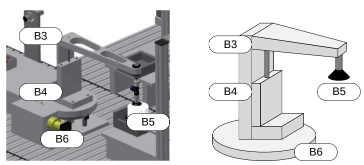
Analogy: Petrinet



Application example: Intelligent Control Agents

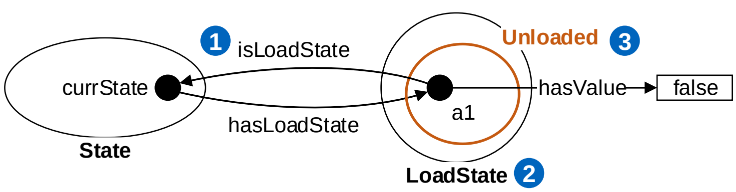
Precondition for the execution of the material grabbing at the bearing position

- Crane unload (**NOT B5**)
- Crane lowered (**NOT B3 AND B4**)
- Crane at horizontal bearing position (**B6 = 90**)



Formulation of background knowledge in OWL:

1. $\text{hasLoadState} \equiv \text{isLoadState}^{-}$ (Inverse relation)
2. $\text{LoadState} \equiv \exists \text{isLoadState.State}$ (Definition of the loading state)
3. $\text{Unloaded} \equiv \text{LoadState} \sqcap \exists \text{hasValue.}\mathit{false}$ (Meaning of „unload")



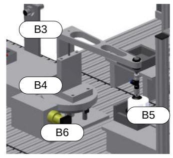
Further partial states are described and handled in an analog way.

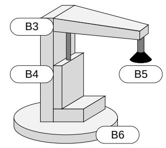
Automation
and Information Systems
Technical University of Munich

Application example.: Intelligent Control Agents

Precondition for the execution of the material grabbing at the bearing position

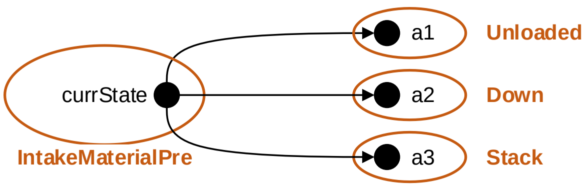
- Crane unload (**NOT B5**)
- Crane lowered (**NOT B3 AND B4**)
- Crane at horizontal bearing position (**B6 = 90**)





Formulation of the precondition for GrabMaterial:

$$\begin{aligned} \text{IntakeMaterialPre} \equiv\ & \exists \text{hasLoadState.Unloaded} \sqcap & \text{(Crane unloaded)} \\ & \exists \text{hasVerticalState.Down} \sqcap & \text{(Crane lowered)} \\ & \exists \text{hasHorizontalState.Stack} & \text{(Crane at horizontal bearing position)} \end{aligned}$$



→ GrabMaterial is possible (Inference mechanism **Realization**).

Automation
and Information Systems
Technical University of Munich

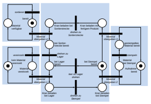
Declarative models and their processing: Rule-based models vs. Ontology-based models

Rule-based models	Ontology-based models
• Modeling paradigm: IF/THEN-rules and -facts → description of concrete situations	• Modeling paradigm: Description of quantities and their relations → Description of a class of situations
• Formalization of „simpler“ knowledge	• Formalization of „more complex“ knowledge
• Inference based on facts → Derivation of new facts based on known facts	• Inference is based on quantity descriptions → derivation of new groups
• Control of complexity by choosing the inference mechanism	• Control of complexity because of expressiveness of the ontology
• Difficult maintenance of big rule bases → size limitation of a rule base	• Simpler maintenance of big ontologies → bigger knowledge bases possible
• Consistency checks only by using additional instruments	• Automatic checks of consistency done by the reasoner

Automation
and Information Systems
Technical University of Munich

Summary – lessons learned

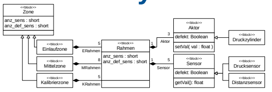
Petrinets



- Modeling of control processes
- Formal verification of process attributes

Knowledge: transition activation (pre-, postedges)

UML and SysML



- Modeling of (software) systems
- Graphical documentation

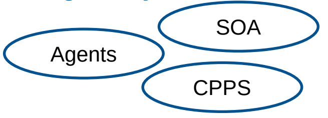
OCL

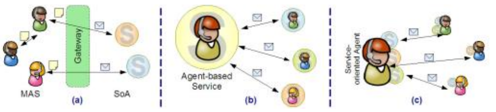
```
context Einlaufzone
inv:   ERahmen->forAll( r : Rahmen |
       r.Sensor->select(s |
          s.oclIsTypeOf(Drucksensor))->size()=3 and
       r.Sensor->select(s |
          s.oclIsTypeOf(Distanzsensor))->size()=2)
```

- Extension mechanism
- Formulation of boundary conditions, invariants, pre- and postconditions

Knowledge: correctness of the models, pre- and postconditions of the operations

Intelligent Systems





- Implementation of intelligent systems in the field of automation technology
- Agents, service-oriented architectures, Cyber-Physical Production Systems

Thank you for your attention!

Univ.-Prof. Dr.-Ing. Birgit Vogel-Heuser

Ordinaria
Automation and Information Systems (AIS)
Mechanical engineering,
Technische Universität München
www.ais.mw.tum.de; vogel-heuser@tum.de